MINISTERUL EDUCATIEI SI INVATAMINTULUI
INSTITUTUL POLITEHNIC "TRAIAN VUIA" TIMISOARA

FACULTATEA DE ELECTROTEHNICA

Ing. Stefan Holban

CONTRIBUTII LA GESTIUNEA RESURSELOR IN SISTEMELE DE CALCUL MULTIMICROPROCESOR

CONDUCATOR STIINTIFIC

Prof.dr.ing. Mircea Petrescu

1988

MINISTERUL EDUCATIEI SI INVATAMINTULUI
INSTITUTUL POLITEHNIC "TRAIAN VUIA" TIMISOARA

FACULTATEA DE ELECTROTEHNICA

Ing. Stefan Holban

CONTRIBUTII LA GESTIUNEA RESURSELOR IN SISTEMELE DE CALCUL MULTIMICROPROCESOR

CONDUCATOR STIINTIFIC

Prof.dr.ing. Mircea Petrescu

1988

CUPRINSUL TEZEI DE DOCTORAT

P R E F A T A

Definirea arhitecturii sistemelor multimicroprocesor validată prin implementarea unor sisteme de operare de maximă eficienţă aplicativă, reprezintă unul din domeniile de vîrf ale tehnicii de calcul. Progresele realizate în acest domeniu au adus un important suport teoretic şi aplicativ, făcînd posibilă concepereа şi elaborarea unor noi norme şi standarde.

In acest context abordarea problematicii referitoare la realizarea unor sisteme de operare multimicroprocesor constituie o modalitate de rezolvare nouă sub raportul gestiunii resurselor întregului sistem de calcul. Aceasta vizează atît aspectele impuse de arhitectură cît şi cele datorate aplicaţiei căreia îi este destinat sistemul de calcul. Diversitatea problemelor teoretice, specifice acestui vast domeniu, precum şi varietatea direcţiilor aplicative a impus cooperarea cu Institutul de Proiectări pentru Automatică Bucureşti, Centrul de Calcul al Institutului Politehnic Timişoara şi Institutul de Tehnică de Calcul Timişoara.

La finalizarea tezei sentimentul recunoştinţei ne adresează memoriei prof.dr.ing.Alexandru Rogojan, alături de care am făcut primii paşi în domeniul cercetării teoretice şi aplicative.

Sentimente de aleasă consideraţie şi profundă recunoştinţă exprim distinsului prof.dr.ing.Mircea Petrescu, care mi-a făcut onoarea de a conduce şi direcţiona cu înaltă competenţă cercetările întreprinse pînă la finalizarea tezei.

In perioada laborioaselor etape ale elaborării diverselor părţi din teză am beneficiat de sprijinul competent şi constant al prof.dr.ing.Vasile Pop, căruia îi exprim stimă, consideraţie şi sincere mulţumiri. Mulţumesc tov.dr.ing.Vasile Baltac pentru solicitudinea şi amabilitatea de care a dat dovadă pentru maniera competentă în care a analizat teza şi pentru observaţiile şi aprecierile pe care le-a formulat. Aduc mulţumiri de asemenea, conf.dr.ing. Strugaru Crişan, pentru atenţia şi competenţa cu care a analizat materialul tezei ca şi pentru recomandările şi aprecierile făcute.

Consideraţie şi aleasă preţuire exprim conducerii Institutului Politehnic Timişoara precum şi prof.dr.ing.Avram Heler pentru încurajările susţinute pe care mi le-a adresat, pentru sprijinul generos acordat în întreaga perioadă de elaborare a tezei.

Doresc să mulţumesc bunilor mei colegi şi prieteni ing.Viorel Coifan şi mat.Petru Pavel Mihai, care, în diverse perioade, au sprijinit realizarea unora din etapele aplicative necesare pregătirii tezei. Mulţumiri pentru sprijin şi colaborare adresez colegilor şi prietenilor dr.ing.Ionel Jian, dr.ing.Ioan Jurcă, dr.ing.V.Creţu , dr.ing.Dan Ciubotariu şi ing.Vancea Romul.

In fine, dar nu în ultimul rînd, sentimentul recunoştinţei şi profundei preţuiri se adresează memoriei tatălui meu lt.col.Valentin Holban care a veghiat în cursul anilor tineri ai formării mele.

Nu se nominalizează aici toate persoanele, colegi şi prieteni cercetători şi tehnicieni care, într-o formă sau alta au contribuit prin competenţă şi spirit de colaborare la realizarea diverselor proiecte de cercetare în cadrul elaborării tezei. Exprim tuturor sentimentul celei mai sincere recunoştinţe.

Stefan Holban

1. INTRODUCERE

1.1. Stadiul actual al rezolvării problemelor ridicate de gestiunea resurselor în sistemele de operare multimicroprocesor.

Dezvoltarea continuă şi rapidă a tehnologiei de realizare a circuitelor integrate pe scară largă şi foarte largă, a declanşat o serie de evenimente a căror impact asupra dezvoltării prezente şi viitoare a echipamentelor electronice şi a tehnicii de calcul este greu de evaluat şi de prevăzut. In cadrul acestor evenimente, apariţia microprocesorului a reprezentat evenimentul major, care prin impactul pe care l-a produs afectează toate domeniile vieţii economice şi sociale. In acest sens, utilizarea microprocesoarelor a condus la obţinerea unor rezultate remarcabile la construirea unor noi aparate de măsură şi control, în conducerea proceselor industriale, în telecomunicaţii, ca şi la declanşarea unui fenomen a cărui consecinţe în prezent sînt departe de a putea fi apreciate. Astfel, microprocesorul a făcut accesibilă tehnica de calcul omului obişnuit, nespecialistului, eveniment care poate deschide perspective de nebănuit în dezvoltarea societăţii umane.

Lucrarea de faţă, vizează domeniul de mare actualitate al utilizării sistemelor de calcul cu mai multe microprocesoare, dedicate unor aplicaţii industriale. Problematica abordată se referă la aspectele pe care le ridică în proiectarea şi implementarea sistemelor de operare, modul în care sînt gestionate resursele sistemului privite atît din punctul de vedere a arhitecturii sistemului, cît şi din cel al aplicaţiei căreia îi este destinat acesta.

Apariţia sistemelor multimicroprocesor (mμP), este rezultatul unei evoluţii firesti în dezvoltarea tehnicii de calcul. Oricît de rapide şi de performante devin microprocesoarele, vor apare întotdeauna aplicaţii a căror cerinţe sînt cu mult mai mari decît performanţele unui singur microprocesor. Un microprocesor nu poate fi întotdeauna suficient de rapid pentru prelucrarea datelor în timp real sau nu poate asigura un timp suficient de răspuns care să satisfacă anumiţi utilizatori. In trecut astfel de aplicaţii erau întîrziate pînă în momentul dezvoltării unui microprocesor nou, mai rapid, care însă odată utilizat mărea substanţial costul aplicaţiei. Alternativa care s-a impus a fost dată de interconectarea a două sau mai multe

microprocesoare care să răspundă cerinţelor aplicaţiei, reprezentînd astfel o alternativă pînă în momentul în care apare un microprocesor mai rapid şi mai performant. Un exemplu privind necesitatea şi utilitatea sistemelor mμP se impune în acest moment, deşi acesta nu se referă la domeniul tehnicii de calcul. Cînd s-a pus problema realizării unor avioane mai rapide, prezenţa unui singur motor ar fi făcut imposibilă realizarea acestui deziderat. Soluţia alternativă, ar fi constat în aşteptarea construirii unui motor mai puternic sau în utilizarea ca în prezent, a două sau mai multe motoare de aceiaşi putere. In urma acestei paralele, mesajul rezultat este că datorită naturii aplicaţiilor sistemele mμP vor fi necesare întotdeauna.

- Datorită dezvoltării rapide a tehnologiei, arhitectura sistemelor mμP în special, a devansat cu mult dezvoltarea unor sisteme de operare destinate acestor tipuri de sisteme. Acest aspect s-a datorat faptului că trecerea la o arhitectură în care apar mai multe procesoare face inoperante multe din soluţiile adoptate în sistemele de operare dedicate unor sisteme monoprocesor. Motivul se datoreşte faptului că de data aceasta, resursele întregului sistem nu mai sînt gestionate de un singur microprocesor, ci de două sau mai multe microprocesoare. Utilizîndu-se experienţa cîştigată în domeniul sistemelor de operare monoprocesor, s-au cristalizat o serie de idei şi concepte de proiectare pentru sisteme cu mai multe microprocesoare. Tehnica multiprogramării pentru un singur procesor s-a extins şi pentru sisteme mμP. In cadrul acesteia, multiprogramarea a fost gîndită ca o împărţire a procesoarelor pe programe sau părţi de programe. Alte tehnici cu caracter experimental mult mai complexe, asigură alocarea primului microprocesor disponibil programului care urmează să fie executat. Intre aceste două extreme există o multitudine de variante, datorate diversităţii aplicaţiilor concrete care pot apărea, care în cele mai multe cazuri solicită un anumit mod de rezolvare a gestiunii resurselor sistemului. Aceste aspecte, au condus în final la sisteme de operare din cele mai diverse în care rezolvarea problemelor legate de gestiunea resurselor este particulară unei arhitecturi şi aplicaţii date. Cu toate acestea, s-a încercat şi s-a reuşit definirea unor linii directoare, care au condus la apariţia unor sisteme de operare ca de exemplu IMMX8o, iRMX realizate de firma INTEL pentru sisteme mμP slab şi tare cuplate. Aceste sisteme de operare nu reprezintă decît o structură de bază care trebuie particularizată funcţie de tipul aplicaţiei, şi deci în ultimă instanţă de modul în care se desfăşoară cooperarea la alocarea resurselor sistemului.

1.2. Descrierea generală a lucrării.

Analiza bibliografică întreprinsă privind realizările şi rezultatele obţinute pe plan mondial şi naţional în domeniul gestiunii resurselor realizate de sistemele de operare destinate sistemelor de calcul μP, scot în evidenţă faptul că în prezent sînt insuficient abordate problemele legate de proiectarea şi implementarea unor astfel de sisteme de operare în care problematica legată de gestiunea resurselor sistemului, văzut ca un tot unitar, să apară cu claritate. Considerînd resursa unui sistem μP ca reprezentînd orice componentă a arhitecturii sistemului de calcul, sau a sistemului de operare implementat, care într-o etapă sau alta de funcţionare a sistemului devine obiectul unei cereri de alocare din partea a una sau a mai multor din componentele aceleiaşi structuri a sistemului μP, s-a reuşit stabilirea unor concepte de taliere şi structurare a unui sistem μP corespunzătoare unei aplicaţii date. In contextul acestei definiţii şi a scopului propus s-au stabilit următoarele obiective:

(a) Definirea resurselor arhitecturii unui sistem de calcul μP şi sistematizarea cunoştinţelor privind mecanismele care sînt implicate în gestionarea lor.
(b) Stabilirea resurselor generale ale unui sistem de operare destinat implementării pe un sistem μP. Determinarea particularităţilor care apar funcţie de tipul aplicaţiei căreia îi este destinat sistemul, ca şi sistematizarea cunoştinţelor privind mecanismele care sînt implicate în gestiunea acestei categorii de resurse.
(c) Determinarea modului în care cele două tipuri de resurse cooperează în contextul sistemului μP văzut ca un tot unitar.
(d) Stabilirea unui model global care să reprezinte într-o manieră coerentă resursele şi mecanismele definite la punctele a, b şi c.
(e) Furnizarea unei metodologii de lucru utile proiectantului de sisteme de operare μP în concepereа şi alegerea celei mai bune structuri a nucleului sistemului de operare μP, particulară unei aplicaţii date, care să conducă la obţinerea celor mai bune performanţe.
(f) Exemplificarea criteriilor de alegere şi proiectare propuse prin realizarea experimentală a nucleului unui sistem de operare μP destinat unei aplicaţii cu caracter general în care arhitectura sistemului este realizată cu module MULTIPROM.

Pornind de la aceste considerente lucrarea abordează într-o manieră originală din punctul de vedere al proiectantului de sisteme de operare, problematica legată de stabilirea structurii sistemului de operare, funcție de modul în care este rezolvată problema gestiunii resurselor sistemului multimicroprocesor.

Obiectivele propuse au condus la structurarea lucrării într-un număr de șapte capitole. Primele trei sînt destinate atingerii obiectivelor propuse la punctele a, b și c iar următoarele trei la definirea criteriilor de proiectare și a descrierii modului de aplicare a acestor criterii la realizarea practică a unui nucleu de sistem de operare μP. Numărul mare de capitole s-a datorat complexității deosebite a problemei, ca și a necesității obținerii unei vederi de ansamblu asupra unui fenomen care înglobează practic tot ceea ce reprezintă în ultimă instanță din punct de vedere arhitectural și al sistemului de operare un sistem de calcul μP.

In capitolul doi se efectuează o analiză detaliată a arhitecturii unui sistem μP privită din punctul de vedere a resurselor pe care le pune la dispoziție o astfel de structură. Se definesc și se analizează principalele resurse care intervin: modulele procesoare, memoria comună, magistralele sistemului, etc. De asemenea se acordă o importanță deosebită modului în care aceste resurse se pot interconecta în cadrul diverselor tipuri de aplicații, punîndu-se accent pe acele structuri care se întîlnesc în aplicațiile cele mai frecvente. Un spațiu mare este rezervat mecanismelor de cooperare între aceste resurse sînt prezentate modurile în care sistemele μP implementează mecanismele de generare a întreruperilor, respectiv modul în care aceste mecanisme se utilizează pentru stabilirea dialogului între componentele arhitecturii sistemului. Un aspect important al cooperării, reprezentat de mecanismul de protecție a unei resurse atunci cînd aceasta este utilizată, reprezintă de asemenea obiectul unei analize atente, fiind prezentate soluțiile adoptate în prezent de diversele tipuri de sisteme. Deoarece dialogul presupune existența unui schimb de informații cît și prezența unui suport adecvat se efectuează un studiu comparativ al diverselor tipuri de magistrale utilizate în prezent, subliniindu-se faptul că acestea reprezintă un element cheie în cadrul unei arhitecturi μP. Afirmația se susține prin faptul că modulele unui sistem μP s-au dezvoltat în jurul unor tipuri de magistrale standard. Modul în care se rezolvă conflictele la alocarea unor resurse comune constituie de asemenea obiectul unei analize aprofundate. Aceasta se concre-

tizează prin prezentarea tehnicilor de arbitrare: serială, paralelă cu autoselecţie, analiza fiecăreia în parte efectuîndu-se pe baza unui graf de stare. Capitolul se încheie printr-o scurtă trecere în revistă a protocoalelor utilizate în cursul dialogului dintre componentele arhitecturii sistemului mμP.

Capitolul trei este dedicat analizei structurii unui sistem de operare mμP şi a funcţiilor pe care acesta trebuie să le realizeze. Analiza se efectuează din punctul de vedere a resurselor de care dispune sistemul de operare, punîndu-se îndeosebi accentul pe fenomenele de concurenţă şi cooperare care apar între procese în timpul execuţiei acestora. Capitolul începe cu prezentarea mecanismelor utilizate în cadrul cooperării care apare în timpul execuţiei între procese. Se evidenţiază particularităţile pe care le prezintă în cadrul unui mμP conceptele de semafor, regiune critică şi operaţie indivizibilă. In acest context, se analizează în detaliu, prin reprezentarea în pseudocod a modului de funcţionare, primitivele WAIT şi SIGNAL a căror prezenţă este fundamentală în definirea procedurilor de acces la o resursă, indiferent de tipul acesteia. Deoarece prin modul de localizare, resursele pot fi locale sau globale, se scot în evidenţă deosebirile care apar în modurile de acces la un tip sau alt tip de resursă. Un mμP, prin însăşi noţiunea de multitudine de procese divers localizate pe care o reprezintă, în care procesele colaborează la realizarea unei anumite aplicaţii, impune stabilirea unor modalităţi de realizare a dialogului, atît între procesele prezente în aceiaşi memorie locală a unui modul procesor cît şi între procesele prezente în memoriile locale a unor module procesoare diferite. Aceste mecanisme sînt analizate în cadrul relaţiei producător / consumator, fiind prezentate cele două primitive fundamentale ATTACH şi DETACH utilizate. Ca şi în cazurile anterioare se acordă o atenţie specială modului în care acest mecanism este utilizat în contextul sincronizării execuţiei proceselor implicate în dialog prezente în întregul sistem mμP. In capitol se face o trecere gradată de la mecanismul fundamental de comunicare care utilizează primitivele ATTACH şi DETACH la un mecanism general de comunicare. Ca primitive de bază ale acestui mecanism, se analizează şi se prezintă SEND şi RECEIVE ca şi diversele variante care apar în practica curentă. O parte importantă a analizei este rezervată proceselor de tratare a întreruperilor ca şi implicaţiile pe care le au acestea asupra mecanismelor de cooperare şi comunicare cu procese de acelaşi tip sau cu procese utili-

zator obişnuite. Complexitatea fenomenului de gestiune a cererilor adresate unei resurse comune, a generat în sistemele de operare concurente monoprocesor, apariţia unor proceduri standard denumite monitoare care preiau asupra lor întreaga responsabilitate a procesului de gestiune. Fiecare din aceste tipuri de monitoare se analizează din punctul de vedere al aplicabilităţii în gestiunea globală a unei resurse dintr-un sistem μP. Analiza care se efectuează pe baza grafului de funcţionare a fiecărui tip de monitor în parte, scoate în evidenţă implicaţiile pe care le prezintă o arhitectură de tip μP asupra modului de utilizare a acestora. Se prezintă deosebirile care apar în definirea structurilor de date a unui monitor, a primitivelor care asigură tranziţia dintr-o stare în alta a monitorului, ca şi în modul de concepere a interfeţelor de intrare / ieşire cu procesele care apelează un monitor. Ultima parte a capitolului este rezervată descrierii nucleului unui sistem de operare μP. Acesta este privit ca un ansamblu care încorporează toate structurile şi mecanismele prezentate anterior în acest capitol. Abordarea din acest punct de vedere a structurii unui nucleu μP dă posibilitatea evidenţierii relaţiilor de interdependenţă care se stabilesc între componentele ce definesc resursele arhitecturii unui μP şi procedurile care definesc la rîndul lor resursele unui sistem de operare μP. Modul în care aceste componente cooperează sau intră în concurenţă unele cu celelalte ca şi mecanismele implicate sînt privite ca formînd pentru o aplicaţie dată un tot unitar. Acest "tot unitar" a cărei analiză şi descriere a reprezentat scopul capitolului trei se consideră că defineşte funcţia generală de gestiune a unui sistem de operare μP. În contextul acestui punct de vedere, capitolul se încheie cu prezentarea şi analiza principalelor primitive ale unui nucleu de operare μP.

In capitolul patru se prezintă şi se analizează legătura care se stabileşte între sistemul μP şi caracteristicile aplicaţiei căreia îi este destinat. O dată stabilite funcţiile şi modul de realizare a mecanismelor unui μP, apare problema modului de localizare a sistemului de operare în cadrul arhitecturii impuse de o anumită aplicaţie. Intregului sistem astfel realizat i se impun realizarea anumitor indici de performanţă, care sînt ceruţi de aplicaţia căreia îi este destinat sistemul. Din analiza bibliografică efectuată, au rezultat un număr de patru mari categorii de nuclee de sisteme de operare μP. Acestea sînt nuclee de tip: monolitic, partiţionat, distribuit şi cu interogare. Fiecare din aceste categorii reprezintă de fapt acelaşi tip de nucleu standard, singurul aspect care le deosebeşte fiind mo-

dul în care procedurile şi structurile de date sînt amplasate în cadrul unei arhitecturi date. Analiza care s-a efectuat devine necesară, deoarece pînă în prezent nu a existat nici un criteriu obiectiv care să justifice alegerea unei structuri de nucleu sau a altei structuri. Din acest motiv, analiza care se efectuează scoate în evidenţă aspectele care deosebesc nu numai tipurile de nucleu respective, ci şi variantele deosebit de numeroase care pot apare în cadrul unei categorii sau a alteia. Pentru aceste variante, în cazurile în care reprezintă abateri de la nucleul standard al unui sistem mμP, se scot în evidenţă deosebirile, şi în măsura în care devine necesar se prezintă tipurile noi de primitive în pseudocod şi structurile de date noi care apar. In cadrul analizei nu se pierde nici un moment din vedere modul în care, în cadrul fiecărui tip de nucleu este rezolvată problema gestiunii resurselor. Acest punct de vedere, dă posibilitatea definirii în lucrare a unei împărţiri a categoriilor de nuclee existente în două mari clase de nuclee de sisteme de operare mμP: cu restricţii şi fără restricţii în gestiunea resurselor sistemului. Este de remarcat faptul că împărţirea în cauză nu este rigidă, între cele două clase fiind prezente variante (care aparţin celor patru moduri de localizare) care rezolvă într-o măsură mai mare sau mai mică problema ridicată de gestiunea resurselor.

Analiza care s-a efectuat în capitolele doi, trei şi patru reprezintă suportul teoretic care utilizat în capitolul cinci a dat posibilitatea definirii unor criterii de proiectare a unui nucleu de sistem de operare mμP, axat pe o anumită arhitectură şi aplicaţie dată. Rezultatele obţinute se bazează pe construirea unui model a gestiunii resurselor într-un sistem mμP. Modelul permite reprezentarea componentelor arhitecturale ale sistemului impuse de o aplicaţie cît şi a componentelor nucleului sistemului de operare ales de utilizator. Datorită complexităţii deosebite, impuse de asamblarea într-un tot unitar a tuturor acestor aspecte, studiul teoretic a condus la definirea a două clase de modele. Prima clasă corespunde nucleelor cu restricţii de sincronizare, iar cea de a doua celor fără restricţii de sincronizare. Cele două clase de modele în care se utilizează tehnica lanţurilor Markov dau posibilitatea ca pe baza datelor furnizate de arhitectura sistemului impusă de natura aplicaţiei şi a tipului de sistem de operare ales, considerat de proiectant optim, să se obţină o serie de indici de performanţă care caracterizează comportarea globală a sistemului mμP. Rezultatele care se obţin prin modelare pot sau nu

confirma justeţea soluţiilor alese de utilizator privind structura nucleului sistemului de operare ales. Pentru a se asigura un caracter de generalitate cît mai mare, cele două clase de modele utilizează în calcule mărimi relative, aspect care permite ca rezultatele obţinute să reprezinte indicaţii directe privind modul de reconfigurare a sistemului de operare, astfel încît să se obţină performanţele maxime posibile. Indicaţiile furnizate se referă la modul optim în care se pot amplasa procesele, fie în memoriile locale ale modulelor procesoare fie în memoria comună a sistemului μP. Un alt rezultat oferit utilizatorului, constă în obţinerea unei indicaţii privind tipul de nucleu de operare care este cel mai potrivit pentru aplicaţia luată în considerare. De asemenea rezultatele permit optimizarea mecanismelor de coordonare şi sincronizare globală a execuţiei proceselor prin definirea unei lungimi optime a mesajelor implicate în dialogul dintre componentele nucleului pe de o parte, şi între procesele prezente în sistemul μP pe de altă parte. Capitolul se încheie cu o serie de recomandări privind atît alegerea unei arhitecturi optime cît şi a celui mai corespunzător tip de sistem de operare pentru aplicaţia căreia îi este destinat sistemul μP.

Capitolele şase şi şapte reprezintă o aplicare a rezultatelor teoretice obţinute în lucrare, în cadrul unei aplicaţii practice de realizare a unui nucleu de operare μP. Nucleul este destinat unui sistem realizat cu module standard din familia MULTIPROM. În prima parte a capitolului şase se face o analiză a magistralei MULTIPROM şi a modulelor construite în jurul acestei magistrale. Analiza devine necesară, deoarece iniţial acestea nu au fost gîndite a fi utilizate ca părţi componente a unei arhitecturi μP. În urma analizei au rezultat a serie de concluzii care au constituit tot atîtea propuneri înaintate beneficiarului contractului, de îmbunătăţire a structurii modulelor în cauză, astfel încît acestea să răspundă cerinţelor impuse de o arhitectură μP. Cea de a doua parte a capitolului este destinată alegerii nucleului sistemului de operare cel mai adecvat. Se propune o structură de tip nucleu monolitic distribuit ca fiind cea mai potrivită, alegerea justificîndu-se prin prisma rezultatelor teoretice obţinute în cadrul lucrării.

În capitolul şapte se descrie modul concret în care s-a realizat nucleul monolitic distribuit. Modul de realizare a nucleului se efectuează gradat, astfel încît să se furnizeze prin acest exemplu practic o metodologie de construire şi implementare cu caracter general.

2. STUDIUL CARACTERISTICILOR ARHITECTURII SISTEMELOR MULTIMICROPROCESOR.

Istoric,primele sisteme multiprocesor au fost realizate utilizînd componente discrete,avînd ca elemente constitutive sisteme de calcul convenţionale. Acest aspect a influenţat concepţia şi modul de realizare a sistemelor cu procesoare multiple şi a determinat un anumit mod specific de abordare a problemelor. Apariţia microprocesoarelor a schimbat însă radical concepţia de realizare a acestei clase de sisteme. Astfel sistemele cu mai multe microprocesoare au preluat parţial sau total toate problemele clasice legate de structura unui astfel de sistem pe de o parte,iar pe de altă parte au generat altele noi.

Prezentul capitol este destinat scoaterii în evidenţă a problemelor noi ridicate de implementarea unui microprocesor într-o structură cu mai multe procesoare şi a modului în care s-a preluat experienţa cîştigată în construirea sistemelor multiprocesor cu sisteme convenţionale.

2.1. Caracteristicile unui calculator monoplacă. Module procesoare.

Scara largă de integrare atinsă în tehnica circuitelor integrate a permis amplasarea unui microcalculator pe o singură placă de circuit imprimat. In general un astfel de microcalculator include o unitate centrală de 8,16 sau 32 de biţi,memorie RAM/EPROM locală, circuite de interfaţă cu magistrala sistemului,logică de manipulare a întreruperilor ca şi resurse locale de I/E.

Plecînd de la această structură,firma Inttel a proiectat o clasă specială de module procesoare denumită SBC (Single Board Computer), ce pot funcţiona singular sau în cooperare cu alte SBC-uri. Denumirea unanim acceptată în terminologia românească este de calculator monoplacă. Deoarece aceste SBC-uri le vom trata ca posibile părţi componente a unui sistem cu mai multe procesoare,privindu-le deci mai mult din punct de vedere al posibilităţilor funcţionale şi mai puţin de structură şi organizare în sine,în cele ce urmează se va utiliza numele general de modul procesor (MP) şi nu de calculator monoplacă,deoarece în cadrul unei structuri multiprocesor modulul procesor reflectă mai bine caracteristica de unitate de calcul independentă care poate fi interconectată cu alte unităţi de aceiaşi

natură,cu unităţi de memorie sau echipamente de I/E.

Structura unui MP depinde de un număr mare de factori ca topologia de interconectare,modul de comunicare inter MP-oare,ca şi de numărul şi tipul de resurse utilizate în comun [We77] . In general MP include un microprocesor,memorii pentru instrucţii (ROM) şi date (RAM),circuite de adresare a memoriei,o logică de programare a întreruperilor ca şi circuite de interfaţă pentru controlul echipamentelor proprii de I/E [**84] . Este posibil ca fiecare MP să conţină o cantitate de memorie care să fie utilizată în comun cu alte MP [BD81] .Dacă o memorie care conţine instrucţii de program este partajată atunci se adaugă suplimentar o memorie tampon (cunoscută sub numele de memorie cache) în care se stochează cele mai utilizate instrucţii. In cazul în care magistrala sistemului este mai mare decît lungimea unui cuvînt se mai adaugă o memorie stivă pentru instrucţii în scopul optimizării încărcării magistralei. O componentă critică a unui MP este interfaţa cu magistrala sistemului multimicroprocesor (mμP). Performanţele şi flexibilitatea topologiei de interconectare sînt direct proporţionale cu complexitatea şi performanţele acestei interfeţe logice. Dacă sistemul mμP este alcătuit din MP realizate cu microprocesoare diferite,atunci interfaţa va trebui să fie unică pentru fiecare unitate componentă în parte (în particular MP pe 8 sau 16 biţi pot comunica pe aceiaşi magistrală,necesitînd însă interfeţe distincte).

O altă componentă critică a unui MP o reprezintă sistemul de translatare a adreselor de program în adrese fizice,atît pentru memoria proprie cît mai ales pentru memoria utilizată în comun. Acest aspect este impus de faptul că pe de o parte programele care se execută concurent pot solicita:

- o cantitate de memorie care de obicei este mai mare decît memoria proprie a unui MP. In acest caz trebuie să se asigure o translatare dinamică a adreselor pentru zonele de memorie disponibile din memoria proprie a unui MP.
- utilizarea unor zone de date comune prezente în memoria altor MP-oare sau într-o unitate de memorie comună distinctă. Referinţele la acest tip de date,va necesita translatarea adreselor în adrese fizice proprii unităţilor în cauză.

O posibilitate de simplificare a acestui sistem de translatare a adreselor,cunoscut sub numele de " memory map ",o constituie realizarea unor module soft care să preia parţial sau total aceste funcţii. Aceasta pe de o parte simplifică problema,dar pe de altă parte lungeşte timpul de adresare,mai ales pentru microprocesoarele care nu au im-

plementată posibilitatea de adresare indirectă (de exemplu INTEL 8080,8085 etc.),în contextul în care cantitatea de memorie cerută creşte.

In concluzie,în cele ce urmează,prin MP vom înţelege un microcalculator pe o singură placă (de exemplu SBC-8080,SBC-8085,SBC-280, SBC-6800,s.a.m.d.,a cărui structură de microcalculator este fie standard,stabilită de firma care a dezvoltat microprocesorul component, fie particulară stabilită de un proiectant pentru o aplicaţie specifică) şi în care interfaţa cu magistrala sistemului μP şi logica de translatare a adreselor depinde de tipul de magistrală utilizat,de topologia sistemului ca şi de aplicaţia căreia îi este destinat sistemul μP.

2.2. Tipuri de topologii de interconectare multimicroprocesor.

Sistemele μP fac parte din clasa sistemelor cu arhitectură MIMD,apărînd atît sub formă sistemelor distribuite,cît şi sub forma sistemelor multiprocesor propriu zise. Sistemele μP cu această organizare (date şi instrucţii multiple) sînt destinate executării paralele a unor taskuri independente care utilizează atît date proprii cît şi rezultate a executării anterioare a altor taskuri,ca şi a combinării rezultatelor execuţiilor a două sau mai multe taskuri în scopul obţinerii de date pentru alte taskuri.

2.2.1. Interconectarea modulelor procesor.

Se utilizează un număr de trei modalităţi de interconectare cu magistrală comună,de tip " crossbar " şi cu porţi multiple.

Sisteme multimicroprocesor cu magistrală comună.

Constau din MP,memorii unităţi de I/E conectate pe o magistrală comună. Modul de interconectare se caracterizează prin:

- complexitatea funcţională scăzută,deoarece fiecare MP şi echipament necesită numai o singură interfaţă de conectare la magistrală;
- posibilitatea adăugării simple de noi module procesor la configuraţie;
- preţ de cost scăzut.

Accesul la magistrală se efectuează sub controlul unui arbitru

de magistrală care pe baza unei scheme de prioritate rezolvă conflictele cauzate de cererile simultane de acces la resurse [Ho82] . Gradul de interferenţă dintre cererile MP la magistrală,depinde de lungimea şi frecvenţa ciclurilor de magistrală a unui MP,de durata unui ciclu de memorie sau I/E,ca şi de numărul de MP-oare care utilizează magistrala. Cea mai scăzută valoare a raportului dintre numărul de cicluri de magistrală cerut de un MP şi numărul total de cicluri disponibile va reprezenta cea mai înaltă capacitate de trecere a sistemului muP. Din acest motiv,utilizarea în cadrul unui MP a unor memorii şi unităţi de I/E reprezintă un factor care măreşte capacitatea de trecere (evident însă că aceasta este limitată superior de capacitatea de trecere a magistralei).

Dezavantajul major al acestei structuri de interconectare constă în aceea că adăugarea suplimentară de noi unităţi complică logica de arbitrare şi micşorează capacitatea de trecere a sistemului,ca şi faptul că un defect al magistralei conduce la căderea sistemului muP.

Sisteme multimicroprocesor cu interconectare " crossbar ".

Sînt sisteme muP în care fiecare MP,memorie şi echipament de I/E este conectat la o magistrală separată (fig.2.1.). In cadrul acestei organizări sistemul poate fi privit ca o mulţime de magistrale care se intersectează formînd o reţea şi în care direcţia de transfer a informaţiei este specificată de comutatorul prezent în fiecare punct de intersecţie de pe magistrală. Sistemul de interconectare de tip "crossbar " este caracterizat de:

- capacitate mare de transfer a datelor,datorită posibilităţii de conectare a fiecărui bloc de memorie la orice MP sau unitate de I/E,ceea ce conduce la o capacitate de trecere foarte mare;
- interfaţa cu magistrala a unităţilor active (MP sau unităţi de I/E) ca şi a celor pasive (memorii) este simplă deoarece nu apar probleme legate de rezolvări de conflicte de acces la magistrală sau de definire a sursei şi destinaţiei în procesul de transmitere a datelor;
- complexitatea mare a sistemului de interconectare,datorită complexităţii hardware a comutatoarelor. De exemplu,în reţeaua de interconectare a sistemului H 4400 (8 MP şi 16 module de memorie realizat de firma Hughes) un comutator conţine de 2,5 ori mai multe componente decît un MP [En76] . Pentru a reduce complexitatea reţelei de interconectare în prezent se recurge la o soluţie de separare a echipamentelor de I/E de reţeaua principală şi de con-

struire a unei reţele suplimentare de interconectare pentru acestea din urmă.

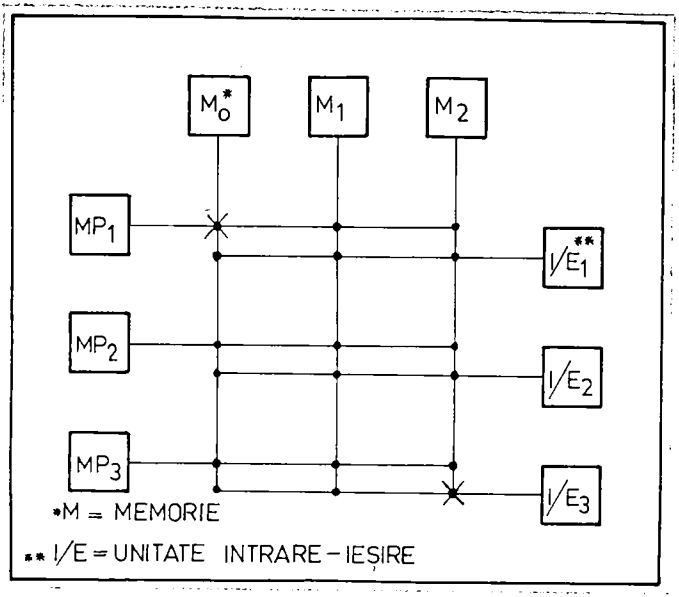


Fig.2.1. Interconectare " crossbar " a unui sistem mμP.
Sînt notate cu X trei conexiuni simultane.

Sisteme mμP cu porţi multiple.

Sînt sisteme mμP în care MP sînt legate la fiecare bloc de memorie şi unitate de I/E prin intermediul unităţilor de interfaţă ale acestora din urmă [En76] , [We77] . Acest concept în care unităţile de interfaţă asigură logica de comutare a MP din reţea poate fi privit ca un mod de implementare a conceptului de interconectare " crossbar " prezentat anterior (fig.2.2.). Principalele caracteristici se referă la:

- deoarece fiecare interfaţă trebuie să aibă un număr de porturi de I/E corespunzător cu numărul MP-oare din sistem, proiectarea acestora devine complicată în cazul în care numărul devine prea mare;

- logica de arbitrare este amplasată în interfeţele blocurilor de memorie şi unităţilor de I/E. Aceasta face ca una din cele mai acute probleme ridicate de acest mod de interconectare să îl constituie modul de rezolvare a conflictelor în cazul prezenţei mai multor cereri de acces (aspectul este valabil şi pentru interfeţele MP-oare în cazul în care mai multe unităţi de I/E solicită acelaşi MP). O soluţie frecvent adoptată în acest caz, constă în acordarea priorităţii acelui MP sau unităţi deI/E ca-

re este legat direct. Experienţa a arătat că pentru a evita degradarea performanţelor în cazul în care numărul MP-oare este mai mare ca 4,o soluţie optimă constă în echiparea cu memorii proprii a MP-oare, aceasta în scopul păstrării în aceste memorii a informaţiei care este utilizată cel mai frecvent,recurgerea la memoria comună efectuîndu-se numai cînd memoria proprie devine insuficientă.

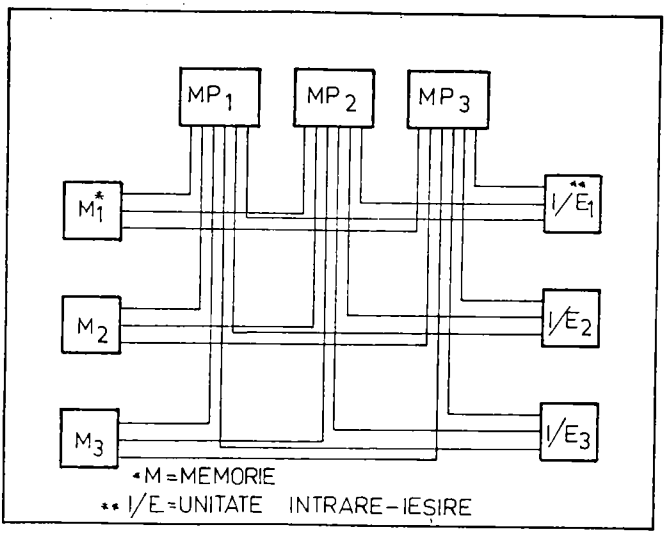


Fig.2.2. Interconectare în sistemul cu porţi multiple a unui sistem μP. Configuraţia sistemului de magistrale este caracterizată de conectarea permanentă între fiecare element al sistemului ceea ce permite simultaneitatea operaţiilor pe magistrală,pentru fiecare MP.

Plecînd de la aceste trei configuraţii de interconectare se pot obţine prin combinare alte posibilităţi de interconectare. De exemplu, o structură de magistrală multiplă comună poate fi obţinută ca o combinaţie de magistrale comune şi un sistem de porţi multiple s.a.m.d. In prezent pentru simplitate cele mai multe sisteme μP utilizează fie o magistrală comună,fie o interconectare în sistemul memorie cu porţi multiple (două sau patru porţi),configuraţia depinzînd în mare măsură de aplicaţia căreia îi este destinat sistemul μP.

2.2.2. Sisteme multimicroprocesor cuplate slab şi tare.

In cadrul clasei de sisteme μP cuplate slab,intră sistemele μP în care fiecare MP are propria sa memorie şi echipamente periferice, comunicînd cu fiecare alt MP prin linii de date seriale. Aceste linii

pot fi locale (în interiorul aceleiaşi camere sau clădiri) sau telefonice [Da82] .

Avantajul important al acestei clase de sisteme μP,constă în aceea că aplicaţia de executat poate fi împărţită. Fiecare MP este practic un microcalculator independent care este mai mult sau mai puţin autonom. Numai în cazul în care în cadrul aplicaţiei este necesară punerea în comun a unor resurse care nu sînt disponibile local se recurge la o procedură de transfer de date sau fişiere.

Fig.2.3. prezintă un sistem μP cuplat tare. Fiecare MP este un microcalculator complect (CPU,memorie ROM şi RAM,interfaţă consolă şi magistrală μP),comunicînd cu un alt MP prin intermediul unei memorii comune. Pentru exemplul dat, [Co78] suplimentar,sistemul μP dispune de un ansamblu de linii cu denumirea HILIN (High LiNe),cu rol de a interconecta între ele mai multe sisteme μP,magistrala propriu zisă a sistemului μP purtînd numele EXBUS (EXternal BUS). In cadrul unui sistem μP cuplat tare,MP poate fi realizat cu μP diferite (referindu-ne tot la exemplul dat s-au utilizat Zilog Z 8o şi Signetic 265o). Si în acest caz aplicaţia este divizată în taskuri,care se pot executa simultan în paralel,deosebirea faţă de sistemele cu-

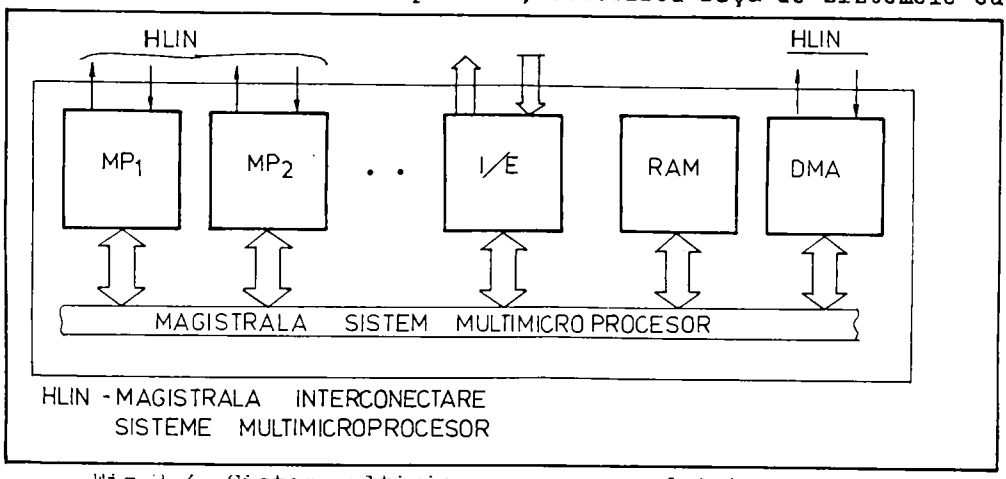


Fig.2.3. Sistem multimicroprocesor cuplat tare.

plate slab,apărînd ca urmare a faptului că schimbul de informaţie între taskuri sînt mult mai numeroase ceea ce necesită apelul frecvent la utilizarea unor resurse comune care reprezintă în ultimă instanţă un mediu intermediar de vehiculare pentru astfel de date.

Reţele multimicroprocesor.

In contextul sistemelor μP,reţeaua apare ca o multitudine de sisteme μPcare sînt interconectate în scopul descentralizării resurse-

lor. Aceste reţele pot fi alcătuite din orice combinaţie de sisteme μP cuplate slab sau tare. In fig.2.4. se prezintă o astfel de reţea,în care sistemele μP formate din MP-oare cuplate tare (structura lor a fost prezentată în fig.2.3.) sînt slab cuplate prin intermediul unor linii de comunicaţie serială [Co78] .

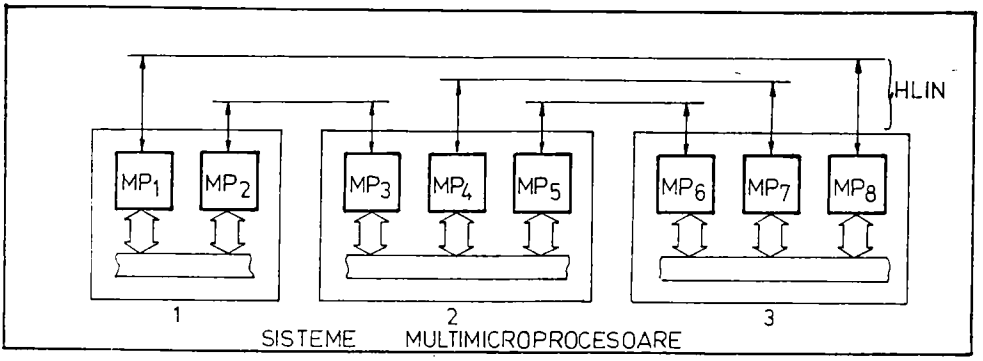


Fig.2.4. Reţea de sisteme multimicroprocesor.

Avantajul acestui mod de interconectare constă în:

- Fiecare MP este un microcalculator complect care poate funcţiona independent într-o arhitectură μP de tip MIMD.
- Complecta modularitate este dată de faptul că nu există nici o limită pentru extinderea sistemului prin adăugarea de module identice (acestea pot fi fie MP-oare care se adaugă la un μP existent,fie un μP complect care se adaugă la reţea.)
- Reţeaua rămîne în funcţiune în cazul defectării unui MP sau chiar a unui μP (aceasta datorită modului de interconectare a μP-oarelor).

2.2.3. Ierarhii organizatorice în sistemele multimicroprocesor.

Intr-un μP existenţa unui fenomen de concurenţă la ocuparea resurselor comune ca şi necesitatea ca unul sau mai multe MP să-şi asume un rol de coordonare a activităţilor a impus necesitatea stabilirii unei ierarhii organizatorice. In cadrul acesteia,apar unităţi master (acestea sînt MP-oare) care pot participa la procesul de ocupare a unei resurse comune (unităţi de I/E,memorii comune),cît şi unităţi slave (acestea sînt MP-oare dedicate unei anumite activităţi) cărora le este interzis accesul la ocuparea unei resurse comune. Din punctul de vedere al desfăşurării activităţilor într-un μP pot apare unul sau mai mulţi masteri care colaborează între ei şi care pot coordona activitatea a una sau mai multe unităţi slave în cadrul a două ierarhii orga-

nizatorice:

- master / slave;
- master / master.

Sisteme master / slave.

Sînt alcătuite dintr-un MP cu numele master (conducător) avînd rolul general de a superviza întreaga activitate care se desfăşoară în sistem şi din unul sau mai multe MP numite slave (subordonate), care realizează fiecare anumite sarcini specifice în cadrul sistemului. Toate MP slave comunică cu un singur master,care acţionează fie ca un concentrator sau distribuitor de informaţii a taskurilor,care sînt executate de fiecare MP subordonat [CC81] . Organizarea master / slave impune o ierarhie rigidă componentelor sistemului (fig. 2.5.). Sistemul se pretează bine pentru situaţiile în care există un volum mare de cereri de I/E şi în general în cazurile în care este necesară prelucrarea distribuită a unui volum mare de informaţie care este schimbat cu exteriorul. Aplicaţii specifice sînt considerate sistemele de control a proceselor industriale,sistemele de achiziţie a datelor etc. In aceste aplicaţii MP slave efectuează o prelucrare parţială a informaţiilor de I/E,primite de la MP master sau trimise acestuia. Informaţiile sînt transferate direct între componentele sistemului utilizînd de obicei o procedură bazată pe întreruperi.

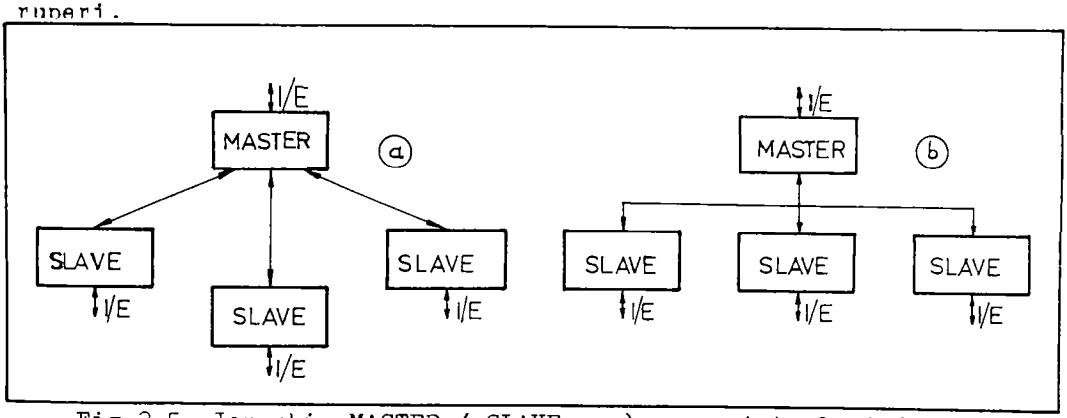


Fig.2.5. Ierarhie MASTER / SLAVE a) cu magistrale individuale
b) cu magistrală comună.

Numărul MP slave din sistem depinde atît de natura aplicaţiei cît şi de capacitatea MP master de a răspunde în timp util cererilor emise de componentele sistemului. Dacă această condiţie nu este înde-

plinită,atunci capacitatea de trecere şi gradul de utilizare a sistemului mμP sînt mici. De asemenea dacă MP master este supraîncărcat cu executarea sarcinilor legate de gestiunea sistemului,pot apare situaţii în care MP slave să fie obligate să aştepte un timp suplimentar atunci cînd cer un dialog cu MP master. Din acest motiv,eficienţa unui sistem mμP cu ierarhie master / slave,depinde în mare măsură de eficienţa sistemului de operare (SO) implementat.

Sisteme master / master.

Sînt sisteme care asigură un grad mare de independenţă pentru fiecare MP (fig.2.6.). Transferurile de informaţie sînt mai puţin frecvente ca în sistemele master / slave,ele fiind determinate de fiecare MP în parte. In cadrul sistemului fiecare MP execută o activitate specifică,avînd posibilitatea de a comunica cu un alt MP,de obicei prin intermediul unei memorii comune [Ra75] , [CC81].

Sistemele astfel ierarhizate se împart în două mari categorii după modul de executare a taskurilor în MP-oare:

- sisteme cu executarea separată a taskurilor în fiecare MP;
- sisteme de prelucrare simetrică sau omogenă în toate MP-oare.

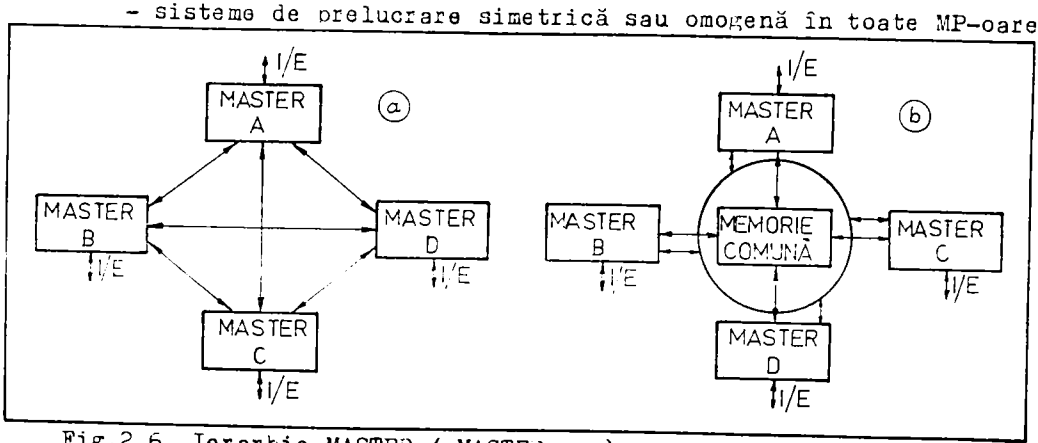


Fig.2.6. Ierarhie MASTER / MASTER a) cu magistrale separate
b) cu memorie comună

In prima categorie fiecare MP execută o anumită activitate specifică. De exemplu,MP care controlează dispozitivele de măsură dintr-un sistem de testare. In cea de a doua categorie,pe măsură ce este necesară executarea anumitor taskuri,se recurge la distribuirea acestora la MP disponibile (care în această situaţie trebuie să fie identice). Acest ultim tip de ierarhie se aplică îndeosebi sistemelor de calcul mμP. Avantajul utilizării eficiente a MP-oare (spre deosebire de primul caz,în care unele MP-oare pot fi foarte încărcate şi altele mai

puţin utilizate,funcţie de natura şi specificul problemei rezolvate), este anulat în bună măsură de complexitatea SO care trebuie să dispună de funcţii de planificare şi gestiune a taskurilor în scopul lansării lor în execuţie.

2.3. Controlul resurselor unui sistem multimicroprocesor.

Taskurile care sînt executate de un mμP utilizează în comun atît resurse hard (MP-oare,memorii,unităţi de I/E,magistrale) cît şi soft (variabile,fişiere,zone de date,etc.).

La aceste resurse pot apare ca urmare a utilizării lor comune mai multe cereri simultan,sau cereri mai prioritare decît cea care utilizează resurse la un moment dat. Acest aspect ridică problema complicată de rezolvare a conflictelor de prioritate la alocarea unei resurse,cît şi de protejare a ei în intervalul de timp cît este utilizată (aspectul este valabil atît pentru resursele hard cît şi pentru cele soft). Modul incorect de abordare poate avea ca urmare, fie obţinerea unei capacităţi mici de trecere,fie o blocare a sistemului ca urmare a punerii în şir de aşteptare a cererilor (emise fie de unităţile hard,fie de taskurile sistemului),la alocarea unor resurse care au fost deja alocate. In prezent,în sistemul mμP pentru asigurarea controlului asupra resurselor comune,cît şi pentru protecţia acestora,se utilizează combinat atît tehnici hard,cît şi soft. Tehnicile hard folosite sînt de trei categorii şi anume: testare indicatori de stare,întreruperi şi arbitrare,utilizate în marea majoritate a cazurilor combinat. Primele două se vor discuta în acest paragraf,cea de a treia în paragraful următor,destinat analizei magistralelor mμP.

2.3.1. Intreruperi.

Intreruperile pot fi utilizate pentru:

- tratarea erorilor (paritate,depăşirea limitelor de adresare);
- semnale de ceas (depăşirea unui interval de timp,ceasul de timp real);
- gestiunea unităţilor de I/E (recunoaştere unitate I/E, transfer de date terminat etc.);
- sincronizarea comunicaţiilor între MP-oare.

Intreruperile din primele două clase,sînt utilizate de microprocesorul existent în MP fiind utilizate de obicei pentru operaţii legate de corectitudinea transferului de date în şi din memoria proprie a MP, de lansarea în execuţie a unor taskuri,mai ales în cadrul unor aplicaţii de timp real. In acest ultim caz,generarea întreruperilor de ceas poate fi efectuată fie individual,fie de un arbitru hard centralizat, utilizîndu-se o linie de întrerupere comună care leagă toate MP-oare. In cadrul fiecărui MP,tratarea acestui tip de întrerupere depinde de destinaţia sistemului mμP.

Intreruperile utilizate pentru gestiunea unităţilor de I/E,permit lansarea transferurilor de date şi controlul modului în care acestea se desfăşoară,atît pentru unităţile de I/E proprii,cît şi pentru cele comune. Dacă utilizarea întreruperilor pentru gestiunea unităţilor proprii unui MP nu ridică probleme dificile,existînd circuite specializate pentru toată gama de μp-oare existente (de exemplu pentru μP-oarele dezvoltate de INTEL 8o8o,8o86,8o85,circuite specializate de tipul 8214,8259 sau 8259 A),care asigură priorităţi fixe,mascate controlabile inclusiv prin taskurile rezidente,utilizarea întreruperilor folosite pentru gestiunea unităţilor de I/E comune reprezintă o problemă dificilă. In prezent pentru gestionarea acestei clase de întreruperi se utilizează două metode [We77] funcţie de numărul unităţilor comune de I/E.

- Dacă numărul unităţilor de I/E comune este mare,se recurge la un comutator centralizat. Practic acesta este un MP destinat numai pentru astfel de operaţii. Acesta efectuează integral tratarea întreruperii primind sau transferînd date de la sau către un MP,al sistemului mμP prin intermediul unei resurse comune,evident dacă este cazul. Tehnica prezintă avantajul implementării unor algoritmi soft de tratare a întreruperilor,care sînt grupaţi centralizat ca şi a utilizării unui număr de linii de întrerupere mic. Dezavantajul constă în faptul că se încarcă sistemul mμP prin adăugarea unui MP suplimentar ca şi faptul că în anumite aplicaţii acesta trebuie să utilizeze un μP mai rapid decît cele existente pe celelelalte MP.

- Dacă numărul unităţilor de I/E este mic,caz în care se poate presupune că întreruperile sînt de cîteva tipuri cu o frecvenţă de sosire relativ mică,se recurge la o metodă descentralizată,care presupune o schemă hard simplă cu rol de memorare a stării întreruperii şi de directare a acesteia spre primul MP disponibil. Pe de altă parte dacă un MPcare este într-o procedură de tratare a unei

întreruperi, solicită emiterea unei întreruperi, aceasta poate fi preluată de primul MP disponibil. Această metodă este mai puţin eficace ca cea centralizată deoarece presupune o periodicitate a semnalelor de întrerupere, faptul că acestea apar la intervale de timp relativ mari ca şi datorită faptului că toate MP trebuie să conţină aceleaşi taskuri de tratare. Metoda are avantajul că nu necesită un MP dedicat acestei operaţii, deoarece după cum s-a văzut tratarea este distribuită tuturor MP.

Intreruperile din ultima clasă sînt utilizate pentru sincronizarea comunicaţiilor între MP-oare. In prezent se utilizează un număr de trei tehnici.

- Prima presupune utilizarea unei linii de cerere de întrerupere care să lege fiecare MP cu fiecare MP din sistemul mμP. Este cel mai simplu mod, avînd însă marele dezavantaj că necesită un număr mare de linii, aspect care face aplicabilă tehnica numai în sistemele mμP cu un număr mic de MP (două sau trei).

- A doua tehnică utilizează o procedură de transfer a cererii de întrerupere de la un MP la alt MP pînă în momentul în care cererea este recunoscută de MP căruia îi este destinată. Tehnica are avantajul utilizării unui singur nivel de întrerupere şi a posibilităţii definirii unui vector de priorităţi la nivelul fiecărui MP. Dezavantajul constă în faptul că necesită amplasarea unor taskuri identice de tratare în fiecare MP ca şi faptul că modul de rezolvare conduce la întîrzieri în transmiterea unei cereri (ca urmare a parcurgerii secvenţiale a MP).

- A treia tehnică utilizează o procedură soft de generare hard a cererilor de întrerupere. Acest aspect o face să fie cea mai utilizată, motiv pentru care se prezintă mai detaliat [FV81]. In cadrul acestui mod de sincronizare a comunicaţiilor fiecare MP este conectat la un nivel de întrerupere. Conectarea poate fi făcută la o parte sau la toate nivelele magistralei de întrerupere. De asemenea la această magistrală sînt conectate ieşirile unor porturi de I/E (port de generare a întreruperilor) care pot fi adresate de fiecare MP din sistem prin intermediul magistralei de adrese şi date (fig.2.7.). La apariţia unui semnal de întrerupere, acesta va fi tratat de fiecare MP în parte care este conectat pe nivelul respectiv de întrerupere a magistralei, (schema din fig.2.7., utilizează circuite INTEL).

De exemplu, pentru configuraţia din fig.2.7., MP2 poate iniţia o întrerupere pe nivelul 2, înscriind în portul de generare a între-

ruperilor configuraţia oo,o4,oo (se înscrie în bitul 2). Impulsul care apare pe linia de întrerupere INT_2 la care este conectată unitatea de control a întreruperilor de MP1,va fi văzut ca un semnal de întrerupere şi tratat ca atare.

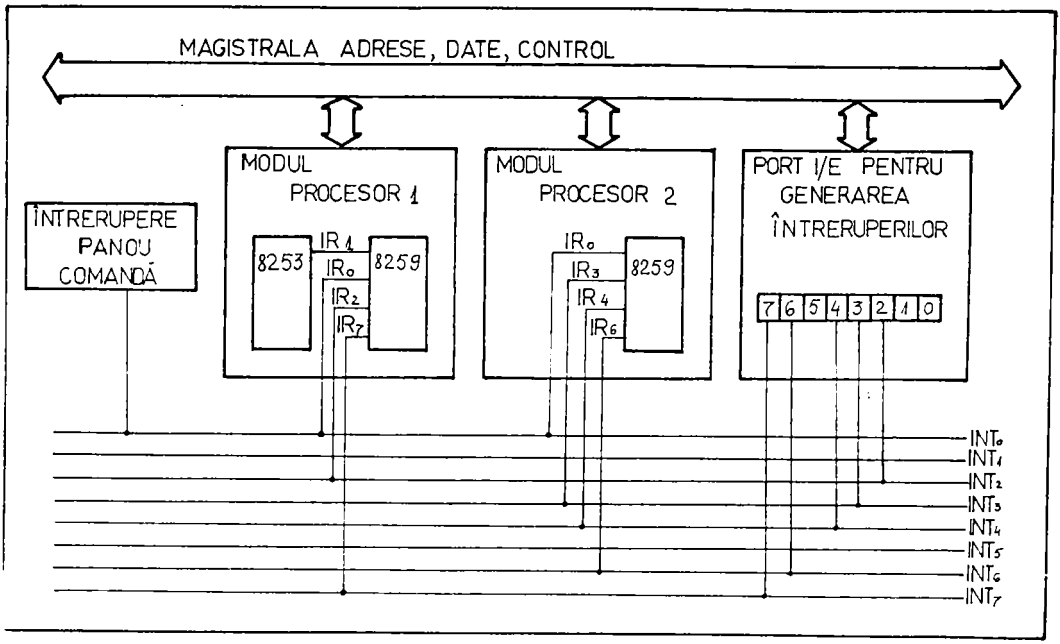


Fig.2.7. Sincronizarea comunicaţiilor prin intermediul întreruperilor.

Astfel prin acest procedeu,fiecare MP poate genera o întrerupere pentru unul sau mai multe MP-oare din sistemul muP,inclusiv pentru el însuşi. Procedura este aplicabilă tuturor categoriilor de sisteme muP.

2.3.2. Indicatori de stare.

Pentru protecţia resurselor comune dintr-un sistem muP se recurge la ataşarea a unu sau mai mulţi indicatori fiecărei resurse în parte, cu rol de semnalizare a stării resursei respective,liberă sau ocupată. Procedura de testare a stării acestor indicatori este cunoscută sub numele de " flag test and set " sau prescurtat de TAS şi constă din următoarele: MP care solicită accesul la o resursă testează starea indicatorului ataşat acesteia,după care dacă:

- indicatorul conţine "1" indicînd că resursa este utilizată, se trece într-o fază de aşteptare pînă la eliberarea sa;

- indicatorul conţine "0" indicînd că resursa este disponibilă,se trece la poziţionarea sa pe "1",indicîndu-se preluarea ei. Valoarea indicatorului rămîne nemodificată pe intervalul în care se utilizează resursa în cauză după care se trece pe "0",specificîndu-se eliberarea sa.

Deoarece utilizarea unei resurse este rezultatul unei concurenţe care are loc între mai multe cereri care sînt prezente simultan se impune ca în cadrul procedurii TAS să se procedeze la selectarea celei mai prioritare cereri care va testa şi eventual modifica indicatorul de condiţie aferent resursei solicitate. Pe de altă parte,în intervalul în care indicatorul este testat şi eventual modificat pot apare asincron una sau mai multe cereri care pot fi mai prioritare decît cea care operează asupra indicatorului în cauză. Pentru a nu se produce interferenţe,este necesar ca în intervalul în care se operează asupra unui indicator,acesta să nu mai poată fi accesibil altor cereri. Această cerinţă impune ca procedura TAS să fie indivizibilă,aspect realizat prin blocarea accesului la magistrală a altor cereri în intervalul de execuţie a unei proceduri TAS [**86].

Procedura TAS poate fi implementată [BB8o,We77] sub două forme:

In primul caz,indicatorii sînt variabile definite în memoria comună a sistemului mμP. Procedura se poate aplica atît pentru unităţile de I/E utilizate în comun,cît şi pentru protejarea unor zone din memoria comună. Implementarea soft prezintă dezavantajul că măreşte timpul de acces la o resursă,în contextul în care se introduc cereri suplimentare de acces la magistrala sistemului mμP.

In al doilea caz,procedura TAS este implementată hard,devenind o unitate de I/E adresabilă de către orice MP a sistemului mμP,în cadrul unor operaţii de I/E. Unui indicator i se asociază un bistabil a cărui conţinut este modificat prin operaţii de citire sau scriere, lansate de MP care solicită resursa căreia îi este ataşat indicatorul în cauză. Indivizibilitatea,se asigură prin faptul că se utilizează un singur impuls de tact al magistralei. Astfel pentru citire,operaţia de citire propriu-zisă şi testul se fac pe frontul crescător al impulsului de tact,iar modificarea indicatorului,dacă anterior a fost 0,se realizează prin setarea bistabilului utilizînd frontul descrescător al aceluiaşi impuls de tact. Pentru eliberarea resursei se lansează o operaţie de scriere care are ca urmare punerea pe zero a indicatorului,operaţie care se realizează prin resetarea bistabilului aferent indicatorului,în cauză,utilizînd frontul descrescător a unui

impuls de tact a magistralei. In prezent sistemele μP se utilizează în majoritatea cazurilor cu proceduri TAS realizate hard datorită avantajelor pe care le prezintă (timp redus de testare a indicatorului,încărcare mică a magistralelor de date şi adrese,fiabilitate,etc.).

In ambele implementări soft sau hard,selectarea cererii celei mai prioritare este realizată în cadrul procedurii TAS de arbitrul de magistrală,procedura asigurînd prin modul de implementare numai indivizibilitatea operaţiilor efectuate asupra indicatorilor de stare.

2.4. Tehnici de arbitrare a cererilor de acces la magistralele unui sistem multimicroprocesor.

Un arbitru are rolul de a accepta cererile de la MP-oare de tip master active,de a rezolva conflictele şi de a încunoştiinţa unităţile sistemului μP de cîştigarea accesului la magistrală. Arbitrii pot apare sub două forme:

- Arbitrii centralizaţi. In această situaţie arbitrul constă dintr-o singură unitate hard. De exemplu INTEL a dezvoltat un circuit specializat pentru modulul procesor SBC 8o/2o [**76,**86b] .
- Arbitrii descentralizaţi. In acest caz,controlul logic al rezolvării conflictelor de acces la magistrale este distribuit în MP-oare prezente în sistemul μP. Aceasta complică structura fiecărui MP,dar pe ansamblu,creşte fiabilitatea sistemului în cazul în care apare un defect în schema de arbitrare.

Ambele forme utilizează aceleaşi metode de arbitrare,în prezent fiind întîlnite un număr de trei metode:

- serială;
- paralelă;
- cu autoselecţie.

Toate aceste metode [Go8o,**83a,MP78,CI81] se vor prezenta pe rînd în cadrul unui model general de arbitrare a cererilor de acces la magistralele unui sistem μP. Pentru fiecare metodă în parte se va construi un model reprezentat printr-un graf de stare în care cercurile vor defini stări,dreptunghiurile acţiuni,iar arcele tranziţii. Pe arce se va trece condiţia (dacă este cazul) care determină trecerea dintr-o stare în alta.

2.4.1. Model pentru studiul arbitrării seriale.

Caracteristici

Tehnica de arbitrare serială,cere ca fiecare MP conectat la magistrală (fig.2.8.) să dispună de o prioritate fixă. Fiecare MP utilizea-

ză un număr de trei semnale. Primul este un semnal de intrare BUS GRANT IN (BGI) prin intermediul căruia MP este informat dacă magistrala a fost cerută de un MP cu o prioritate mai mare. Cel de al doilea,este un semnal de ieşire BUS GRANT OUT (BGO) care este conectat la intrarea BGI a MP cu prioritatea mai mică,imediat următor. Pentru MP cu cea mai mare prioritate semnalul BGI este întotdeauna activ. Cel de al treilea,este un semnal care apare pe o linie comună care leagă toate MP într-o conexiune SAU numit BUS ACCEPT BUSY RELEASE (BABR). Semnalul indică starea curentă a magistralei,liberă sau ocupată,avînd rolul de a preveni intrarea pe magistrală a unui MP atunci cînd aceasta este ocupată de altul. Prioritatea de acces la magistrală,poate fi schimbată,prin modificarea poziţiei de interconectare a MP,două modalităţi fiind prezentate în fig.2.8.b şi c,unde în primul caz se recurge la modificarea conexiunilor semnalelor BGI şi BGO,iar în al doilea caz la utilizarea unei scheme matriciale de interconectare.

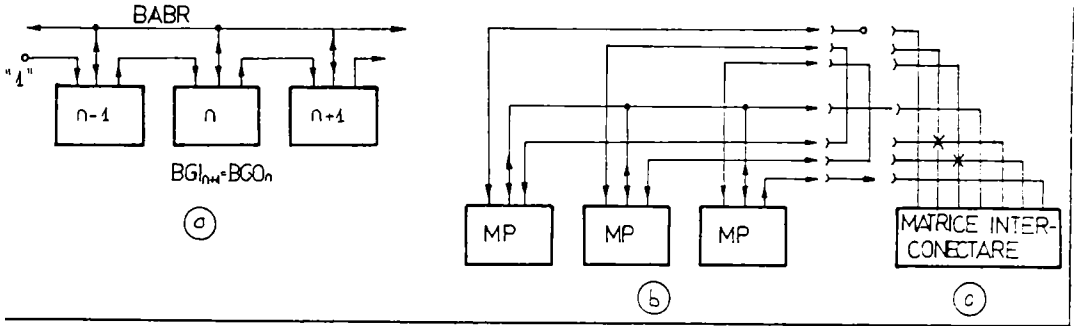


Fig.2.8. a) Tehnica de arbitrare serială.
b,c) Schimbarea priorităţilor în arbitrarea serială prin schimbarea poziţiei sau prin utilizarea unei scheme matriciale de interconectare.

Graful de stare

După cum se vede din fig.2.9.,arbitrul de magistrală care este prezent în fiecare MP potenţial utilizator al sistemului se poate găsi în una din următoarele stări:

- MP PASIV. Starea nu se părăseşte dacă nu apare o cerere REQUEST (RQ_n) internă de utilizare a magistralei,emisă de MP cu numărul n arbitrului propriu. De asemenea arbitrul de magistrală aflat în această stare poate lua parte la procedura de acordare a acceptului de utilizare a magistralei făcînd egal semnalul propriu de ieşire BGO_n cu semnalul de intrare primit BGI_n.

- MP ACTIV. Trecerea în această stare se realizează în două etape. Prima etapă începe o dată cu apariţia unui semnal asincron de utilizare a magistralei RQ_n, urmată de dezactivarea lui BGO_n asincron cu BUSCLK, (impulsuri de sincronizare a transferurilor pe magistrală). A doua etapă, constă într-o fază de aşteptare a propagării lui BGO_n prin toate MP care urmează. După ce a aşteptat ca semnalul BGO_n să fie recunoscut de toţi arbitrii MP cu prioritate mai mică (aceasta cu scopul evitării apariţiei unor cereri multiple de acces la magistrală, manifestate prin activarea lui BABR), arbitrul MP care a emis cererea aşteptată în starea MP ACTIV pînă cînd BGI_n propriu este activat şi BABR dezactivat, caz în care trece în starea MP UTILIZATOR. Dacă BGI_n a fost dezactivat la un moment dat, arbitrul MP în cauză va rămîne în această stare şi după un interval de timp va trece în starea MP PASIV, cazul apărînd în momentul emiterii unei cereri cu prioritate mai mare.

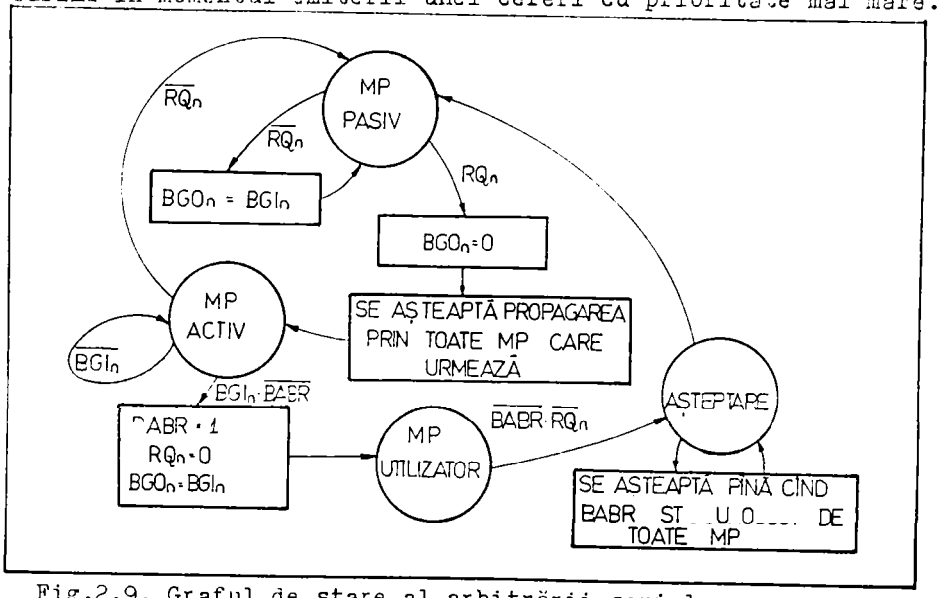


Fig.2.9. Graful de stare al arbitrării seriale.

- MP UTILIZATOR. Starea corespunde situaţiei în care arbitrul MP care a efectuat cererea a cîştigat magistrala, fapt semnalat celorlalţi potenţiali utilizatori prin activarea lui BABR. De asemenea se trece la egalarea semnalului de ieşire propriu BGO_n cu semnalul de intrare BGI_n, ceea ce are ca efect scoaterea arbitrului din competiţia imediată de cîştigare din nou a magistralei, după care se resetează RQ_n. Procedura de transfer începe din momentul în careBABR este activat şi după un număr variabil de perioade

BUSCLK transferul este terminat şi un nou MP poate prelua controlul magistralei. Transferul datelor şi selectarea următorului MP din cele care cer accesul la magistrală are loc paralel. După ce transferul a avut loc arbitrul părăseşte starea MP UTILIZATOR prin dezactivarea lui BABR şi RQ_n.

- ASTEPTARE. Este o stare intermediară care se introduce cu un scop dublu. Pe de o parte,se urmăreşte ca magistrala să nu fie recîştigată din nou de arbitrul care a cîştigat anterior magistrala,şi de a permite astfel ca şi alţi arbitrii să treacă în starea MP UTILIZATOR. Pe de altă parte,se evită accesul repetitiv la magistrală a arbitrului MP cu cea mai mare prioritate (este cazul în care magistrala este ocupată alternativ de doi arbitrii prezenţi în MP-oare cu cea mai mare prioritate).

2.4.2. Model pentru studiul arbitrării paralele.

Caracteristici

Tehnica de arbitrare paralelă se utilizează atunci cînd se impune o metodă de selecţie mai flexibilă sau cînd întîrzierile datorate propagării semnalelor datorită modului de legare a MP în arbitrarea serială devin prea mari. Metoda cere ca fiecare arbitru de magistrală a fiecărui MP să dispună de linii proprii prin care acestea sînt legate la o anumită unitate de arbitrare paralelă (fig.2.1o.).

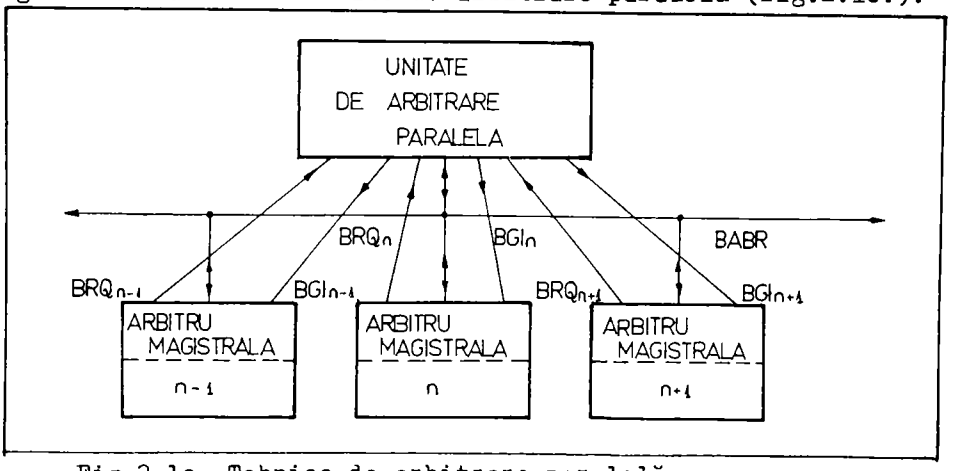


Fig.2.1o. Tehnica de arbitrare paralelă.

Similar ca şi în cazul arbitrării seriale,fiecare arbitru de magistrală utilizează un număr de trei semnale în care BGI_n şi BABR au aceiaşi semnificaţie,iar cel de al treilea BRQ_n care reprezintă o cerere de acces la magistrală (BUS REQUEST)este identic cu BGI_n.

Unitatea de arbitrare paralelă are rolul de a primi cererile de acces la magistrală şi de a da acceptul de utilizare a aceleia care di spune de prioritatea cea mai mare. Este de menţionat faptul că prioritătile de acces şi în acest caz sînt fixe, unitatea de arbitrare paralelă avînd însă posibilitatea analizei lor în paralel şi de a supraveghea modul de funcţionare a fiecărui arbitru de magistrală.

Diagrama de stare

După cum se vede din fig.2.11., iniţial fiecare arbitru de magistrală care este prezent în fiecare MP potenţial utilizator al sistemului se găseşte în starea MP PASIV. In momentul apariţiei unei cereri interne de utilizare a magistralei RQ_n în MP cu numărul n, arbitrul său de magistrală trece la activarea lui BRQ_n după care trece în starea MP ACTIV. In această stare rămîn toate MP-oare pînă în momentul în care unitatea de arbitrare dă cîştig uneia prin activarea semnalului său de intrare BGI_n. In momentul în care BABR este anulat deci magistrala devine disponibilă, MP trece în starea MP UTILIZATOR.

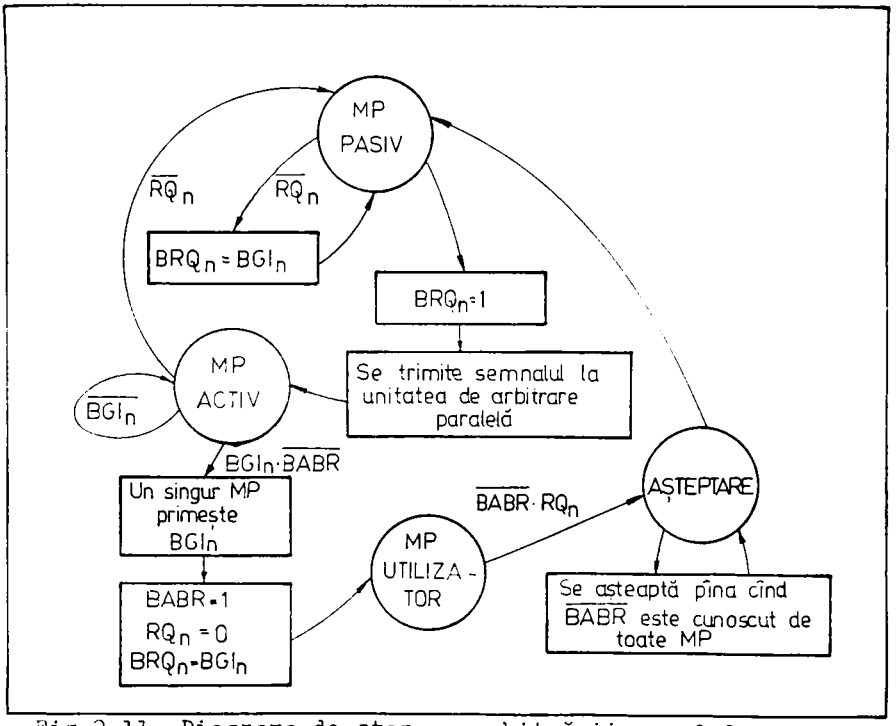


Fig.2.11. Diagrama de stare a arbitrării paralele.

In continuare restul diagramei este identică ce cea de arbitrare serială.

Arbitrare paralelă cu rotaţia priorităţilor

O variantă a arbitrării paralele care uneori este considerată ca reprezentînd o metodă separată este cea în care se asigură rotaţia priorităţilor MP-oare,în scopul acordării unor şanse egale de acces la magistrală fiecărui MP în parte. Există un număr mare de variante care merg de la o rotaţie simplă a priorităţilor pînă la algoritmi complecşi de rotaţie. Pentru a ilustra acestă variantă a arbitrării paralele se va analiza o tehnică de rotaţie propusă în [CI81] ,în contextul modelului de arbitrare propus fără a mai prezenta diagrama de stare şi timp care sînt identice cu cele de la arbitrarea paralelă.

Tehnica de rotaţie a priorităţilor cere prezenţa în unitatea de arbitrare paralelă a unui cuvînt de stare PRIORITY STATUS WORD,prescurtat PRSW care conţine priorităţile asociate fiecărui MP. De exemplu,pentru un număr de patru MP cuvîntul de stare să presupunem că este PRSW=(4,1,2,3) în care poziţia codifică numărul MP iar valoarea prioritară de acces - 1 reprezentînd cea mai înaltă prioritate,iar 4 cea mai mică (pentru exemplul dat MP cu numărul unu are cea mai mică prioritate,iar MP cu numărul doi cea mai mare).

Mecanismul de arbitrare a cererilor de acces la magistrală poate fi descris astfel: presupunem că la un moment dat PRSW=(4,1,2,3) şi că o cerere de acces este efectuată de MP cu numărul 3 şi 4. Deoarece MP cu numărul 3 are o prioritate mai mare (2) decît MP cu numărul 4 (prioritate 3),arbitrul de magistrală va permite accesul lui MP3. Dacă în intervalul în care magistrala este acordată lui MP 3 apare o cerere de acces din partea lui MP 1 (prioritate 4),aceasta va fi refuzată. Dacă însă cererea este formulată de MP 2 (prioritate 1),magistrala va fi luată lui MP 3 atîta timp cît MP 2 are nevoie de ea. Cînd accesul în acest ultim caz MP 2 eliberează magistrala,PRSW va fi schimbat din (4,1,2,3) în (3,4,1,2),observîndu-se că MP care a eliberat magistrala primeşte cea mai mică prioritate,restul MP-oare primind o prioritate care este mai mică cu o unitate decît în faza anterioară. Se observă că poziţia relativă din punct de vedere a priorităţilor nu se schimbă pentru MP 3,4 şi 1. Presupunem acum,că MP 2 nu formulează nici o cerere de acces în intervalul în care magistrala este acordată lui MP 3. Cînd MP 3 eliberează magistrala,PRSW se va schimba din (4,1,2,3) în (3,1,4,2). Se poate observa că MP 2 nu şi-a pierdut poziţia sa relativă în raport cu toate celelalte MP-oare cu excepţia celui care a terminat accesul pe magistrală şi care primeşte cea mai mică prioritate de acces din sistem. Procedura se desfăşoară similar în continuare,asigurînd în final o repartiţie echitabilă a magistralei

tuturor MP-oare.

2.4.3. Model pentru studiul arbitrării cu autoselecţie.

Caracteristici

Tehnica de arbitrare prezentată în fig.2.12.,constă dintr-o operaţie de comparare a codurilor de prioritate efectuată de însăşi MP-oare care au cerut accesul la magistrală. Pentru aceasta se utilizează un număr suplimentar de linii de stare (acestea pot fi considerate că alcătuiesc o magistrală aparte),linii care poartă numele de CODED SYSTEM STATUS$_k$,prescurtat CSS$_k$,cu rolul general de a defini MP care a cîştigat magistrala. Fiecare arbitru de magistrală prezent în fiecare MP îi este atribuit un cod de prioritate,utilizat de către acesta în competiţia de cîştigare a magistralei. Fiecare linie în parte CSS$_k$ (numărul de linii,dat de k depinde de codul de prioritate,de exemplu, pentru k=5 sînt posibile definirea unui număr de 2^5-1 deci 31 nivele de prioritate) este legată într-o conexiune de tip SAU la aceiaşi poziţie binară a codului de prioritate din fiecare arbitru de magistrală. Procedura de autoselecţie începe prin depunerea pe liniile CSS$_k$ a codului de prioritate de către arbitrii MP-oare care cer accesul pe magistrală. Într-o a doua etapă se trece la compararea codului propriu cu cel existent pe liniile CSS$_k$,începînd cu bitul cel mai semnificativ al codului şi terminînd cu cel mai puţin semnificativ,simultan pentru aceiaşi poziţie binară de către toţi arbitrii de magistrală implicaţi în competiţie. Dacă valoarea binară prezentă în poziţia k a codului este egală cu cea existentă pe linia CSS$_k$,atunci MP rămîne în competiţie,dacă nu se retrage.

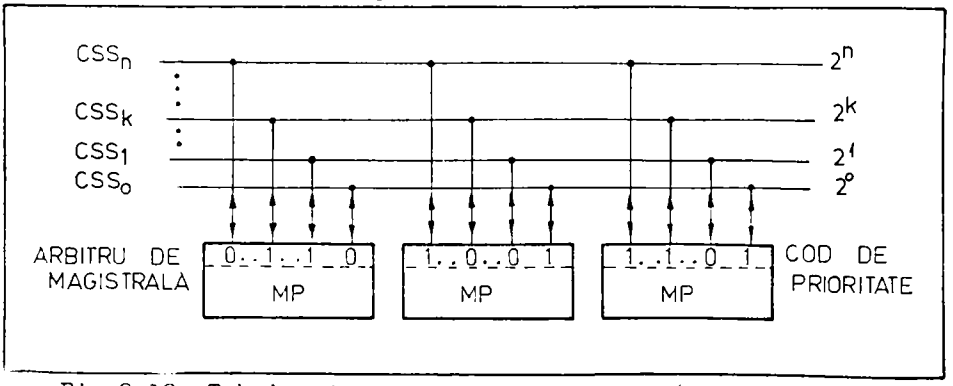


Fig.2.12. Tehnica de arbitrare cu autoselecţie.

Rezultatul final îl constituie selectarea MP cu cea mai mare prioritate de acces. Operaţia de comparare,care după cum se observă se de-

făşoară secvenţial,se poate realiza simplu de către un comparator comandat de impulsurile BUSCLK prezent în fiecare arbitru de magistrală. Spre deosebire de tehnicile de arbitrare prezentate anterior,în cazul acesteia nu se mai utilizează semnalele BGI,BGO şi BRQ care nu îşi mai au rostul,se utilizează însă în schimb semnalul BUS USER SELECT prescurtat BUSEL care are menirea de a declanşa procedura de autoselecţie. Semnalul poate fi emis de oricare din arbitrii MP care cer accesul la magistrală.

Diagrama de stare

Este prezentată în fig.2.13.,şi după cum se observă arbitrul de magistrală care este prezent în fiecare MP potenţial utilizator al magistralei se poate găsi în una din următoarele stări:

- MP PASIV. Similar,ca şi în cazul celorlalte două tehnici de arbitrare prezentate anterior,corespunde situaţiei în care MP nu solicită un acces pe magistrală. Din această stare se poate ieşi în momentul apariţiei cererii interne de utilizare,semnalată prin activarea semnalului RQ_n;
- MP ACTIV. Arbitrul de magistrală a MP cu numărul n trece în

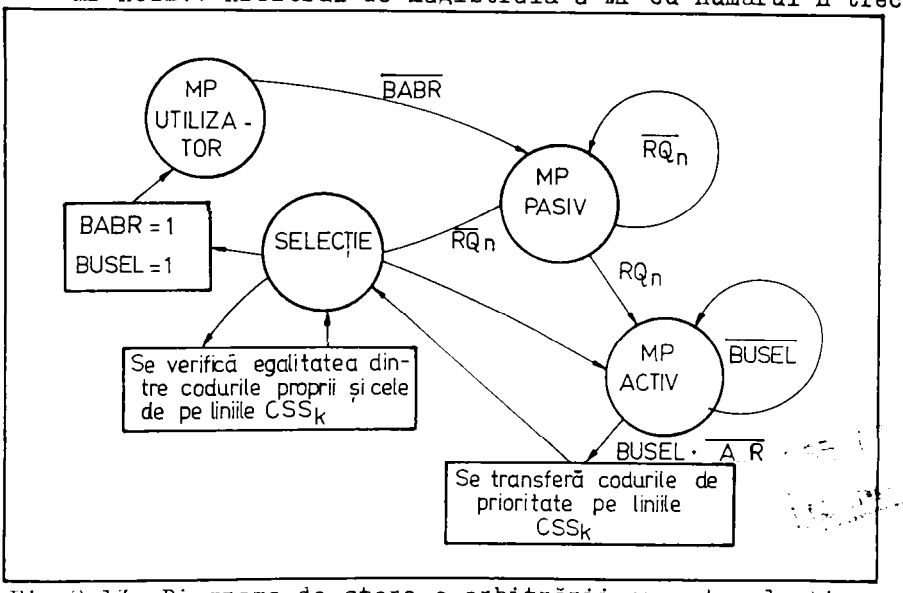


Fig.2.13. Diagrama de stare a arbitrării cu autoselecţie.

această stare în momentul activării lui RQ_n,şi rămîne atîta timp cît nu este primit acceptul de declanşare a procedurii de autoselecţie. Startul este dat de MP care se găseşte în starea MP UTILIZATOR prin activarea lui BUSEL. În acest moment se depun codurile de prioritate pe liniile CSS_k după care se trece în etapa

propriu zisă de selectare a celui mai prioritar arbitru de magistrală şi deci a celui mai prioritar MP;

- SELECTIE. In această stare se efectuează compararea la nivelul fiecărui arbitru de magistrală care a solicitat accesul codurilor proprii cu cele existente pe liniile CSS_k, simultan pentru toate MP care au cerut accesul. Compararea se face poziţie binară cu poziţie binară, apariţia unei neconcordanţe însemnînd eliminarea din competiţie a MP în cauză. Ultimul arbitru rămas activează BABR, indicînd prin aceasta ocuparea magistralei şi trecerea în următoarea stare ca şi declanşarea următoarei proceduri de autoselecţie;

- MP UTILIZATOR. Corespunde etapei în care se efectuează transferul de către MP cîştigător. In paralel cu transferul datelor are loc şi procedura de autoselecţie pentru celelalte MP care nu au primit acceptul de utilizare a magistralei.

2.4.4. Concluzii privind tehnicile de arbitrare.

Rezolvarea conflictelor generate de cererile multiple de acces la magistralele unui sistem μP, reprezintă una din problemele critice, de concepţia folosită depinzînd în mare măsură performanţele obţinute. Alegerea unei tehnici sau a alteia, este strîns legată de aplicaţia căreia îi este destinat sistemul, criteriul, tehnică de arbitrare cît mai simplă, în condiţiile obţinerii unor performanţe (capacitate de trecere, timp de răspuns) ridicate fiind întotdeauna urmărit în momentul alegerii unei sau a alteia din aceste tehnici. O caracterizare succintă, în cîteva cuvinte, care să dea o imagine globală a noţiunii de arbitrare şi a tehnicilor utilizate se va prezenta în cele ce urmează.

Arbitrarea serială este din punct de vedere conceptual simplă, dar prezintă o serie de limitări şi neajunsuri care îngrădesc în multe cazuri utilizarea. Astfel în multe situaţii, utilizarea de priorităţi fixe reprezintă un impediment. In cazul în care este acceptată această situaţie, nu este posibilă conectarea a mai mult de 3 MP master datorită întîrzierilor de propagare de-a lungul liniilor BGI-BGO. Arbitrarea paralelă este utilizată în cazul unor sisteme mai complexe în care sînt prezente un număr mai mare de 3 MP master (în cazul unei magistrale de tip Intel Multibus, numărul maxim de MP este de 16), cărora li se atribuie priorităţi fixe. Tehnica impune utilizarea unei unităţi de arbitrare paralelă, compusă dintr-un decodificator şi codificator de prioritate. Deci considerabil mai rapidă decît metoda arbitrării seriale, rămî-

ne cu dezavantajul unor nivele fixe de prioritate,care în anumite aplicaţii reprezintă o limitare serioasă a performanţelor sistemului mμP. O rezolvare parţială a acestui impediment s-a realizat prin introducerea conceptului de rotaţie a priorităţilor. Metoda deşi asigură o flexibilitate mai mare,nu rezolvă în totalitate problema,deoarece,în timp nu face altceva decît să"uniformizeze" accesul la magistrală a MP-oare,fără a exista posibilitatea modificării prin soft a priorităţilor de acces,aceasta în condiţiile în care este necesară prezenţa unui hard adiacent care măreşte complexitatea arbitrului de magistrală.

Arbitrarea cu autoselecţie,elimină dezavantajele prezentate anterior,permiţînd modificarea nivelelor de prioritate de către taskurile care sînt executate la un moment dat de un MP. Tehnica are dezavantajul complicării soft a programelor care se execută împlicînd o analiză atentă a evoluţiei nivelelor de prioritate în timp (o secvenţă necorespunzătoare poate înrăutăţi catastrofal performanţele, putînd să conducă la eliminarea definitivă din competiţie a unor MP). Un alt dezavantaj îl constituie faptul că deşi din punct de vedere hard soluţia de implementare nu pune probleme deosebite,pînă în prezent,metoda nu este standardizată,înţelegînd prin aceasta că o serie de magistrale standard destinate unor aplicaţii mμP nu o prevăd,neexistînd circuite specializate destinate implementării ei (spre deosebire de arbitrarea paralelă şi serială unde astfel de circuite sînt prezente). O explicaţie ar constitui-o complicaţiile de realizare a implementării soft a priorităţilor şi a sincronizării acestora în timp.

In concluzie,alegerea uneia sau a alteia din tehnici,depinde puternic de aplicaţia căreia îi este destinat sistemul mμP,urmărindu-se de fiecare dată atingerea unui optim,care rezultă din punerea în comun a mai multor deziderate: realizare hard şi soft,preţ de cost scăzut,fiabilitate ridicată.

2.5. Protocoale de comunicaţie pe magistrală.

In tehnica de calcul cuvîntul protocol desemnează un mecanism prin care se stabileşte modul în care urmează să se efectueze un transfer de informaţie. Dacă la nivelul sistemului de operare protocolul desemnează mecanismul prin care se efectuează schimbul de informaţie între taskuri,la nivelul magistralei,aceasta va desemna modul în care:

- se efectuează cererea de acces la magistrală;
- se acceptă cererea de acces;
- se efectuează transferul.

Din acest punct de vedere,protocolul de comunicaţie pe magistrală reprezintă un ansamblu de semnale (desemnate prin nivele de tensiune sau curent),generate de interfeţele de magistrală a unităţilor sistemului. Aceasta face ca din punct de vedere ierarhic,aceste protocoale să reprezinte cel mai scăzut nivel întîlnit într-un sistem. O schemă de clasificare [Th72] este prezentată în fig.2.14. Se observă că sînt prezente două tipuri de protocoale sincrone şi asincrone. In cazul unui protocol sincron,modul în care se efectuează transferul de informaţie între unitatea emiţătoare şi cea receptoare este prestabilit (intervale de timp,succesiune de impulsuri),aspect care implică obligativitatea ca ambele unităţi să utilizeze impulsuri de sincronizare furnizate de un ceas unic. Pentru protocolul în schimb asincron nici o informaţie privind intervalul de timp nu este disponibil aprioric. Indicii de performanţă care se utilizează pentru a aprecia calitatea protocolului utilizat,se referă la viteză şi posibilitatea de transmisie eronată.

SINCRON
ASINCRON
  COMANDA UNIDIRECTIONALA
    SURSA CONTROLATA
    DESTINATIA CONTROLATA
  CERERE / CONFIRMARE
    NECUPLAT
    SEMICUPLAT
    CUPLAT

Fig.2.14. Mecanisme de comunicaţie.

Protocoale de comunicaţie asincronă

Se împart în două categorii:
- comandă unidirecţională;
- cerere / confirmare.

Protocol de tip comandă unidirecţională.

In cadrul acestui tip de mecanism nu există o confirmare a recepţionării informaţiei. Este de două tipuri corespunzătoare unităţilor care sînt controlate - sursa sau destinaţie (fig.2.15),în care sem-

nificaţia notaţiilor pentru semnalele prezentate pentru valoarea logică adevărată este:

DATA - dată disponibilă la unitatea sursă;

RDY - magistrală liberă;

REQUEST - cerere de recepţie la unitatea destinaţie,cu condiţia că magistrala este liberă;

ACCEPT - confirmare de recepţie emisă de unitatea destinaţie.

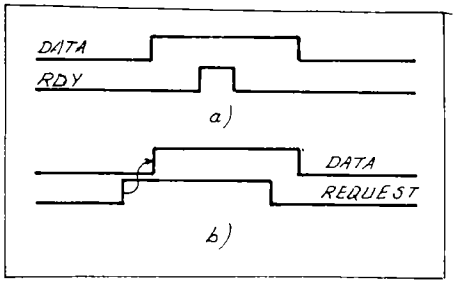


Fig.2.15. Mecanismul de comunicaţi unidirecţional.
a) sursa controlată
b) destinaţie controlată

Problema care se pune este aceea de a asigura recepţia datei în intervalul în care aceasta este prezentată. Aceasta implică o ajustare iniţială a intervalelor care apar, motiv pentru care deşi mecanismul este simplu şi asigură o viteză mare de transmidie nu se poate aplica în cazul în care magistrala este împărţită de unităţi cu viteze de transmisie diferite.

Protocol de tip cerere / confirmare.

Din mecanismul de comunicaţie prezentat în fig.2.16.,se observă că în toate cele trei situaţii care pot apare,sincronizarea dintre unitatea emiţătoare şi cea receptoare este asigurată de semnalul ACCEPT. Cele trei variante de comunicaţie: necuplat,semicuplat şi cuplat,au apărut datorită modului diferit de tratare a semnalului de magistrală disponibilă. Pentru acest mecanism de transmisie erorile datorate recepţiei unei date care nu este prezentă pe magistrală sau

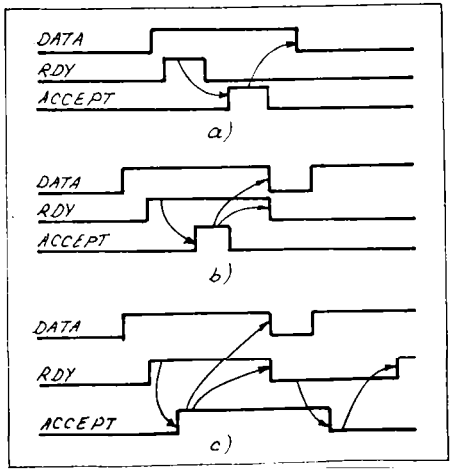


Fig. 2.16. Mecanismul de comunicaţie cerere / confirmare.
a) Necuplat
b) Semicuplat
c) Cuplat

de recepţionare de două ori a aceleiaşi date este exclusă rămînînd numai prezente erorile datorate magistralei.

Protocoale de comunicaţie sincronă

Comunicaţia sincronă implică corelarea unităţilor sursă şi destinaţie pe un timp global standard. In această situaţie problemele care trebuiesc rezolvate se referă la:

- mecanismul de sincronizare;
- transmisia şi verificarea datelor.

Mecanismul de sincronizare

Impulsurile de ceas pot fi generate fie central fie local. In cazul sistemului centralizat,impulsurile aceluiaşi ceas sînt utilizate de toate unităţile conectate la magistrală. Sistemul este succeptibil la erori datorită fenomenului de "skew" care poate apare. A doua variantă presupune emiterea la anumite intervale de timp a unor semnale de sincronizare a ceasului care sînt utilizate de fiecare unitate în parte,pentru sincronizarea ceasurilor proprii (de obicei numărătoare). Aceste semnale pot fi transmise pe o linie separată sau intercalat între datele schimbate de unităţi. Si în aceste două cazuri pot apare probleme,în prima situaţie datorită fenomenului de "skew" care apare între impulsul de sincronizare şi transmisia datelor pe magistrală, iar în al doilea caz prin complicarea sistemului de selectare a impulsului de sincronizare dintre datele propriu zise.

Transmisia şi verificarea datelor

Deoarece mecanismul de transmisie sincronă necesită o corelare perfectă a unităţii emiţătoare cu cea receptoare,se impune alocarea în acelaşi moment a unor cantităţi egale de timp (cuante de timp) ambelor unităţi,intervalul corespunzînd de altfel momentului în care magistrala este liberă şi poate fi utilizată. Există două posibilităţi de alocare:

- dedicată;
- nededicată.

In procedura de alocare dedicată intervalul de timp în care poate fi folosită magistrala,acesta este atribuit pe rînd fiecărei unităţi în parte,indiferent de gradul de utilizare a magistralei de către acestea. Procedura se aplică dacă unităţile sistemului sînt de acelaşi tip,intervalul fiind calibrat funcţie de cea mai lentă unitate. In procedura de alocare nededicată aceste intervale sînt alocate funcţie de

cererile emise de unităţi. Faptul că este necesară prezenţa unui algoritm de alocare pe de o parte complică modul de implementare a procedurii,dar pe de altă parte asigură o flexibilitate mai mare sistemului.

2.6. Tehnici de comunicaţie între modulele procesor.

Prin însăşi definiţia sa,un sistem mμP implică prezenţa unui dialog între MP-oare care îl compun. Iniţierea acestui dialog,impune pe de o parte ca partenerul de dialog să fie atenţionat de necesitatea unui schimb de informaţie,iar pe de altă parte,de definirea unei modalităţi de desfăşurare a dialogului între cei doi parteneri implicaţi în schimbul de informaţie.

In prezent atenţionarea partenerului de dialog se efectuează prin intermediul sistemului de întreruperi. Modul în care acestea sînt utilizate s-a prezentat în paragraful 2.3.

Desfăşurarea propriu zisă a dialogului între două MP se efectuează în două moduri:

- direct;
- printr-un intermediar.

In modul direct de dialog un MP adresează direct memoria proprie a partenerului. Această modalitate este în general mai puţin utilizată deoarece implică o serie de complicaţii. Pe de o parte este necesar ca spaţiul de adrese comun să fie distribuit în toate memoriile proprii ale MP-oare. Deci memoria unui MP va conţine un spaţiu propriu cît şi un spaţiu care poate fi adresat de restul MP-oare,ceea ce atrage după sine complicaţii în realizarea taskurilor utilizator. Pe de altă parte,fiecare memorie a unui MP va trebui să dispună suplimentar de un port de acces prin care să fie recepţionată informaţia de pe magistrala sistemului ceea ce impune prezenţa unei posibilităţi de adresare dublă a memoriei ceea ce evident complică considerabil structura unui MP. Conceptul cunoscut sub denumirea de "dual-port" este utilizată în cazurile în care timpul necesar pentru a transfera o informaţie de la un MP la alt MP reprezintă o resursă critică a sistemului,situaţie în care rapiditatea transferului este contrabalansată de complicaţia hardului unui MP.

In dialogul efectuat printr-un intermediar,informaţia este preluată iniţial de o unitate intermediară care poate fi utilizată în comun de toate MP-oare prezente în sistem,după care într-o a doua e-

tapă la unitatea în cauză are acces partenerul de dialog care preia informaţia respectivă. Ca element intermediar poate fi utilizată o unitate de I/E comună,în general un disc,soluţie mai puţin utilizată datorită timpului mare de transfer cît şi a costului ridicat pe care îl presupune în cazul în care aceasta este utilizată numai pentru dialogul dintre două MP. Soluţia care s-a impus şi care este utilizată în prezent în majoritatea sistemelor mμP este de a rezerva rolul de intermediar,unei unităţi de memorie comune [We74,We77,CC81] . Modul de utilizare a acesteia în cîteva cuvinte este următorul. In memoria comună pentru fiecare MP este prevăzută o căsuţă poştală "mailbox" ca şi un indicator de stare a acesteia. Indicatorul va specifica dacă în căsuţa poştală este prezent sau nu un mesaj. In cazul în care este prezent,se specifică lungimea şi adresa mesajului ca şi sensul de transfer din memoria comună spre memoria proprie,sau din memoria proprie spre memoria comună. Un MP care doreşte iniţierea unui dialog cu alt MP intră în competiţia de cîştigare a magistralei, după ce a cîştigat-o depune un mesaj în căsuţa poştală corespunzătoare partenerului de dialog,poziţionează indicatorul acesteia pe ocupat şi procedează la emiterea unei întreruperi destinate atenţionării acestuia din urmă. MP partener de dialog în urma primirii întreruperii va intra la rîndul său în competiţia de cîştigare a magistralei. In momentul cîştigării acesteia,va trece la verificarea conţinutului căsuţei şi la efectuarea transferului,funcţie de sensul solicitat,după care procedează la poziţionarea pe liber a indicatorului de stare a căsuţei poştale şi va elibera magistrala. Procedura de transfer este simplă şi nu necesită un hard adiţional. Singurele complicaţii intervin în sistemul de operare mμP şi se referă la faptul că în cazul în care căsuţa poştală a MP care este solicitat să devină partener de dialog,este ocupată,este necesară punerea într-un şir de aşteptare la căsuţa poştală respectivă a cererii în cauză. Sincronizarea şi planificarea cererilor de dialog reprezintă una din funcţiile cele mai importante a unui sistem de operare mμP şi ea va fi discutată într-un capitol ulterior.

2.7. Particularităţi de încorporare în structuri multimicroprocesor a microprocesoarelor pe 8,16 şi 32 de biţi.

In încheierea acestui capitol considerăm necesară o trecere în revistă a facilităţilor pe care le prezintă din punct de vedere a încorporării în structuri mμP familiile de μP-oare existente în prezent

Studiul se va face din punct de vedere a cerinţelor pe care le impune o structură mμP pe de o parte,iar pe de altă parte din punctul de vedere a modalităţilor în care μP-oarele de 8,16 şi 32 de biţi pot realiza principalele funcţii impuse de structura mμP. Acestea din urmă se vor referi la modalităţi de realizare a excluderii mutuale la accesul la o resursă comună ca şi la modul în care se asigură interconectarea între MP-oare.

2.7.1. Cerinţe impuse microprocesoarelor de o structură multimicroprocesor.

Microprocesoarele care intră în componenţa MP-oare din cadrul unui sistem mμP trebuie să aibă posibilitatea identificării ciclurilor de maşină şi a stărilor stroburilor pentru magistrală [Pa83, **87a] . Acest lucru este necesar pentru controlul magistralei cît şi pentru detectarea şi identificarea defecţiunilor. O intrare de comandă READY care permite μP-orului să intre în WAIT,este necesară în cazurile cînd dispozitivul adresat nu răspunde în cadrul unui singur ciclu maşină. Circuitul trebuie să fie prevăzut cu o intrare HOLD ce opreşte μP şi plasează adresele,datele şi magistralele de comandă în stare de impedanţă ridicată la terminarea instrucţiei ce se execută şi activează ieşirea HLDA. Cînd se îndepărtează semnalul HOLD,μP începe execuţia următoarei instrucţii în secvenţă. Pentru a îmbunătăţi accesul la magistralele sistemului mμP se impune ca μP să dispună atît de o ieşire de tip BUS REQUEST cît şi de o intrare de tip BUS GRANT. Aceste două semnale sînt utile doar pentru structurile mμP reduse. Sistemele mai mari impun pe lîngă necesitatea prezenţei acestor două semnale şi a unor memorii tampon cu rol de a optimiza accesul cererilor la magistralele sistemului cît şi a unor controlere de magistrală cu rolul de a coordona aceste cereri de acces. Utilizarea magistralelor sistemului va fi îmbunătăţită dacă se prevede o instrucţie de tip DELAY pentru buclele de timp interne în aşteptarea accesului la o resursă comună,fără ca să se fi obţinut accesul la magistrală. Pentru a îmbunătăţi comunicaţiile interprocesor este necesară lărgirea posibilităţilor de adresare prin utilizarea adresării indirecte şi indexate ca şi prin utilizarea unor registre interne de 16 şi 32 de biţi. De asemenea,comunicaţiile între MP-oare datorită diversităţii situaţiilor care apar,impun necesitatea ca μP-oarele să dispună de un număr mare de nivele de întrerupere ca şi de posibilitatea vectorizării acestora.

Este de menţionat că principiile enunţate mai sus au un caracter general,fiecare firmă producătoare de μP-oare respectîndu-le într-o manieră proprie funcţie de tipul μP-orului pe 8,16 sau 32 de biţi, fapt care reflectă în ultimă instanţă experienţa cîştigată în realizarea sistemelor mμP.

2.7.2. Realizarea excluderii mutuale.

Deoarece problemele legate de stabilirea unor structuri mμP,au apărut după realizarea primelor μP-oare,este normal ca acestea să nu dispună de facilităţi deosebite de încorporare în structuri mμP. Astfel existenţa ciclurilor WAIT pentru sincronizarea transferului de date între μP şi dispozitivele periferice mai lente nu este suficientă pentru realizarea excluderii mutuale. Cele două operaţii esenţiale ale excluderii mutuale,blocarea magistralei şi realizarea operaţiei indivizibile de TAS,nu se pot efectua direct prin metode soft,neexistînd instrucţiuni distincte. In această situaţie se recurge la o soluţie hard de implementare,prin definirea unui fanion de tip TAS ca şi port de I/E. Efectuînd o instrucţiune de citire, conţinutul bistabilului fanion este transferat către μP şi bistabilul este pus pe "0". La o instrucţie de scriere,bistabilul este pus pe "1". Indivizibilitatea este asigurată prin modalitatea de realizare a operaţiei de citire / poziţionare obţinută pe durata unui singur impuls BCLK/.

Se realizează operaţiile:

IN TAS =
[ACC:=FLAG
IF FLAG=1 THEN FLAG:=0]

şi respectiv:

OUT TAS =
[FLAG=1]

cu specificarea că operaţiile cuprinse între paranteze pătrate indică indivizibilitatea. Operaţiile IF THEN se realizează într-un singur tact de către o schemă hard.

Pentru excluderea mutuală la μP-oarele pe 16 şi 32 biţi au apărut instrucţii specifice de asigurare a excluderii mutuale [Sc83, Pt85] . Astfel μP I8086 are prevăzut semnalul LOCK/ ce poate fi activat prin program pentru a împiedica un alt MP de a obţine magistrala. Acesta poate fi utilizat pentru a implementa un mecanism de excludere mutuală prin definirea unui semafor soft,cu rolul de a re-

glementa accesul cererilor simultane la o resursă comună a sistemului mμP. Semnalul LOCK/ este generat de μP I8086 sau μP I8088 de unitatea BIU (Bus Interface Unit) atunci cînd unitatea EU (Execution Unit) execută instrucţia cu prefix LOCK. Semnalul LOCK/ rămîne activ în timpul execuţiei instrucţiei ce urmează prefixului LOCK. Dacă un alt MP solicită utilizarea magistralei sistemului mμP,cererea în cauză este înregistrată dar nu este onorată pînă la terminarea instrucţiei blocate prin prefix. Un exemplu de secvenţă TAS,realizată soft are în acest caz următorul aspect:

```
Verif:  MOV   AL,1      ; AL=1 şi blocare
LOCK    XCHG  Sem,AL    ; TAS blocat
        TEST  AL,AL     ; Set fanioane
        JNZ   Verif     ; Buclă dacă LOCK există
                        ; Regiune critică
        MOV   Sem,0     ; Anulează LOCK
```

Semnalul LOCK/ furnizează doar informaţii pentru I8086 sau I8088, privind necesitatea blocării magistralei pe perioada executării instrucţiei precedate de prefixul LOCK. Responsabilitatea pentru obţinerea magistralei mμP rămîne în sarcina arbitrilor de magistrală I8289,rezidenţi pe fiecare MP echipat cu μP I8086 sau I8088.

Pentru realizarea mecanismului de excludere mutuală μP-oarele pe 16 biţi care fac trecerea spre μP-oarele pe 32 biţi ca de exemplu, MC68000 ca şi μP-oarele pe 32 biţi au prevăzut instrucţii TAS distincte [KM81,Ma81] . Un exemplu de secvenţă TAS,realizată soft pentru această categorie de μP-oare de tip MC68000 pe 16 şi 32 biţi are următorul aspect:

```
Verif:  TAS   Sem    ; TAS blocat
        BNE   Verif  ; Buclă,dacă Semaforul nu este 0
                     ; Regiune critică
        CLR   Sem    ; Resetează Semaforul
```

In acest caz instrucţia TAS verifică dacă semaforul este resetat, fapt care indică disponibilitatea resursei,şi în caz afirmativ procedează la setarea acestuia. Pe perioada executării instrucţiei,magistrala sistemului mμP este blocată,similar ca şi în cazul discutat anterior.

2.7.3. Modalităţi de realizare a interconectării. Magistrale standard multimicroprocesor.

După cum s-a arătat se utilizează 3 organizări pentru interconec-

tarea MP-oare,a memoriei comune şi a resurselor de I/E. Acestea sînt:

- magistrală comună multiplexată în timp;
- memorie multiport;
- conectare matriceală de tip "crossbar"

Cea mai ieftină organizare,magistrala comună multiplexată în timp, este cea care s-a impus,existînd în prezent două tipuri de astfel de magistrale: MULTIBUS,realizată de Intel şi AMS-M realizată de Siemens. In cadrul acestor două tipuri de magistrală,care au devenit standarde în ceea ce priveşte modalităţile de interconectare,s-au dezvoltat o serie de module care permit o configurare rapidă funcţie de tipul aplicaţiei,a unei structuri μP. Celelelte două organizări pentru interconectarea MP-oare cunosc o dezvoltare mult mai redusă,constituind realizări singulare în cadrul arhitecturii μP. Datorită importanţei deosebite pe care o prezintă la stabilirea caracteristicilor unui sistem μP,în continuare se va efectua o foarte scurtă descriere a acestor două tipuri de magistrale.

Magistrala MULTIBUS

Liniile magistralei pot fi grupate în următoarele categorii: linii de adresă; linii de adresă bidirecţionale; linii de întrerupere multinivel; linii de control şi sincronizare. Liniile de adresă şi date sînt comandate de dispozitive three - state,în timp ce liniile de întrerupere şi alte linii de control sînt comandate de dispozitive cu colector deschis.

Modulele care folosesc sistemul MULTIBUS sînt interconectate într-o relaţie master / slave. Un master de magistrală poate comanda liniile de adresă şi comandă,aspect care face posibil asigurarea controlului magistralei de către acesta. Arhitectura MULTIBUS este prevăzută atît pentru module master (MP-oare) şi slave (blocuri de memorie comună,module de I/E),pe 8 cît şi pe 16 biţi. Este de menţionat că magistrala permite interconectarea a unu sau mai mulţi masteri de magistrală în contextul prezenţei a unu sau mai multe module slave. Arbitrarea magistralei devine funcţională cînd mai mult de un master cere controlul magistralei în acelaşi timp. Un CLK de magistrală este furnizat de obicei de un master de magistrala şi poate fi derivat independent de clockul MP. CLK-ul de magistrală oferă o referinţă de sincronizare pentru conţinutul magistralei între cererile de magistrală ale masterilor. Aceasta permite ca masterii cu viteză diferită să împartă resursele pe aceiaşi magistrală, transferurile pe magistrală efectuîndu-se asincron respectînd CLK-ul de magistrală.Prin aceasta viteza de transfer este dependentă numai

de dispozitivul de transmisie şi de recepţie. Proiectarea magistralei a prevenit situaţia ca modulele master lente să nu fie handicapate în încercările lor de a cîştiga controlul magistralei,dar nu a restricţionat viteza la care modulele mai rapide pot transfera date prin aceiaşi magistrală. Odată ce o cerere de magistrală este acordată,transferurile singulare sau multiple citire / scriere pot continua.

In prezent,în cadrul acestui standard de interconectare sînt prezente două tipuri de magistrale MULTIBUS I,destinate interconectării MP-oare dotate cu µP-oare pe 8 şi 16 biţi şi MULTIBUS II,destinat realizării de arhitecturi muP care includ şi uP-oare pe 32 de biţi [**86b] . Dintre modulele standard puse la dispoziţie de MULTIBUS II pentru realizarea unei structuri mµP pot fi menţionate:

- modulul iSPB Parallel System Bus care implementează magistrala MULTIBUS II,asigurînd o viteză de transfer de pînă la 4oMB/sec;
- modulul iSBC 286/loo Single Board Computer care reprezintă un MP realizat cu µP I8o286;
- modulul iSBC 386/loo Single Board Computer care reprezintă un MP realizat cu µP de 32 de biţi I8o386;
- modulul iSBC MEM/312,31o,32o,34o,care reprezintă o memorie comună biport de 0.5MB,1MB,2MB şi respectiv 4MB;
- modulul iSBC 186/224PP,reprezentînd un controler a perifericelor de I/E;
- modululiSBC CSM/001 pentru generarea CLK-ului de magistrală reprezentînd în acelaşi timp o unitate specializată pentru controlul şi coordonarea traficului pe magistrală MULTIBUS II.

Magistrala AMS-M

AMS-M este o magistrală multimaster paralel pentru transferul de date pe 8/16 biţi într-un spaţiu de memorie de 16MB. Magistrala permite interconectarea unui număr de pînă la 16 MP-oare master,în condiţiile unui transfer de date cu o viteză de pînă la loMB/sec. Ceea ce o deosebeşte de magistrala MULTIBUS II o constituie faptul că aceasta permite inclusiv transferul serial a datelor. Este de remarcat de asemenea faptul că magistrala deşi dispune de module proprii, permite interconectarea unor module standard care aparţin magistralei MULTIBUS II.

3. MECANISMELE SI COMPONENTELE UNUI SISTEM DE OPERARE MULTIMICROPROCESOR.

Un sistem de μP trebuie să implementeze un număr de trei funcţii sistem care se referă la:

- sincronizare;
- comunicare;
- excludere mutuală.

Mecanismele care implementează aceste funcţii sînt ierarhizate pe două nivele. La primul nivel,cel inferior,acestea sînt responsabile cu coordonarea execuţiei proceselor,reprezentînd din acest punct de vedere mecanismele standard ale unui sistem de operare concurent. In cadrul celui de al doilea nivel,funcţiile respective sînt responsabile cu coordonarea activităţii modulelor procesoare în cadrul sistemului μP,privit de data aceasta ca un tot unitar. Intre cele două nivele există o legătură strînsă,o serie de funcţii specifice fiecărui mecanism în parte interferîndu-se cu funcţiile aceluiaşi mecanism situat la un nivel diferit. Din acest motiv,analiza acestor mecanisme se va face începînd cu nivelul inferior,cel responsabil cu concurenţa la nivelul unui procesor,urmînd ca în momentul analizei nucleului unui sistem μP să se detalieze caracteristicile macanismelor care implementează aceste funcţii la nivelul ierarhic imediat superior.

3.1. Cooperarea proceselor.

Există un număr de două relaţii fundamentale de cooperare pe care le pot avea procesele şi anume:

- utilizarea unei resurse comune care nu a fost produsă de nici una dintre ele;
- de producător / consumator,în care un proces creează date care urmează să fie utilizate de către celălalt proces.

Aceste două tipuri de cooperare [HH81] au generat şi impus un număr de patru probleme: excluderea mutuală,sincronizarea,interblocarea, determinarea. Modul în care s-au rezolvat aceste probleme au condus la definirea funcţiilor şi a mecanismelor corespunzătoare amintite anterior.

- Excluderea mutuală se referă la necesitatea de a forţa utilizarea strict secvenţială a unei resurse de către procesele care intră în competiţie pentru cîştigarea ei;
- Sincronizarea se referă la necesitatea coordonării operatii-

lor pe o resursă comună efectuate de procesele care cooperînd, solicită resursa în cauză;

- Interblocajul descrie situaţia în care procesele sînt puse mutual în stare de aşteptare a unor evenimentee cauzate de alte procese. In cazul unei cooperări de tip producător / consumator în care relaţia de cooperare este bilaterală,fiecare proces poate fi pus în aşteptarea unor date care sînt produse de celălalt; deoarece ambele sînt în aşteptare nimic nu se produce de nici un proces,fapt care cauzează apariţia interblocajului;

- Determinarea se referă la posibilitatea ca procesele care au acces la o zonă de date comună să le modifice în intervalele de timp în care alte procese care au început să utilizeze zonele în cauză sînt puse în stare de aşteptare. Acest aspect impune ca alterarea zonelor de date comune să fie astfel coordonată încît acestea să fie întotdeauna determinate în momentul în care un proces şi-a întrerupt acţiunea asupra lor.

3.2. Regiuni critice şi semafoare.

Mecanismele menţionate anterior sînt realizate prin utilizarea unui număr de trei concepte şi anume:

- regiune critică;
- operaţie indivizibilă;
- semafoare.

Regiunea critică reprezintă o secţiune de cod dintr-un sistem concurent care poate fi controlată numai de un proces la un moment dat. Din modul de definire apare cu claritate faptul că cel mai important aspect în prelucrarea controlului unei regiuni critice de către un proces îl constituie modul în care acesta obţine intrarea şi respectiv realizează ieşirea din regiunea critică. O primă modalitate de rezolvare o constituie definirea unei variabile de stare logice (realizată prin soft sau hard) a cărei valoare adevărată sau falsă indică dacă regiunea este ocupată şi deci procesul trebuie să aştepte,sau dacă este liberă şi deci poate fi ocupată. Modificarea valorii variabilei de stare este făcută de proces la intrarea,respectiv ieşirea din regiunea critică. Această modalitate,reprezintă cea mai simplă formă de acces la o regiune critică şi a fost definită de Dijkstra sub numele de operaţii P şi V pe semafoare (în acest caz semaforul apare ca o simplă variabilă de stare). Aceste nume s-au ge-

neralizat,astfel încît în prezent,operaţia de acces la o regiune critică (resursă comună) este cunoscută sub numele de operaţie primitivă P pe semaforul care indică starea resursei,iar eliberarea acesteia ca o operaţie primitivă V executată de proces în momentul părăsirii regiunii critice [Di65] . In cazul în care cererile de acces apar la intrarea regiunii critice asincron,nu apare nici o problemă; dacă însă acestea apar sincron,este necesară pe lîngă existenţa unei proceduri de arbitrare şi a unei modalităţi de protejare a variabilei de stare pe perioada testării şi eventual a modificării acesteia,aspect care a condus la conceptul de operaţie indivizibilă.

Operaţia indivizibilă reprezintă o secvenţă de cod care nu poate fi întreruptă de nici un eveniment hard sau soft al sistemului. Din acest punct de vedere însăşi operaţia indivizibilă reprezintă o regiune critică mică,protejată însă de influenţa oricăror evenimente externe în momentul execuţiei. Dacă se indică cu paranteze drepte indivizibilitatea,atunci procedura de intrare,respectiv ieşire dintr-o regiune critică pe care o poate apela un proces va avea următorul aspect în pseudocod:

```
Procedure: Intrare în regiunea critică
    Begin
      Test: If [Var=falsă then Var:=adevărat
                else] go to Test;
    End
Procedure: Ieşire din regiunea critică
    Begin
      [Var:=falsă;]
    End
```

Modul acesta de rezolvare (care are ca echivalent procedura "test and set" (TAS),analizată într-un capitol precedent),prezintă dezavantajul că procesul trebuie să efectueze continu procedura de testare a variabilei de condiţie,pînă în momentul în care prin modificarea acesteia de procesul care ocupa regiunea critică primeşte dreptul de acces. Această formă de aşteptare activă măreşte considerabil încărcarea procesului şi ca urmare conduce la o creştere artificială a gradului de încărcare a acestuia. Dezavantajul a condus la apariţia conceptului de semafor,care în prezent reprezintă soluţia care se acceptă în majoritatea cazurilor.

Semaforul reprezintă o construcţie logică utilizată pentru controlul intrării şi ieşirii dintr-o regiune critică fără a produce fe-

nomenul de aşteptare activă,fiind alcătuit dintr-o variabilă de stare numerică,un şir de aşteptare ca şi din proceduri pentru introducerea şi extragerea proceselor din şirul,de aşteptare.

Pentru înlăturarea fenomenului de aşteptare activă s-a introdus o stare nouă denumită WAIT,în care sînt întîrziate (suspendate) procesele care au găsit regiunea critică ocupată. In acest context semaforul este compus dintr-un şir de aşteptare şi o variabilă de stare numerică cu rol de contorizare. Aceasta din urmă reprezintă o extensie a variabilei de stare logice din metoda prezentată anterior avînd un dublu rol. Pe de o parte,este utilizată pentru protecţia regiunii critice,avînd o valoare iniţială unu,care devine zero şi mai mică ca zero,cînd regiunea critică este ocupată,iar pe de altă parte are rolul de a contoriza numărul de procese care au fost suspendate şi care deci pot fi eventual restartate,valoarea fiind dată de modulul conţinutului variabilei de stare.

Operaţiile pe semafoare în pseudocod sînt următoarele:

```
/ * Declarare semafor * /
    Type Semafor=record
           Contor: integer;
              Sir: Sir-procese End
/ * Iniţializare semafor * /
    Procedure: Semafor-Init (semafor)
        Begin
         Contor.semafor:=1; Pentru protecţia regiunii critice
         Iniţializează Sir.semafor;Funcţie de disciplina şirului
        End
/ * Intrare în regiunea critică-primitiva P * /
    Procedure: WAIT (semafor)
        Begin
        [Contor.semafor:=Contor.semafor-1;
         If Contor.semafor < 0 then
            Begin
              Identifică procesul apelant;
              Introdu procesul în Sir.semafor;
              Restartează următorul proces din GDE;
            End]
        End WAIT;
/ * Ieşire din regiunea critică-primitiva V * /
    Procedure: SIGNAL (semafor)
```

```
Begin
[Contor.semafor:=Contor.semafor + 1;
 If Contor.semafor ≤ 0 then
 Begin
   la următorul proces din Sir.semafor;
   Pune procesul în şirul proceselor gata de execuţie;
 End]
End SIGNAL;
```

In rezumat orice semafor standard este format dintr-un şir de aşteptare şi o variabilă de stare aferentă cu rol de contorizare. Dacă un proces cere accesul într-o regiune critică,execută două operaţii asupra semaforului aferent regiunii în cauză şi anume:

- o operaţie de tip P; utilizînd procedura WAIT;
- o operaţie de tip V; utilizînd procedura SIGNAL.

In cadrul acestor proceduri (în cadrul sistemului de operare se utilizează numele de primitivă pentru aceste prodeduri pentru a le deosebi de procedurile proceselor) se realizează pentru:
WAIT;dacă variabila de stare decrementată cu unu este 0,procesul are acces la resursă; în caz contrar (mai mică ca 0),procesul este suspendat (întîrziat) într-o stare de aşteptare în şirul aferent semaforului şi se încearcă relansarea următorului proces prezent în şirul de aşteptare a proceselor gata de execuţie ataşat procesorului.
SIGNAL; ieşirea din regiunea critică; dacă variabila de stare incrementată cu unu,este mai mică sau egală cu 0,procesul suspendat în şirul de aşteptare al semaforului este trecut în şirul de aşteptare a proceselor gata de execuţie (GDE) ataşat procesorului.

Este de remarcat în final faptul că operaţiile pe semafoare reprezintă de asemenea regiuni critice,fapt semnalat în pseudocod prin utilizarea parantezelor drepte.

3.3. Analiza mecanismelor de sincronizare şi excludere mutuală.

Au rolul de a asigura coordonarea execuţiei proceselor,astfel încît în contextul asigurării corecte a înlănţuirii logice impuse de destinaţia sistemului,să permită unui proces activ să se blocheze sau să suspende,respectiv să activeze alte procese. Aceste cerinţe sînt dictate de necesitatea ca de-a lungul întregului proces să se asigure transmisia corectă a datelor între procese ca şi determinismul zonelor de informaţie comună. Realizarea acestor cerinţe se obţine utilizînd conceptele prezentate anterior,iar folosirea lor a condus la următoarele tipuri de metode care implementează mecanismele de sincronizare şi

excludere mutuală [Th86,An78] .

Metoda aşteptării active

Este o soluţie aplicată în cadrul utilizării în excluziune mutuală a unei resurse critice. Metoda implementează cea mai simplă formă de sincronizare în care sînt utilizate operaţii de tip P şi V pe variabila care defineşte starea resursei. Metoda este simplu şi uşor de aplicat,avînd însă dezavantajul apariţiei stării active de aşteptare pentru procesele care nu au reuşit intrarea în regiunea critică.

Metoda semafoarelor

Metoda realizează sincronizarea proceselor utilizînd operaţii de tip P şi V pe semaforul care controlează starea resursei. Metoda prezintă avantajul că înlătură fenomenul de aşteptare activă procedînd la trecerea proceselor care nu au primit acceptul de intrare într-un şir de aşteptare în care rămîn suspendate pînă în momentul în care este permis accesul la resursă. Dezavantajul constă în complicarea procedurilor de intrare şi ieşire din regiunea critică ca urmare a apariţiei unei noi stări în care poate fi prezent un proces. In pofida acestei limitări,metoda este larg folosită pentru definirea mecanismelor de sincronizare.

Metoda prezintă o variantă cunoscută sub numele de sincronizare indirectă prin semafoare. Aceasta constă în utilizarea unor semafoare private; un semafor este considerat privat unui proces,dacă acesta poate utiliza operaţii de tip P şi V,în timp ce oricărui alt proces i se autorizează numai utilizarea primitivei V. Semaforul în aceste condiţii are valoarea iniţială întotdeauna egală cu O. In acest ultim caz,ocuparea regiunii critice poate fi făcută de procesul care are autorizată numai utilizarea primitivei V o singură dată,sau în forma unei comenzi de lansare efectuată de procesul care are implementate ambele primitive.

Metoda sincronizării directe

Operaţia de sincronizare este realizată prin intermediul relaţiilor logice între procese,rezolvate în sensul în care acestea interacţionează în timp. Sincronizarea trebuie înţeleasă în cadrul acestei metode în sensul stabilirii unor relaţii de ordine între instrucţiunile executate de procese. Pentru a realiza acţiunea directă a unui proces asupra altuia,primitivele utilizate trebuie să accepte ca parametrii,procesul asupra căruia se intenţionează se se acţioneze.

Acţiunea este deci programată explicit şi cunoaşterea identităţii

procesului este absolut necesară.

Metoda sincronizării indirecte prin evenimente

Operaţia de sincronizare este realizată prin intermediul evenimentelor. Acestea sînt definite ca reprezentînd situaţii care pot provoca o schimbare în starea unui proces. Procesele care cooperează îşi coordonează în acest caz acţiunile utilizînd un set de variabile de stare comune în care se memorează informaţii privind evoluţia în timp a evenimentelor care se iau în considerare. Este evident în acest caz, că setul de variabile reprezintă o regiune critică la care accesul se face utilizînd metoda semafoarelor. Implementarea evenimentelor se face diferenţiat; astfel la un moment dat,un proces poate aştepta îndeplinirea mai multor evenimente,sau un eveniment poate fi aşteptat de mai multe procese.

Pentru ultimele două metode avantajul utilizării constă în simplificarea considerabilă a nucleului sistemului de operare ca urmare a reducerii numărului şi complexităţii primitivelor cerute; dezavantajul constînd în faptul că este necesară cunoaşterea exactă a derulării în timp a execuţiei proceselor şi a apariţiei evenimentelor care le activează,aspect care este aplicabil numai unor sisteme mμP strict specializate şi în care numărul proceselor este relativ mic.

3.4. Mecanisme de comunicare utilizate în coordonarea centralizată.

Coordonarea interacţiunii dintre procese nu se limitează numai la sincronizarea execuţiei acestora prin emiterea unor ordine de activare,respectiv de dezactivare de către un proces către alt proces.

Un alt aspect al coordonării îl constituie schimbul de informaţii între procese care implică definirea unui mecanism de comunicare căruia i se impun realizarea a două obiective. Primul obiectiv se referă la asigurarea schimbului corect de date între procesele care comunică,iar cel de al doilea constă în realizarea sincronizătii dintre procesele aflate în dialog utilizînd mecanismele de sincronizare existente ale sistemului de operare. Se utilizează două metode de comunicare;

- prin zone comune;
- prin mesaje.

Ambele metode utilizează mecanisme specifice de gestiune a comunicaţiei. Datorită faptului că ambele metode necesită o analiză detaliată a noţiunii de proces,mecanismele în cauză se vor analiza în exten-

so după discutarea caracteristicilor conceptului de proces aşa cum apare în sistemele mμP.

3.4.1. Conceptul de proces.

O definiţie general acceptată pentru noţiunea de proces este aceea prin care procesul este definit ca reprezentînd activitatea rezultată din execuţia unui program secvenţial care lucrează cu date şi variabile proprii de către un procesor [Sh74] . Logic,fiecare proces are propriul lui procesor şi program. In realitate însă,două procese diferite pot fi împărţite de acelaşi program ori acelaşi procesor. Astfel un proces nu este echivalent cu un program,după cum nu este echivalent cu un procesor; în fapt reprezentînd o pereche procesor,program în execuţie.

Execuţia unui proces poate fi descrisă ca o secvenţă de vectori de stare $s_0, s_1, \ldots s_i, \ldots$, în care fiecare vector de stare s_i reprezintă cantitatea de informaţie cerută de procesor pentru o execuţie. Componentele vectorului de stare s_i a procesului pot fi schimbate fie prin execuţia procesului în cauză căruia îi aparţin,fie prin execuţia altor procese care împart (utilizează în comun) unele din componentele în cauză. In această ultimă situaţie spunem că procesele cooperează. Este evident că sînt prezente şi componente ale vectorului de stare s_i care sînt modificate numai de procesul căruia îi aparţin,acestea formînd setul variabilelor proprii [Pa83] . Din modul în care s-a definit noţiunea de proces şi modul în care acesta se execută apar două categorii:

- procese independente care nu cooperează cu nici un alt proces;
- procese concurente,care interacţionează unul cu celălalt. Ele pot coopera modificîndu-şi reciproc componentele vectorului de stare s_i prin transmiterea de informaţii,sau intra în competiţie pentru utilizarea anumitor tipuri de informaţii sau echipamente comune definite ca resurse.

Dacă se examinează un proces în orice moment de timp,acesta poate fi: în execuţie sau blocat. Un proces este (logic) în execuţie, dacă este fie executat pe un procesor sau ar putea fi executat de un procesor dacă acesta este disponibil; în acest ultim caz,procesul este într-o stare gata de execuţie. Un proces este blocat dacă nu poate intra în execuţie pînă nu primeşte un semnal,mesaj,sau resursă de la un alt proces.

Prin examinarea logică a proceselor şi ignorînd numărul de procesoare fizic disponibile se pot asigura soluţii pentru orice tipuri de probleme ridicate de sistemele de operare mμP; aceasta cu condiţia asigurării unei cooperări corecte între procese,indiferent dacă împart sau nu aceleaşi procesoare.

3.4.2. Modalităţi de realizare a comunicării prin intermediul zonelor comune.

In cadrul comunicaţiei prin zone comune între procese,mecanismul fundamental constă în prezenţa unei zone comune în care un proces producător amplasează o serie de elemente şi din care un proces consumator preia elementele,la care se adaugă un mecanism pentru a determina starea zonei tampon. In cadrul acestui proces de intercomunicaţie,prezentat în fig.3.1.,relaţiile dintre cele două procese implicate se schimbă,unul fiind dependent de celălalt datorită modului în care se complectează,respectiv se extrage informaţia din zona tampon comună.

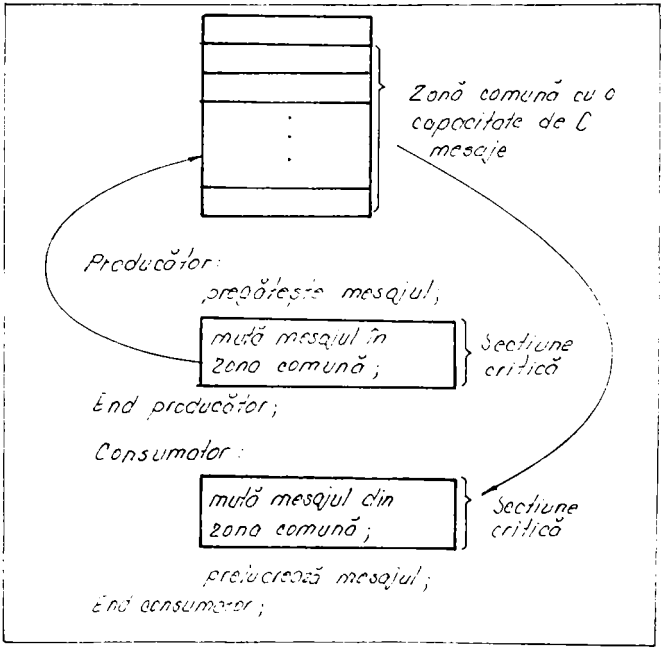


Fig.3.1. Intercomunicaţia producător / consumator cu utilizarea unei zone comune.

Apariţia acestui aspect implică o generalizare a noţiunii de semafor în sensul că acesta devine o variabilă de contorizare în loc de a

fi o variabilă logică [HH81,We77] .

In prezent există două metode de implementare a mecanismului de intercomunicaţie cu utilizarea de zone comune de tip producător / consumator. Cele două metode sînt:

- utilizarea unei comunicaţii directe producător / consumator
- realizarea comunicaţiei prin intermediul monitoarelor.

Comunicaţia directă producător / consumator.

In cadrul acestui mod de comunicare fiecare proces are acces la zona comună şi fiecare proces efectuează operaţii direct pe semafoarele generalizate. Cînd zona comună este plină,procesul producător nu va încerca să amplaseze un alt mesaj în zona comună,fapt care implică trecerea acestuia într-o stare de aşteptare. Procesul consumator va trebui să semnaleze procesului producător cînd o zonă mesaj din zona comună este liberă,astfel ca acesta să poată trece la amplasarea unei alte informaţii în zona eliberată. In cazul în care însă zona comună este liberă,cel care trece în stare de aşteptare este procesul consumator; stare din care va ieşi numai în momentul în care procesul producător va semnala că un alt mesaj a fost mutat în zona comună. Pentru implementarea acestui mecanism sînt suficiente definirea a două semafoare cu semnificaţia " zonă comună liberă ",respectiv," zonă comună complectată ",ca şi a două primitive WAIT şi SIGNAL. In scopul simplificării comunicaţiei dintre un proces producător şi unul consumator,se mai adaugă suplimentar un şir de aşteptare care va conţine adresele zonelor mesaj din zona comună care sînt complectate (sau depinzînd de strategia adoptată sînt libere), ca şi două primitive responsabile în principal cu gestiunea acestui şir: ATTACH şi DETACH. Datorită importanţei prezentate de acest mod de comunicare se va analiza în continuare cum se realizează,prezentîndu-se rolul fiecărei primitive implicate în dialog.

O primitivă WAIT (index semafor) apelată va cere sistemului de operare blocarea procesului consumator sau producător din care a fost apelată zona comună dacă aceasta este liberă sau complectată.

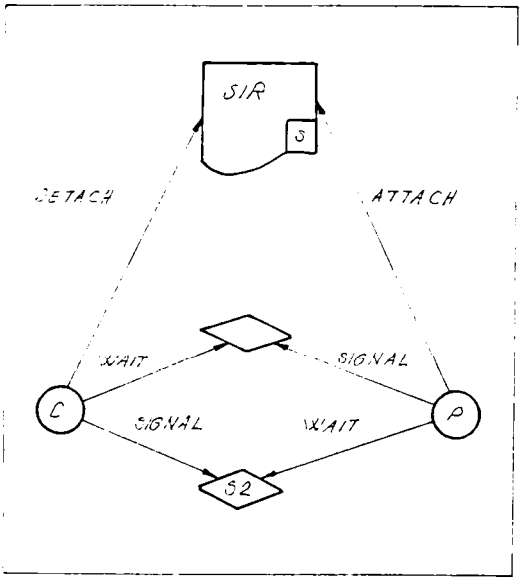
O primitivă SIGNAL (index semafor) apelată va cere sistemului de operare deblocarea unui proces consumator sau producător care a fost blocat din cauza condiţiei zonă comună liberă,respectiv zonă comună complectată.

O primitivă ATTACH (index şir,adresă zonă mesaj) va introduce adresa zonei mesaj care s-a complectat (sau s-a eliberat) în şirul de aşteptare a cărui index apare ca parametru.

O primitivă DETACH (index şir,adresă zonă mesaj) va scoate adresa zonei mesaj care s-a complectat (sau s-a eliberat) din şirul de aşteptare a cărui index apare ca parametru.

Graful mecanismului de intercomunicaţie de tip consumator / producător în care s-a notat cu:

S_1 semaforul condiţiei zonă comună liberă;
S_2 semaforul condiţiei zonă comună complectată;
SIR şirul de aşteptare a adreselor zonelor de mesaj ocupate;
S semaforul pentru controlul secţiunii critice SIR este prezentat în fig.3.2.



Producător
pregăteşte mesajul;
WAIT (S_1);
WAIT (S);
mută mesajul în zona comună;
ATTACH (SIR,adresă mesaj);
SIGNAL (S);
SIGNAL (S_1);
End Producător;
Consumator:
WAIT(S_2);
WAIT (S);
mută mesajul din zona comună;
DETACH (SIR,adresă mesaj);
SIGNAL (S);
SIGNAL (S_2);
prelucrare mesaj;
End Consumator;

Fig.3.2. Mecanismul intercomunicaţiei dintre procese de tip producător (P) / consumator (C).

Din modul de definire a acestui tip de comunicaţie rezultă că algoritmul de planificare şi sincronizare a dreptului de acces la zonele comune de către procese,este divizat,fiind prezent în fiecare pereche de procese care schimbă informaţii. Dacă pentru sistemele cu un număr mic de procese acest tip de comunicaţie este acceptabil pentru sistemele în care numărul acestora este mare apar complicaţii în concepera logică a procedurii de gestiune a zonelor comune ca şi în asigurarea unei structuri modulare a mecanismului de coordonare şi sincronizare. Ca un rezultat direct în acest caz are loc o creştere sub-

stanţială a timpului de răspuns a sistemului.

Realizarea comunicaţiei prin intermediul monitoarelor

Comunicaţia directă dintre procese,prezintă dezavantajul fragmentării procedurii de planificare şi sincronizare în fiecare proces implicat în schimbul de informaţie. Cumularea problemelor de planificare şi sincronizare într-o singură regiune critică a dat posibilitatea realizării schimbului de informaţie prin intermediul unei singure regiuni critice de cod,respectiv prin intermediul a ceea ce a primit denumirea de monitor. Modul în care un proces producător este sincronizat cu unul sau mai multe procese consumatoare,(vezi fig.3.3) presupune parcurgerea următoarelor etape:

- orice proces poate chema monitorul utilizînd o procedură de intrare cu un set fix de parametrii;
- toate procedurile de intrare includ o primitivă WAIT pe intrarea monitorului;
- parametrii de intrare sînt examinaţi de monitor şi procesul care a apelat monitorul este fie servit,fie pus în aşteptare pînă în momentul îndeplinirii unui set de condiţii,cînd poate fi servit.

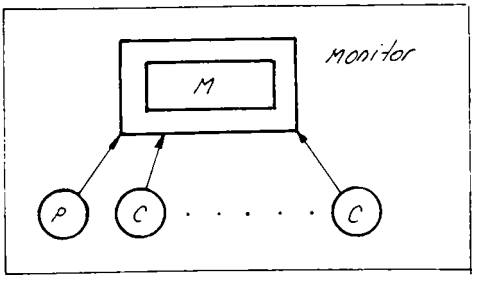


Fig.3.3. Mecanismul comunicaţiei prin monitor între procese de tip producător (P) / consumator (C).

3.4.3. Mecanisme de realizare a comunicării prin intermediul transmiterii mesajelor.

Prin extinderea noţiunii de comunicaţie între procese apare posibilitatea ca un proces activ să solicite la un moment dat trimiterea unui mesaj altui proces activ,ca şi posibilitatea ca un proces activ să solicite un mesaj de la un alt proces activ.

Acest schimb de mesaje în care pot fi implicate mai multe procese

emiţătoare,respectiv receptoare,poate condiţiona sau nu,execuţia pe mai departe a proceselor în cauză,motiv pentru care sînt necesare mecanisme pentru blocarea respectiv deblocarea proceselor implicate în schimbul de mesaje ca şi pentru determinarea modului în care se vehiculează mesajele: prin intermediul memoriilor comune,a porturilor de I/E sau pe linii de telecomunicaţie. Există două tipuri de intercomunicaţie prin mesaje. Primul se referă la un trafic unidirecţional al schimbului de mesaje,cel de al doilea la un proces generalizat de schimb de mesaje între procese [HH81] .

Mecanism de trafic unidirecţional

Acesta reprezintă un mecanism simplu în care conexiunile dintre procesele implicate sînt unice,fiind utilizate numai în intercomunicaţiile dintre procese de tip producător / consumator în care schimbul de mesaje este unidirecţional.

Primitivele fundamentale implicate în procesul de intercomunicaţie de tip mesaj,poartă numele de SEND şi respectiv RECEIVE.

SEND reprezintă primitiva prin care un proces trimite un mesaj altui proces. In cea mai simplă formă SEND va specifica în parametrii numele procesului căruia îi este adresat mesajul ca şi adresa mesajului. Mecanismul prin care mesajul ajunge la procesul destinatar poate fi implementat în două moduri.

In primul caz,mesajul este copiat într-un şir de aşteptare ataşat procesului destinatar. In cazul în care mesajul este destinat mai multor procese poate fi copiat în fiecare şir de aşteptare ataşat proceselor destinatare,sau într-o variantă mai economică într-un singur şir de aşteptare,cu specificarea numărului de procese destinatare. Execuţia primitivei SEND,poate fi urmată sau nu,de blocarea procesului expeditor pînă în momentul în care procesul destinatar primeşte mesajul.

In al doilea caz,se apelează la un proces specializat în gestionarea mesajelor. Procesul va dispune de şiruri de aşteptare pentru fiecare proces implicat în schimbul de mesaje,şiruri care sînt invizibile pentru procesele expeditoare. In momentul executării unui SEND,procesul de gestiune a mesajelor va fi activat şi mesajul va fi copiat în şirul de aşteptare corespunzător procesului destinatar. In continuare transmiterea mai departe a mesajului rămîne în sarcina procesului de gestiune a mesajelor care va întrerupe execuţia procesului destinatar pentru a forţa expedierea către acesta a mesajului în cauză. Sirurile care conţin mesaje schimbate între procesele expeditoare şi cele receptoare îşi modifică dinamic mărimea funcţie de

modul în care evoluează procesul de gestiune a mesajelor.

RECEIVE reprezintă primitiva pe care un proces destinatar o utilizează în două scopuri. Primul scop,constă în confirmarea primirii mesajului astfel încît acesta să poată fi prelucrat de destinatar. Cel de al doilea,constă în emiterea unei cereri de primire a unui mesaj din partea unui proces expeditor. In ambele cazuri procesul de gestiune a mesajelor este apelat procedînd fie la scoaterea mesajului din şir în primul caz,fie la înregistrarea cererii în cel de al doilea caz.

Mecanismul de expediere / primire a unui mesaj,prezentat mai sus implică o relaţie directă între expeditor şi destinatar,relaţie care în cazurile mai complexe este controlată de un proces de gestiune a mesajelor care se vehiculează. Din acest punct de vedere primitivele SEND şi RECEIVE,reprezintă analogul primitivelor ATTACH şi DETACH,prezentate în cazul comunicaţiei prin zone comune,problematica rămînînd aceiaşi,cu singura deosebire că zona comună conţine de data aceasta un singur mesaj. In această idee,diagrama de stare corespunzătoare modului de comunicaţie prin zone comune rămîne valabilă ca şi modul de implementare a primitivelor SEND / RECEIVE cu amendamentul prezentat mai sus.

Mecanism de trafic generalizat

Faţă de funcţiile mecanismului de intercomunicaţie prezentat anterior se introduc în plus funcţii referitoare la asigurarea:

- comunicaţiilor bidirecţionale;
- confirmării primirii unui mesaj;
- prezenţei unor căi multiple de vehiculare a mesajelor de diverse tipuri între procese.

Primitivele utilizate sînt aceleaşi,SEND respectiv RECEIVE,cu unele variaţii de formă impuse de noile funcţii care sînt încorporate. Din acest motiv,analiza acestui tip de mecanism,se va face din punctul de vedere a funcţiilor suplimentare care le introduc şi nu al modului de funcţionare a primitivelor în sine.

Confirmarea primirii unui mesaj reprezintă o notificare cerută de procesul expeditor,că mesajul a fost primit de către procesul de gestiune a mesajelor sau de către destinatar. Operaţia este independentă de faptul că expeditorul aşteaptă un răspuns din partea destinatarului şi poate fi realizată prin execuţia unui SIGNAL de către destinatar. Verificarea confirmării poate fi efectuată de expeditor printr-un WAIT (anumite sisteme de operare prevăzînd pentru aceasta o primitivă specifică numită CHECK (verifică) care are de fapt ace-

laşi rol cu WAIT). Confirmarea primirii este utilizată pentru deblocarea unui proces,sau pentru a permite unui proces activ trimiterea în continuare a altui mesaj.

Realizarea căilor multiple de vehiculare a mesajelor este asociată cu conceptul de cutie poştală. In cadrul acestui concept,un proces trimite un mesaj care include şi numele expeditorului printr-o operaţie SEND,la unul sau mai multe procese destinatar,după care poate verifica printr-un WAIT dacă mesajul a fost primit. O cutie poştală reprezintă un şir de aşteptare asociat cu un destinatar care poate primi mesaje de la două sau mai multe procese expeditoare. Conceptul de cutie poştală este utilizat în cazul în care procesul expeditor cunoaşte numai numele procesului căruia urmează să-i expedieze mesajul (fără a cunoaşte numele şirului asociat procesului destinatar). Astfel apare un şir unic de aşteptare a mesajelor primite de un proces în care fiecare mesaj în parte identifică procesul expeditor. Procesul destinatar poate astfel să confirme procesului expeditor primirea mesajului prin execuţia unui SIGNAL (care utilizează identificatorul prezent în mesaj). Gestiunea cutiilor poştale poate fi realizată în cadrul unui proces de gestiune a mesajelor conţinute în şirurile de aşteptare. Este de remarcat faptul că prin utilizarea acestui cocept,se asigură practic legături multiple între toate procesele prezente în sistem.

Comunicaţiile bidirecţionale au apărut practic ca o concluzie logică a prezenţei în mesaj a identificatorului procesului căruia îi aparţine şi a extinderii acestui concept de identificare şi la răspunsurile emise de destinatari. Astfel identificatorul unui răspuns conţine pe lîngă identificatorul procesului căruia îi este destinat şi un identificator legat de mesajul al cărui răspuns este. Aceasta face ca atît mesajele iniţiale cît şi răspunsurile să poată fi plasate în acelaşi şir de aşteptare. Ca urmare,şirurile de tip cutie poştală conţin în cazul general un amestec de mesaje şi răspunsuri care provin de la procesele care au intrat într-o comunicaţie tip mesaj cu procesul căruia îi este ataşat şirul. Acest aspect a schimbat clar semnificaţia primitivei RECEIVE,care are rolul de a solicita primirea unui mesaj (scoaterea lui din cutia poştală),şi de a impune introducerea unei primitive noi RECEIVE ANSWER,care pe lîngă funcţia realizată de RECEIVE asigură şi emiterea unui răspuns pentru mesajul primit. Blocarea procesului poate fi asociată cu ambele funcţii. O gestiune specială a răspunsurilor de către sistemul de operare nu este necesară deoarece acestea părăsesc sistemul.

3.4.4. Particularităţile mecanismului de comunicare între procese şi subprogramele de tratare a întreruprilor.

Subprogramele de tratare a întreruperilor (STI) reprezintă o componentă majoră a tuturor sistemelor de operare mμP. Un STI poate fi considerat un proces de tip special,deoarece pe de o parte prezintă structura şi toate caracteristicile funcţionale ale unui proces,iar pe de altă parte datorită evenimentelor care declanşează lansarea în execuţie a unui STI,apar restricţii în ceea ce priveşte mărimea intervalului de timp în care STI urmează să îşi termine execuţia.

Modul de transfer a informaţiei se poate face utilizînd cele două mecanisme de intercomunicaţie prezentate anterior,la care se adaugă următoarele restricţii:

- un STI nu poate aştepta pe un semafor datorită limitării timpului în care trebuie să îşi termine execuţia;
- deoarece marea majoritate a STI sînt destinate pentru a deservi echipamentele de I/E este necesară în acest caz,definirea unei proceduri care să activeze STI numai dacă echipamentul de I/E este pregătit.

Faptul că un STI nu poate aştepta pe un semafor,poate conduce la apariţia unui blocaj în cazul în care un proces în curs de execuţie într-o regiune critică protejată prin semafor este întrerupt de un STI care aşteaptă pe acelaşi semafor. Acest aspect a dus la modificarea mecanismului prin care procesele şi STI se protejează una faţă de cealaltă şi prin urmare implicit,a modului de acces la regiunea critică. Astfel într-un sistem monoprocesor se utilizează pentru protejarea proceselor unul faţă de celălalt semafoarele şi o procedură de dezactivare a întreruperilor pentru a proteja procesele şi STI de alte STI. In acest mod un STI nu va avea nevoie să aştepte înainte de a avea accesul la o regiune critică deoarece apariţia întreruperii înseamnă că regiunea critică este liberă. Metoda este complectată în sistemele multimicroprocesor prin utilizarea unei combinaţii de "test and set" şi de blocare a magistralei.

Comunicaţia dintre un proces producător şi un STI consumator

In cazul condiţiei "zonă comună complectată" nu apare nici o problemă,procesul producător rămînînd în aşteptare pe semafor pînă în momentul în care STI a terminat de prelucrat informaţia şi semnalează ataşarea zonei tampon în cauză la şirul zonelor tampon libere. Rămîne de rezolvat problema în cazul condiţiei "zonă comună liberă" deoarece STI lansat în execuţie de o întrerupere provenită de la un

echipament de ieşire (E) nu poate aştepta pe semafor pînă în momentul în care are loc complectarea de procesul producător cu date a zonelor de mesaje care urmează să fie prelucrate. Problema se rezolvă în două moduri.

In primul caz se utilizează un STI lansat în execuţie de un ceas de timp real cu rolul de a verifica dacă echipamentul de E este disponibil şi zona tampon este complectată. In cazul în care cele două condiţii sînt realizate se lansează STI-ul consumator. Metoda are avantajul că menţine separarea dintre procese şi STI şi dezavantajul că adaugă mai multe întreruperi care urmează să fie prelucrate de procesor. In cel de al doilea caz,se construieşte o procedură care este chemată de procesul producător în momentul în care zona tampon a fost complectată. Procedura este prantic încorporată în procesul producător şi are rolul de a verifica dacă echipamentul de E este disponibil iar în caz afirmativ de ai furniza primul caracter al mesajului. In continuare,echipamentul de E va emite întreruperile care vor lansa în execuţie STI-ul consumator. Această soluţie deşi nu menţine modularitatea dintre procese şi STI are avantajul că este eficientă. Mecanismul de transfer este prezentat în fig.3.4.,în care cu S_1 s-a notat semaforul condiţiei zonă comună liberă,iar cu P procedura de declanşare a echipamentului dacă acesta este pregătit.

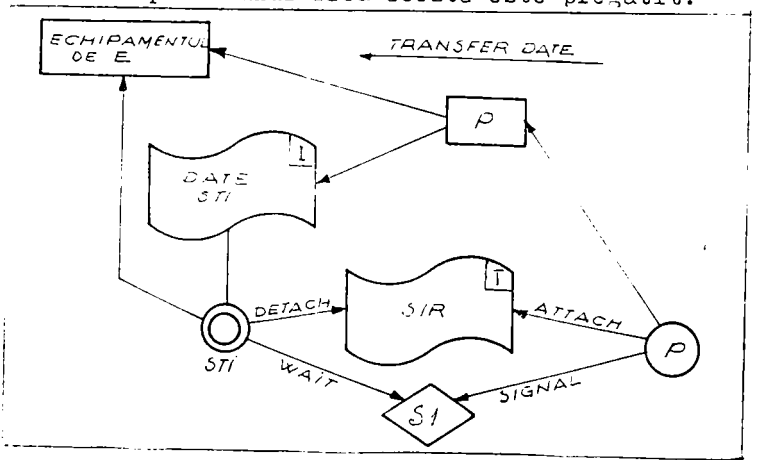


Fig.3.4. Mecanismul de comunicaţie dintre un proces producător şi un STI consumator.

Comunicaţia dintre un STI producător şi un proces consumator

In cea de a doua situaţie procesul producător care este de tip STI, furnizează informaţii pentru un proces consumator. In cazul condiţiei "zonă comună liberă",sincronizarea se realizează similar ca în cazul

în care STI este de tip proces producător,deoarece acesta din urmă va efectua numai o semnalare a faptului că zonele tampon sînt complectate. Problema care rămîne de rezolvat se referă la asigurarea sincronizării pe condiţia "zonă comună complectată",deoarece STI nu are posibilitatea de a aştepta pe semafor. Si în acest caz există două modalităţi de rezolvare a problemei.

Prima modalitate presupune definirea a două sau mai multe zone tampon care să fie complectate circular astfel încît STI să nu fie pus în situaţia de a aştepta. Problema care se pune în acest caz,este de a defini un număr suficient de mare de zone tampon astfel ca să nu existe şansa ca STI să fie pus în situaţia să aştepte.

A doua modalitate presupune dezautorizarea întreruperilor de la echipamentul de intrare (I) de către STI după complectarea zonelor tampon. In continuare rămîne în sarcina procesului consumator ca să reautorizeze întreruperile de la echipamentul de I în momentul în care a terminat de prelucrat datele din zona tampon. De asemenea procesul consumator va lansa în execuţie un modul pentru restartarea echipamentului de I în cauză. Erorile care pot apare în acest caz se datoresc datelor care se pierd între momentul dezautorizării întreruperilor şi a restartării din nou a echipamentului de I,respectiv a reautorizării întreruperilor. Mecanismul de transfer este complementar celui prezentat în fig.3.4.

Comunicaţia dintre un STI producător şi un STI consumator

In cea de a treia situaţie,ambele procese producător şi consumator sînt de tip STI. Realizarea comunicaţiei presupune evitarea fenomenelor de aşteptare,aspect care se realizează dezautorizînd întreruperile în STI aflat în execuţie pentru STI-ul complementar producător sau consumator. Ca şi în celelalte două tipuri de comunicaţie, sînt necesare module independente de restartare a echipamentelor periferice care sînt gestionate de STI consumator şi producător în cauză.

3.5. Extinderea mecanismelor de cooperare în sistemele multimicroprocesor. Monitoare.

3.5.1. Natura şi structura monitoarelor.

Un monitor reprezintă generalizarea unei regiuni critice de cod şi de date care este utilizată în comun de diferite procese în conformitate cu o anumită disciplină de planificare [BB8o] . Prin urmare un monitor reprezintă o entitate statică care este activată de pro-

cesele care solicită accesul la zonele de date comune incluse,şi care astfel activează procedurile care fac parte din monitor. Un monitor poate fi utilizat ca o structură de sincronizare a proceselor concurente cu rol de asigurare a comunicaţiilor între procese şi planificare a acestora (aceasta din urmă fiind privită nu ca o planificare conform unei anumite discipline ci ca o asigurare a înlănţuirii proceselor care se apelează unul pe celălalt). Pentru realizarea acestui deziderat sînt folosite combinaţii de condiţii de stare şi şiruri de aşteptare,ansamblu cunoscut sub numele de variabile condiţionate. Variabilele condiţionate,pot fi implementate direct sau utilizînd semafoare. Astfel monitorul apare ca o colecţie de proceduri asociate cu un tip de resursă,fiind responsabil de alocarea resursei şi de controlul referinţelor la aceasta. Fiecare resursă din sistemul mμP are un monitor specific. Procesele nu pot opera direct pe zonele de date comune; ele pot numai chema procedurile monitor care asigură accesul la datele comune. Pentru a asigura consistenţa,monitorul permite numai ca un singur proces să fie activ la un moment dat. Dacă alte procese apar în momentul în care procedurile acestuia sînt executate,procesele în cauză vor rămîne în aşteptare. Pentru realizarea acestor sarcini codul program al unui monitor poate fi privit ca avînd două părţi logice. Prima este reprezentată de algoritmul pentru utilizarea resursei împărţite (comune). Cea de a doua,este responsabilă cu mecanismul de planificare a accesului proceselor la resursa în cauză. Astfel un monitor din punct de vedere a structrurii de date şi cod are următoarea alcătuire:

- zona de date proprii ale monitorului;
- zona de definire a procedurii de intrare;
- zona de definire a primitivelor.

Datorită importanţei avute în gestionarea resurselor unui sistem mμP fiecare din aceste părţi componente se vor analiza în următoarele paragrafe.

3.5.2. Tipuri de monitoare utilizate în sistemele multimicroprocesor.

Un proces care a cîştigat dreptul de a utiliza resursa comună gestionată de monitor se execută sub controlul exclusiv al acestuia. In cursul execuţiei procesul poate cere verificarea îndeplinirii unor anumite condiţii. In cazul în care acestea nu sînt îndeplinite,monitorul blochează execuţia pînă în momentul în care condiţiile cerute sînt îndeplinite. Modul în care monitorul gestionează aceste procese,le

blochează şi respectiv le relansează în execuţie,defineşte mecanismul de planificare,scopul final al acestuia,fiind de a garanta controlul exclusiv a resursei de către monitor. Din acest motiv,mecanismul de planificare reprezintă şi criteriul de clasificare a monitoarelor [Bu78] . Modul de reprezentare a activităţii de coordonare şi planificare a execuţiei proceselor este făcut utilizînd modelele unor automate cu stări finite (reprezentarea grafică efectuîndu-se cu diagrama de stare).

Toate tipurile de monitoare existente în prezent pot fi descrise utilizînd un număr de cinci stări:

- WAIT;(ASTEPTARE) la intrarea monitorului; procesul aşteaptă ca monitorul să devină disponibil.
- ACTIVE (ACTIV); procesul este executat exclusiv sub controlul monitorului.
- BLOCKED (BLØCAT); apare ca un rezultat al apelului efectuat de un proces activ care se pune el însuşi în această stare,în aşteptarea îndeplinirii a uneia sau a mai multor condiţii.
- PENDING (INTIRZIAT); apare ca un rezultat al unei cereri de activare pe care o primeşte un proces aflat în aşteptare din partea unui proces activ. Reprezintă tot o stare de aşteptare dar de maximă prioritate,fiind utilizată numai de monitorul de tip 3.
- ELIGIBLE (ELIGIBIL); reprezintă o stare intermediară între stările BLOCAT şi ACTIV,în care procesele care au fost activate aşteaptă ca procesul ACTIV să părăsească monitorul; este utilizată numai de tipul de monitor 4.

Evenimentele care provoacă tranziţiile între stările automatului sînt aceleaşi pentru toate tipurile de monitoare; deşi există modalităţi şi tipuri de implementări diferite. Cele patru evenimente principale sînt:

- ENTER (INTRARE); cerere emisă de un proces aflat în exteriorul monitorului,în scopul cîştigării controlului acestuia. Procesul care a efectuat cererea,fie închide intrarea monitorului intrînd în starea ACTIV,fie intră în stare ASTEPTARE la intrarea monitorului.
- EXIT (EXIT); corespunde momentului în care procesul părăseşte monitorul; ca urmare se redă controlul monitorului pentru activarea procesului cu cea mai mare prioritate aflat în aşteptare (sînt excluse procesele aflate în starea BLOCAT),sau pentru redeschiderea intrării monitorului; în unul sau în celălalt caz,apa-

re în urma unui apel terminat prin RETURN.

- BLOCK (BLOCARE); procesul activ prezent în urma acestui eveniment intră în starea BLOCAT sau INTIRZIAT,unde aşteaptă un apel de deblocare; controlul este redat monitorului pentru procesul cu prioritatea cea mai mare aflat în stare de aşteptare,sau pentru deschiderea intrării monitorului.

UNBLOCK (DEBLOCARE); reprezintă evenimentul care trece un proces din starea BLOCAT în starea ACTIV,ELIGIBIL,sau din nou în BLOCAT,funcţie de valorile variabilelor de stare şi tipul monitorului.

Datorită faptului că blocarea şi deblocarea unui proces reprezintă o operaţie complexă,o serie de monitoare au diversificat gama evenimentelor succeptibile să producă aceste stări. Astfel pentru BLOCARE apar:

SLEEP, STALL, DELAY, SUSPEND

iar pentru deblocare:

CONTINUE, WAKEUP, RESTART,PROCEED

Diferenţele care apar,se vor semnala în momentul în care se va prezenta fiecare monitor în parte. Reprezentarea acestora se va face utilizînd diagramele de stare în care cercurile vor defini stări,iar arcele,tranziţii dintr-o stare în alta.

In cazul în care pentru tranziţii se vor utiliza linii pline,se va semnala faptul că tranziţia dintr-o stare în alta este forţată de însăşi procesul care se execută,iar pentru tranziţiile definite prin linii întrerupte,faptul că procesul aflat într-o stare de aşteptare este forţat să treacă în altă stare în urma unui apel efectuat de procesul activ aflat sub controlul monitorului (prioritaţile se vor semnala prin numere încercuite,în care numărul 1 are cea mai mare prioritate).

Monitor de tip 1 (Pascal)

Figura 3.6a.,prezintă automatul pentru monitorul de tip 1 utilizat de limbajul concurent Pascal,în care apar suplimentar evenimentele DELAY (INTIRZIE) şi CONTINUE (CONTINUA) care realizează funcţiile de dezactivare şi de semnalizare pe intrarea monitorului şi respectiv de activare a unui proces. In fapt CONTINUE reprezintă o procedură specială de ieşire care este apelată de un proces activ pentru un proces aflat în stare de aşteptare (BLOCAT). In acest caz procesul care a activat procesul blocat va trebui să părăsească monitorul imediat. Acest tip de monitor prezintă dezavantajul major,că în situaţiile practice de planificare,apar cazuri în care două sau mai multe proce-

se pot fi activate înainte ca procesul care a amorsat procedura de activare să părăsească normal monitorul (deci numai un proces poate fi continuat la un moment dat atunci cînd condiţiile solicitate de acesta sînt îndeplinite).

Monitor de tip 2 (Manager)

Figura 3.6b.,prezintă automatul pentru monitorul de tip 2. Pentru blocarea şi respectiv deblocarea unui proces se introduc evenimentele RESTART (RESTART) şi respectiv SUSPEND (SUSPENDARE). Spre deosebire de monitorul de tipul 1,un proces activ poate reactiva (restarta) un număr nelimitat de procese care însă părăsesc imediat monitorul.

Monitor de tip 3 (Mediator)

Figura 3.6c.,prezintă,automatul pentru monitorul de tip 3. Pentru a înlătura dezavantajele prezentate de monitoarele de tip 1 şi 2 se adaugă o stare suplimentară TRANZITIE,astfel încît un proces poate planifica alte procese fără a fi silit să părăsească monitorul permanent. Procedura de planificare în cauză introduce un număr de evenimente suplimentare:

- PROCEED (TRANZITIE); reprezintă o cerere emisă de procesul activ de a fi trecut într-o stare tranzitorie în scopul deblocării unui proces BLOCAT (care devine fie activ,fie părăseşte monitorul).
- STALL (DEZACTIVEAZA); reprezintă o blocare a unui proces activ,urmată de relansarea unui proces prezent în starea TRANZITIE sau ASTEPTARE la intrarea monitorului. Dacă nu este posibilă reactivarea nici unuia,INTRAREA monitorului este deschisă.

O menţiune aparte trebuie făcută pentru evenimentul EXIT,deoarece procedura corespunzătoare este răspunzătoare suplimentar de investigarea stării TRANZITIE,înainte de a investiga starea ASTEPTARE pe intrare,pentru luarea în considerare a unui nou proces de către monitor. Din acest motiv evenimentul EXIT este înlocuit pentru acest tip de monitor prin DEPART (EXIT şi verificare).

Deşi monitorul prezintă o serie de avantaje faţă de tipurile prezentate anterior,are în acelaşi timp un dezavantaj semnificativ care pentru monitoarele ce sînt solicitate de un număr mare de procese devin greu de depăşit. Acesta se referă la faptul că structurile de date care sînt cerute de un proces care a pierdut temporar controlul pot fi modificate în mod necorespunzător de un proces care a cîştigat temporar controlul monitorului.

Monitor de tip 4 (Gladiator)

Monitorul prezentat în fig.3.6d.,înlătură dezavantajele monitorului de tip 3 prezentat anterior. Cînd un proces este activat de un alt proces,el intră într-o stare TRANZITIE din care procesele sînt selectate şi reactivate ulterior pentru a cîştiga controlul monitorului. Un proces poate avea astfel mai multe reactivări,fiindu-i garantat că nici un alt proces reactivat nu va avea posibilitatea de acces la structurile sale de date pînă în momentul în care renunţă la controlul monitorului. Pentru realizarea acestui aspect este adăugată o stare suplimentară ELIGIBIL,în care procesele blocate pot fi trecute prin apariţia evenimentului WAKEUP (ACTIVEAZA),timp în care procesul activ care deţine controlul monitorului îşi continuă execuţia. Două evenimente introduse suplimentar SLEEP (DEZACTIVEAZA) şi LEAVE (echivalentul lui EXIT),pot face ca un proces aflat în starea ELIGIBIL să treacă în starea ACTIV,(sau dacă nici un proces nu este în aşteptare să deschidă intrarea în monitor). Evenimentul WAKEUP (ACTIVEAZA) este astfel evident protejat,deoarece procesul activ menţine controlul monitorului,interferenţă avînd loc numai în timpul evenimentelor SLEEP (DEZACTIVEAZA) sau LEAVE (EXIT). Astfel datele monitorului nu se vor schimba în cursul execuţiei evenimentului ACTIVEAZA,singurele evenimente critice rămînînd DEZACTIVEAZA şi EXIT.

3.5.3. Structura de date şi organizatorică a unui monitor.

Indiferent de tipul monitorului,acesta va supraveghea procesul de coordonare a execuţiei proceselor de tip producator / consumator. Acest aspect face ca monitorul să supervizeze modul în care procesele preiau,creează sau completează o serie de zone tampon privite de acestea ca reprezentînd un grup de resurse comune.

Pentru realizarea acestui deziderat,monitorul dispune de două tipuri de structuri de date [DD81] . In cadrul primei structuri sînt definite datele utilizate în cadrul procesului de obţinere a controlului monitorului de catre un proces. Astfel condiţia monitor liber sau ocupat,are asociate semafoare controlate de nucleul sistemului de operare. Aceste semafoare sînt definite în cadrul structurii de date proprii a monitorului. In cadrul celei de a doua structuri de date monitorul dispune de şiruri de aşteptare în care sînt prezente adresele zonelor de date comune care sînt manipulate de procesele care solicită controlul monitorului,la care sînt asociate variabile de condiţie. Astfel în condiţiile monitorizării unor procese de tip producător / consumator,sînt

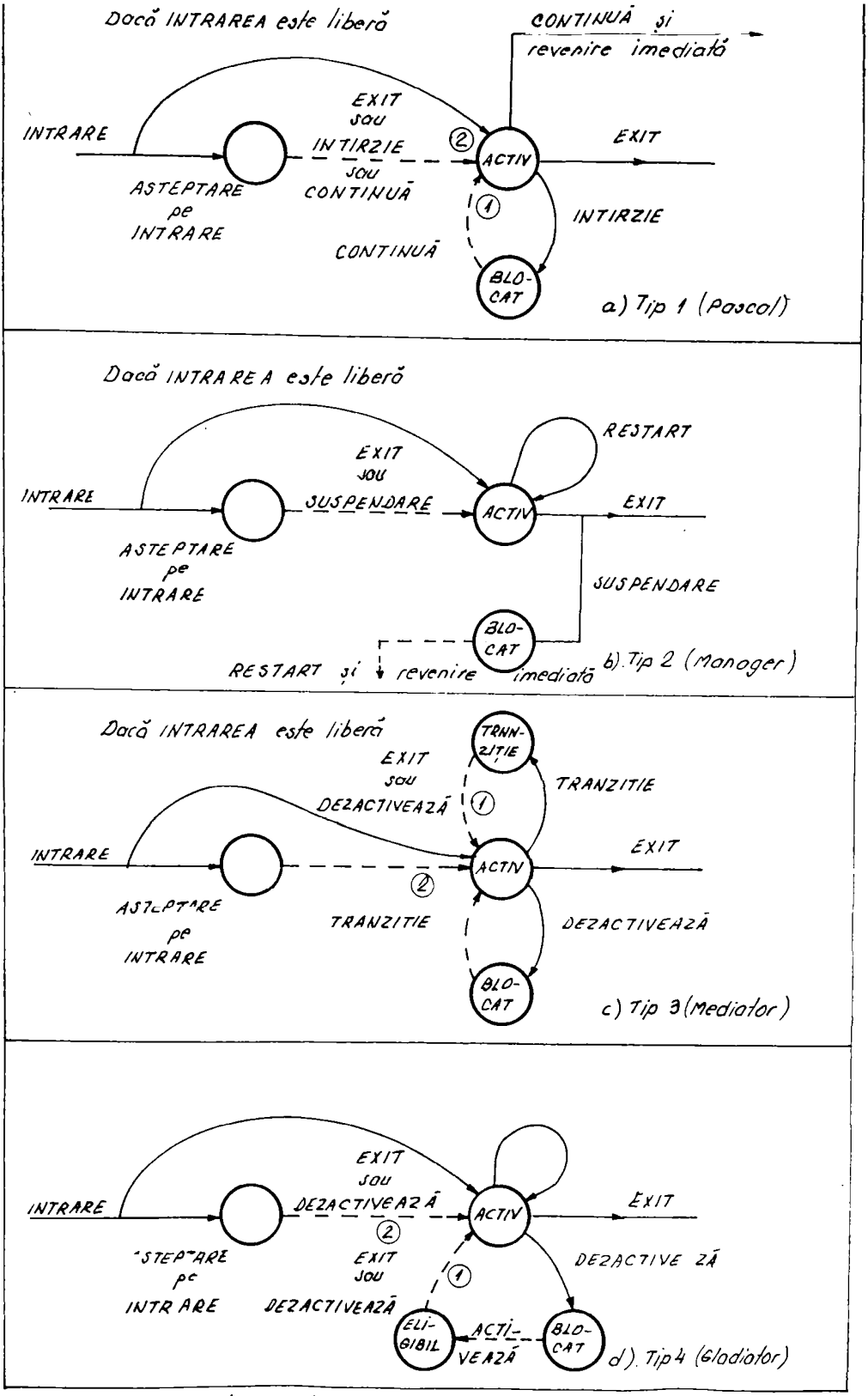


Fig. 5. Tipuri de monitoare

utilizate două tipuri de şiruri de aşteptare;

- şirul de aşteptare a zonelor de date comune libere;
- şirul de aşteptare a zonelor de date comune complectate;

la care sînt asociate următoarele variabile:

- aşteaptă pentru revenire;
- aşteaptă pentru execuţie.

Primitivele monitorului

Sînt definite de evenimentele care cauzează trecerea dintr-o stare în alta a automatului cu stări finite care implementează o anumită procedură de planificare monitor. Funcţie de tipul monitorului, primitivele diferă, procedurile care le implementează descriind modul de coordonare a execuţiei proceselor la resursa gestionată, în conformitate cu graful de stare a monitorului. Procedurile care realizează primitivele în cauză fac parte explicit din structura de cod a monitorului.

Interfaţa monitor/proces

Are rolul de a stabili cadrul în care urmează să se desfăşoare execuţia procesului sub controlul monitorului. Aceasta presupune apelarea din proces a procedurii de intrare în monitor, apelare care are un dublu scop:

- pe de o parte se solicită obţinerea controlului monitorului prin apelarea primitivei ENTER;
- pe de altă parte în caz de reuşită, de a poziţiona în şirul de aşteptare a zonelor de date comune, a adresei zonei care se va prelucra sau se va complecta. Din acest moment execuţia procesului se va face în continuare numai sub controlul monitorului. In cursul execuţiei procesul va trece dintr-o stare în alta a monitorului pînă în momentul în care îşi termină execuţia şi pierde controlul acestuia. Este foarte important de subliniat faptul că procedura de intrare în monitor, reprezintă de fapt descrierea modului în care un proces producător sau consumator se poate deplasa dintr-o stare în alta a automatului corespunzător monitorului utilizat. Cu alte cuvinte procedura de intrare în monitor reprezintă în fapt "descrierea monitorului" în cauză. Este evident că intrarea, respectiv ieşirea dintr-un monitor ca şi mecanismele implementate de partea de cod a acestuia, trebuie să fie transparente pentru codul proceselor care cer accesul la zona de date inclusă în monitor [Pa83].

3.6. Arhitectura nucleului sistemului de operare multimicroprocesor.

3.6.1. Funcţiile şi structura logică a nucleului.

Nucleul reprezintă o regiune critică care controlează funcţiile sistemului legate de fenomenul de concurenţă. Conform acestei definiţii, nucleul este o regiune critică de cod ca şi un monitor, reprezentînd astfel din punct de vedere logic, monitorul fundamental al sistemului. Orice activitate în care este cerută schimbarea stării unui proces va implica în mod obligatoriu nucleul. Această cerinţă, implică şi justifică faptul că însuşi nucleul este la rîndul lui o regiune critică. Pe lîngă această funcţie, într-un sistem mμP nucleul trebuie să gestioneze structurile de date care identifică MP-oare în care se vor executa numai anumite procese şi să asigure în fapt izolarea structurii hard de structura logică a sistemului [Pa83] .

Pentru a putea gestiona şi controla concurenţa dintre procese şi monitoare în contextul unei anumite structuri hard, o serie de aspecte legate de modul de acces în monitor, de execuţia unui proces ca şi de asigurarea unei structuri optime a sistemului de operare trebuie să îşi găsească răspuns în cadrul fiecărui nucleu. Din punct de vedere logic, nucleul unui sistem de operare mμP este divizat în trei părţi:

- nucleul principal;
- primitive şi structuri de date;
- interfaţa de intrare şi ieşire din nucleu.

Din punct de vedere structural, nucleul apare ca o procedură care este apelată de procese şi monitoare în care primitivele solicitate sînt definite de către parametrii. Această procedură are următorul aspect general:

```
Procedure NUCLEU (parametrii)
/ * INTRARE * /
  Dezautorizează întreruperile procesorului;
  Salvează registrele procesorului;
  Apelare NUCLEU cu adresarea arbitrului;
/ * PRINCIPAL * /
  Salvează adresa stivei procesorului;
  Validează parametrii;
  Apelare procedură;
/ * IESIRE * /
  Eliberare NUCLEU cu adresarea arbitrului;
  Restaurează registrele procesorului;
```

Autorizează întreruperile;
end

3.6.2. Interfaţa de intrare şi ieşire a nucleului.

Interfaţa de intrare şi ieşire a nucleului apare ca parte constitutivă a acestuia, reprezentînd legătura cu limbajul de nivel înalt în care sînt scrise procedurile corespunzătoare, proceselor şi monitoarelor. Ambele interfeţe trebuie să apară astfel structurate ca apelarea nucleului să apară sub forma apelului unui subprogram sau proceduri cu parametrii, a cărui sintaxă să fie conformă cu cea a limbajului în care este scrisă procedura (proces, monitor) din care se efectuează apelul. Acest aspect asigură transparenţa faţă de limbajele în care sînt scrise procedurile din care este chemat. Problemele care trebuiesc analizate la acest nivel se referă la structura interfeţei de intrare respectiv a celei de ieşire din nucleu şi la funcţiile pe care trebuie să le realizeze.

Interfaţa de intrare

Realizarea ei depinde de modul de localizare a nucleului, fie în memoria privată a unui MP destinat exclusiv realizării funcţiilor nucleului, fie în memoria comună, caz în care funcţiile sînt realizate de MP care solicită accesul în nucleu. Specificînd de fiecare dată deosebirile care apar în cazul celor două moduri de localizare, interfaţa de intrare trebuie să realizeze următoarele:

- Dacă funcţiile nucleului sînt realizate de MP care a cerut execuţia unei primitive a nucleului, este necesar ca interfaţa să asigure dezactivarea întreruperilor acestuia. Aceasta deoarece nucleul reprezintă o regiune critică protejată la întreruperile care pot apare de la diversele echipamente de I/E aparţinînd MP în cauză. Dezactivarea întreruperilor este urmată de salvarea registrelor procesorului, deoarece parametrii funcţiei apelate sînt transferaţi prin intermediul lor. Cele două aspecte dispar dacă funcţiile nucleului sînt realizate de un MP separat.

- Deoarece nucleul este o regiune critică, interfaţa trebuie să asigure apelul propriu zis al acestuia, modul de acces fiind din punct de vedere logic similar cu accesul în monitor. Si în această, situaţie apar două aspecte. Primul, se referă la necesitatea arbitrării cererilor de acces în nucleu, provenite de la diversele MP-oare ale sistemului mμP, cel de al doilea la necesitatea asigurării statutului de re-

giune critică a nucleului. Cele două aspecte prezintă rezolvări diferite funcţie de modul de localizare a nucleului. Dacă nucleul este prezent în memoria comună a sistemului μP,cererile de acces în nucleu sînt arbitrate de dispozitivul de alocare a magistralei la care au acces toate MP-oare,iar protecţia se realizează printr-un dispozitiv de tip "test and set",acelaşi care este utilizat în procesul de alocare a memoriei comune. In cazul în care nucleul are alocat un MP propriu,se impune utilizarea unui dispozitiv hard propriu de arbitrare a cererilor de acces,similar cu cel utilizat pentru alocarea magistralei. Protecţia regiunii critice se asigură în cazul acesta printr-un dispozitiv "test and set" propriu ataşat la MP în cauză.

Din cele arătate se observă că indiferent de modul de localizare a nucleului,sincronizarea cererilor de acces se face utilizînd o formă modificată de semafor (fără şir de aşteptare) realizată hard,aspect care impune utilizarea metodei aşteptării active a cererilor în faţa intrării nucleului. In majoritatea cazurilor aceasta reprezintă o soluţie optimă,deoarece întîrzierile introduse de execuţia primitivei solicitate sînt nesemnificative. Dacă însă aplicaţia impune un număr mare de apeluri lansate din procesele prezente în MP-oare (aspect care poate implica o blocare a traficului pe magistrala comună) devine necesară introducerea unui şir de aşteptare a apelurilor nucleului. Aceasta va conduce la un dispozitiv de arbitrare hard mult mai complex ca şi la adăugarea unui mecanism prin care MP-oare să fie puse în stare de aşteptare (aspect realizat fie prin trecerea în stare de aşteptare a microprocesorului existent pe MP în cauză,dacă acest lucru este posibil,fie realizînd o buclă de aşteptare soft din care ieşirea să se facă printr-o întrerupere generată de arbitru).

Interfaţa de ieşire

Indiferent de modul de localizare a nucleului,prima sarcină constă în eliberarea acestuia,operaţie realizată prin apelarea dispozitivului "test and set" în scopul resetării acestuia. Operaţia este urmată de autorizarea întreruperilor şi de restaurarea registrelor (dacă nucleul este localizat în memoria comună).

Nucleul principal

Este localizat între interfaţă şi procedurile care implementează primitivele nucleului avînd următoarele atribuţiuni:

- Efectuează salvarea adresei stivei procesului sau monitorului din care s-a efectuat apelul şi iniţializează stiva proprie;
- Utilizînd informaţiile transmise prin registre şi zona de stivă a apelului,determină tipul primitivei care se solicită după ca-

re face verificarea parametrilor corespunzători funcţiei solicitate.

Operaţia este urmată de lansarea în execuţie a procedurii corespunzătoare;

- Restaurează stiva procesului sau monitorului din care s-a efectuat apelul după care dă controlul interfeţei de ieşire a nucleului.

3.6.3. Analiza structurilor de date utilizate de nucleu.

Principala sarcină a unui nucleu este atît de a asigura concurenţa implicată de procesele care cooperează aflate în diverse stări cît şi de a asigura necesităţile impuse de disciplinele de planificare a monitoarelor. Aceasta implică definirea de semafoare,stive,şiruri de aşteptare s.a.m.d.,care pot fi definite atît în nucleu,cît şi în procesele şi monitoarele sistemului. Principalele structuri de date utilizate,cît şi semnificaţia lor (dacă este cazul),vor fi prezentate şi analizate pe scurt în cele ce urmează.

Stive

Se rezervă una pentru fiecare proces şi o stivă pentru nucleul sistemului de operare. Este evident că stivele proceselor nu fac parte din nucleu,fiind numai apelate de acesta.

Semafoare

Se defineşte cîte un semafor pentru fiecare stare a nucleului şi a monitoarelor,cît şi pentru fiecare variabilă de condiţie utilizată. Semafoarele pot aparţine,fie direct nucleului,caz în care sînt identificate prin index,fie pot fi încorporate direct în procese,caz în care sînt identificate de nucleu prin adrese.

Blocuri de control a proceselor

La nivelul nucleului fiecare proces este identificat printr-un bloc de control al procesului (BCP) care trebuie să conţină suficiente informaţii pentru a permite punerea în aşteptare sau restartarea procesului din orice şir de aşteptare în care acesta a fost suspendat. Setul minim de informaţii,conţinute de un BCP prezentat în pseudocod este următorul:

```
Type Bloc-Control-Proces
   Record
     Numele procesului;
     Procesorul pe care se execută;
     Prioritatea de execuţie;
```

Adresa stivei proprii;
Adresa primei instrucţii;
Conţinutul registrelor atunci cînd procesul este suspendat;
end

Tabel de identificare a proceselor

Este încorporat în nucleu şi conţine informaţii care identifică ce procesoare pot executa anumite procese, care procese şi ce procesoare sînt active la un moment dat.

Siruri de aşteptare a proceselor care solicită lansarea în execuţie

Pentru fiecare procesor în parte se defineşte cîte un şir de aşteptare care va conţine procesele identificate prin adresele BCP aferente care sînt gata de execuţie. In cele ce urmează, şirurile de acest tip se vor identifica cu iniţialele GDE (Gata De Execuţie). Este evident că pentru fiecare semafor în parte se va defini de asemenea un şir de aşteptare care va conţine BCP în aşteptare pe semaforul în cauză.

3.6.4. Primitivele nucleului.

Procedurile care implementează primitivele unui nucleu reprezintă componenta structurală şi funcţională cea mai importantă a unui sistem de operare mμP. Din acest motiv analiza acestora se va face pe clase de primitive, funcţie de finalitatea funcţională şi structurală pe care o prezintă în cadrul unui nucleu. Acest punct de vedere conduce la distingerea următoarelor clase de primitive şi proceduri responsabile cu:

- controlul execuţiei unui proces;
- controlul proceselor de tip STI;
- intercomunicaţia dintre procese;
- iniţializarea nucleului;
- planificarea proceselor care se execută.

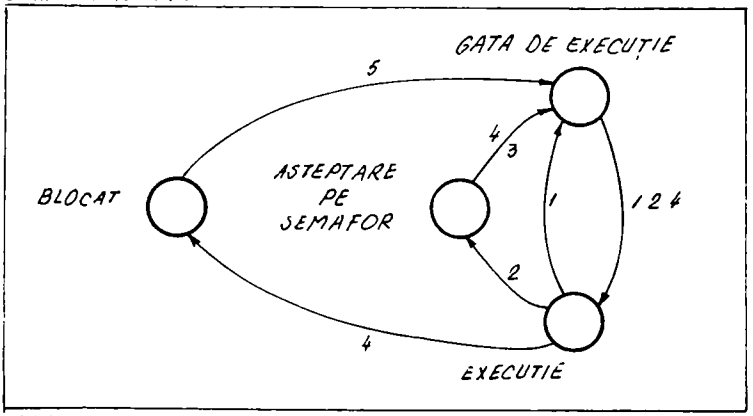
Primitive care controlează execuţia proceselor

Se definesc în urma construirii automatului cu stări finite care specifică stările nucleului şi modul în care se efectuează tranziţia dintr-o stare în alta. Automatul se construieşte plecînd de la automatul care defineşte modul de execuţie a unui proces la care se ada-

ugă un număr de stări suplimentare. Stările automatului sînt:

- EXECUTIE; reprezintă starea în care un proces este executat;
- ASTEPTARE PE SEMAFOR; un proces rămîne în aşteptare în şirul unui semafor;
- BLOCAT; procesul este blocat în interiorul unui monitor ca urmare a neîndeplinirii unor condiţii,sau a cererii efectuate explicit de un alt proces;
- GATA DE EXECUTIE; procesul îndeplineşte toate condiţiile pentru a fi lansat în execuţie;

Evenimentele care cauzează trecerea dintr-o stare în alta,vor defini setul de primitive responsabile cu stările unui nucleu şi sînt prezentate în fig.3.7.



Primitive care afectează starea nucleului

1. PREEMPT; 2. WAIT; 3. SIGNAL
4. BLOCK AND SIGNAL ; 5. UNBLOCK

Fig.3.7. Diagrama de stare a nucleului

Primitivele,rolul îndeplinit şi structurile de date afectate,sînt analizate în cele ce urmează:

WAIT (index semafor)

Variabila de condiţie ataşată semaforului este decrementată cu unu. Dacă valoarea acesteia este mai mică ca zero,procesul care a apelat primitiva este trecut în şirul de aşteptare a semaforului,după care se încearcă relansarea în execuţie a unui alt proces care aşteaptă.

SIGNAL (index semafor)

Variabila de condiţie ataşată semaforului este incrementată cu unu. Dacă valoarea acesteia este mai mică sau egală cu zero,procesul care a apelat primitiva este trecut în şirul de aşteptare GDE.

BLOCK-AND-SIGNAL (index semafor)

Primitiva realizează două funcţii; în cadrul primei funcţii,procesul care a apelat primitiva este trecut în starea BLOCAT,iar în cadrul celei de a doua,este executată o primitivă SIGNAL pe semaforul care apare în partea de argument;

WAIT-AND-SIGNAL (index semafor 1,index semafor 2). Reprezintă o variantă mai eficace a celei prezentate anterior deoarece se specifică explicit cele două semafoare pe care se efectuează funcţiile specificate anterior.

UNBLOCK (index proces)

Primitiva asigură mutarea unui proces din starea BLOCAT în şirul proceselor gata de execuţie GDE.

PREEMPT

Este utilizată de ceasul unui STI în scopul stopării execuţiei unui proces şi de a iniţia execuţia altui proces. Primitiva realizează în fapt o multiplexare a proceselor,procesul întrerupt fiind trecut în şirul GDE.

Primitive pentru controlul proceselor de tip STI

Sînt utilizate aceleaşi tipuri de primitive ca pentru procesele obişnuite,ceea ce se modifică însă este strategia de utilizare a acestora. In prezent,sînt utilizate un număr de trei strategii derivate din modul în care operaţiile de I/E sînt puse în acord cu cerinţele formulate de nucleu.

In cadrul primei strategii,procesul care solicită execuţia unei operaţii de I/E va efectua:

- o apelare a unei resurse de tip monitor pentru a cere planificarea operaţiei;
- o apelare a nucleului pentru realizarea operaţiei de I/E atunci când autorizaţia a fost dată;
- o apelare din nou a monitorului pentru a elibera echipamentul.

Strategia implică o regie mare a sistemului de operare,ceea ce are ca efect o utilizare slabă a echipamentului în cauză şi o încărcare suplimentară a procesorului.

In cadrul celei de a doua strategii,se procedează la încorporarea în nucleu a disciplinei de planificare şi a şirului de aşteptare aferent pentru fiecare tip de echipament utilizat. Metoda implică definirea unei primitive de tip "START I/O",care este apelată de procesul care solicită utilizarea unui anumit tip de echipament. Strategia prezintă avantajul că reduce regia de sistem deoarece se evită apelarea monitoarelor şi ca urmare duce la creşterea gradului de utilizare

a echipamentelor de I/E. Dezavantajul constă în faptul că se complică considerabil nucleul sistemului de operare.

In cadrul celei de a treia strategii,se definesc procese de I/E specializate pentru fiecare echipament de I/E în parte. Procesele pot cere efectuarea unor I/E prin depunerea cererilor respective cu parametrii corespunzători operaţiei solicitate într-o listă ataşată monitorului responsabil cu controlul execuţiei procesului de I/E. Acesta va relansa succesiv procesul de I/E cu parametrii specificaţi în lista de cereri,timp în care procesele care au iniţiat schimbul de informaţii cu perifericele rămîn în aşteptare pînă la terminarea operaţiilor în cauză. Avantajul acestei metode,constă în faptul că se asigură modularitatea sistemului prin definirea unor procese separate de I/E pentru fiecare echipament în parte,eliminîndu-se necesitatea interconectării între toate echipamentele şi toate procesoarele sistemului mP. De asemenea regia nucleului sistemului este mică neapărînd complicaţii în construirea lui,rezultînd în acelaşi timp şi o utilizare bună a echipamentelor în cauză.

Singurele dificultăţi majore care apar de altfel la toate cele trei metode analizate se referă la faptul că atunci cînd o întrerupere este recunoscută,STI devine o regiune critică aflată sub protecţia sistemului de întreruperi care este dezautorizat. Acest aspect face ca apelurile la nucleu efectuate de un STI să necesite un mecanism special de intrare în nucleu.

Primitive pentru realizarea intercomunicaţiei între procese

Modul de realizare a comunicaţiei dintre două procese s-a analizat într-un paragraf anterior,cînd s-a făcut şi o prezentare a primitivelor utilizate. Din acest motiv în acest paragraf prezentarea lor este făcută succint.

Primitivele utilizate sînt:

ATTACH (index şir,adresă zonă mesaj)

Are rolul de a ataşa adresa zonei de date specificată în parametrii primitivei în şirul de aşteptare a zonelor de date specificat pe ultima poziţie din şir.

DETACH (index şir,adresă zonă mesaj)

Primitiva returnează adresa zonei de date prezentă pe prima poziţie din şirul de aşteptare a cărui index este specificat de parametrii. In cazul în care şirul este liber,se returnează valoarea zero.

Primitive care realizează iniţializarea nucleului

Procedura care asigură iniţializarea sistemului de operare poate

fi amplasată în memoria unui MP destinat special operaţiei de iniţializare a sistemului,fie în memoria comună. Operaţia de iniţializare începe obligatoriu cu setarea indicatorilor de stare TAS (vezi paragraful 2.3.2.),pentru a împiedica intrarea accidentală la început în nucleu a cererilor adresate de STI.In această situaţie iniţială,intrarea în nucleu se va face evitînd secvenţa de testare a indicatorilor de stare TAS. Din acest moment operaţia de iniţializare utilizează o secvenţă de două primitive.

BEGIN

Primitiva este chemată o singură dată şi are rolul de a iniţializa structurile de date ale nucleului; de a pune pe zero variabilele de condiţie ataşate semafoarelor şi de a elibera şirurile de aşteptare ataşate semafoarelor. Primitiva poate executa (operaţia poate fi proprie fiecărui MP în parte) şi iniţializarea variabile lor şi zonelor de date proprii fiecărui MP în parte înainte ca sistemul de întreruperi să fie autorizat.

DECLARE PROCESS (index proces,index microprocesor,prioritate adresă stivă,adresă de start)

Primitiva are rolul de a iniţializa un proces şi de a crea PCB corespunzător. De asemenea în urma declarării,procesul este trecut în şirul GDE corespunzător MP pe care urmează să fie executat procesul în cauză.

Proceduri destinate planificării proceselor

Sînt utilizate pentru modificarea poziţiei unui proces aflat într-un şir de aşteptare. Deoarece apar în cadrul primitivelor definite anterior fiind incluse în structura nucleului cu toate că prezintă structura şi funcţiile unor primitive,ele apar ca proceduri ale nucleului. Procedurile utilizate sînt:

ADD (şir de aşteptare,index PCB sau adresă zonă mesaj)

Procedura are rolul de a amplasa un proces reprezentat prin indexul blocului de comandă a procesului sau o zonă mesaj reprezentată prin adresa sa în spatele şirului de aşteptare care apare ca parametru a procedurii.

REMOVE (şir de aşteptare)

Se utilizează pentru scoaterea unui element (adresă de zonă mesaj sau index proces) din faţa şirului de aşteptare care apare ca parametru.

RUN (index PCB)

Procesul a cărui index apare ca parametru este trecut într-un şir de aşteptare GDE în scopul lansării sale în execuţie.

NEXT

Procedura asigură alegerea primului proces dintr-un şir de tip GDE, conform unei anumite discipline de planificare pentru a fi lansate în execuţie.

4. CONCEPTE PRIVIND STRUCTURA UNUI SISTEM DE OPERARE MULTIMICROPROCESOR.

Atunci cînd se pune problema stabilirii unei structuri a nucleului unui sistem mμP, se pleacă întotdeauna de la structura automatului cu stări finite care descrie funcţionarea acestuia. Aceasta face ca stările de bază în care se poate găsi un nucleu să fie cele discutate în capitolul 3 şi anume: BLOCAT, ASTEPTARE, EXECUTIE, INTRERUPT, iar primitivele care asigură trecerea dintr-o stare în alta, WAIT, SIGNAL, BLOCK, UNBLOCK, PREEMPT.

Aspectele care conduc ca în cadrul acestei organizări să apară mai multe tipuri de structuri de nucleu, se referă la modul în care apar evenimentele în timp, ceea ce face predictibilă sau nu lansarea în execuţie a proceselor şi deci modul în care se face coordonarea şi sincronizarea [KN78, Mu78, RGel] . Astfel în majoritatea aplicaţiilor, apariţia unor evenimente nu este predictibilă, ceea ce implică pe de o parte utilizarea STI-lor, iar pe de altă parte forţează adoptarea unei structuri de nucleu care să conţină şiruri de aşteptare pentru procese care pot fi apelate de oricare alte procese, la momente de timp ce nu pot fi prevăzute. Cel de al doilea tip de structură rezultă în condiţiile în care apariţia în timp a evenimentelor este predictibilă ceea ce atrage după sine cunoaşterea exactă a modului în care se coordonează în timp procesele.

Cele două aspecte menţionate conduc în primul caz la structuri în care nucleele au şiruri de aşteptare, iar în cel de al doilea caz la nuclee fără şiruri de aşteptare. Din prima categorie fac parte nucleele monolitice, partiţionate şi distribuite, iar din cea de a doua nucleele de tip interogare.

4.1. Arhitectura de tip nucleu monolitic a unui sistem de operare multimicroprocesor.

Denumirea provine din cauza amplasării compacte a codului şi structurilor de date aferente nucleului. Diferitele tipuri de variante apar ca urmare a alocării sau nu a unui MP specializat în executarea codului nucleului ca şi a modului în care,dacă se alocă un MP dedicat,este încorporat în cadrul structurii sistemului mμP [EF73].

4.1.1. Nucleu monolitic localizat în memoria comună

Componentele nucleului ca şi structurile de date sînt amplasate în memoria comună a sistemului mμP,fig.4.1. In momentul în care un MP cîştigă accesul în nucleu,executarea apelului comportă realizarea următoarelor secvenţe de operaţii:

dezautorizarea întreruperilor;
testarea variabilei TAS;
salvarea registrelor în stiva procesului apelant;
comutarea pe stiva nucleului;
selectarea primitivei;
executarea operaţiei implicate în primitivă;
comutarea pe stiva procesului;
restaurarea registrelor din stiva procesului apelant;
deblocarea variabilei TAS;
autorizarea întreruperilor;

Deoarece codul nucleului este executat de MP în memoria căruia este prezent procesul apelant,este de remarcat faptul că dezautorizarea întreruperilor este obligatorie pentru a preveni fenomenul de blocare în cazul apariţiei unei întreruperi de la un STI local,în momentul în care procesorul execută operaţiile implicate de primitiva apelată. Structura adoptată [BB8o] prezintă avantajul obţinerii unui nucleu cu o lungime mică a codului,optim din punct de vedere al alcătuirii logice,ceea ce conduce la micşorarea spaţiului de memorie comună utilizată. Dezavantajul principal îl constituie prezenţa unui timp de răspuns mare a sistemului în ansamblu. Acesta apare ca urmare pe de o parte a creşterii traficului pe magistrala comună (datorat apelurilor adresate nucleului de fiecare MP în parte,ca şi a faptului că structura de date a nucleului este prezentă în totalitate în memoria comună),iar pe de altă parte a execuţiilor funcţiilor nucleului de către MP apelant a cărui procesor lucrează cu sistemul de întreruperi dezautorizat.

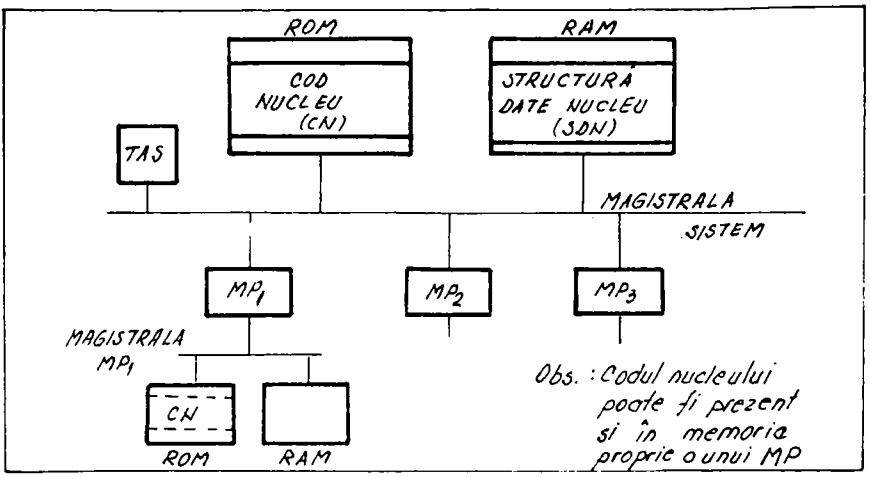


Fig.4.1. Structura sistemului cu nucleu monolitic în memoria comună.

Îmbunătăţiri ca introducerea autorizării şi dezautorizării întreruperilor în bucla de testare a stării variabilei TAS, pentru a da posibilitatea recunoaşterii evenimentelor de timp real şi de a lansa UTI-urile corespunzătoare ca şi crearea de copii a unor funcţii a nucleului în diversele memorii private ale MP-oare componente, nu conduc la o mărire corespunzătoare a timpului de răspuns care să justifice sporirea cantităţii de memorie proprie utilizată, în condiţiile în care are loc complicarea structurii logice a nucleului apărută ca urmare a divizării acestuia.

4.1.2. Nucleu monolitic de tip echipament de intrare / ieşire.

Denumirea provine din faptul că nucleul este amplasat în memoria unui MP slave care apare în contextul întregului sistem mpP ca un echipament de intrare / ieşire, fig.4.2. MP slave este protejat printr-un dispozitiv de tip TAS acelaşi care este utilizat şi pentru protecţia memoriei comune. Un aspect caracteristic îl constituie faptul că nu este prezentă o interfaţă unică de intrare / ieşire a nucleului; Transferul parametrilor către şi dinspre nucleu efectuîndu-se direct de către MP apelant prin intermediul operaţiilor de intrare / ieşire. Lipsa unei interfeţe unice obligă în schimb MP apelant să efectueze toate operaţiile legate de lucrul cu stiva pentru a salva şi restaura starea proceselor proprii. Cu alte cuvinte interfaţa de intrare/ie-

şire a nucleului este proprie fiecărui proces în parte,putînd fi distribuită fie în memoria fiecărui MP care aparţine sistemului mμP,fie amplasată în totalitate în memoria comună a sistemului mμP.

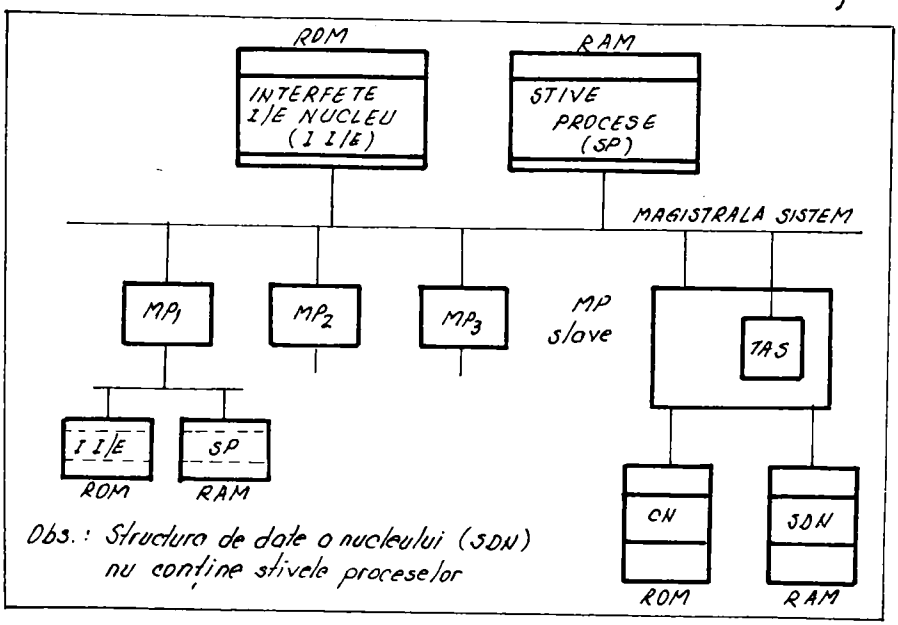


Fig.4.2. Structura sistemului cu nucleu monolitic de tip echipament de intrare/ieşire.

Ca şi în cazul structurii cu nucleu localizat în memoria comună MP care a făcut apel va utiliza metoda aşteptării active pentru a primi accesul în MP slave,respectiv în nucleu. Executarea unui apel a unei funcţii comportă realizarea următoarei secvenţe de operaţii:

dezautorizează întreruperile procesorului apelant;
test variabila TAS;
salvare registre;
scrie parametrii direct în nucleu (nu în memoria comună)
execută o buclă de aşteptare;
citeşte rezultatul din nucleu;
restaurează adresa stivei;
restaurează registrele din stivă;
resetează variabila TAS;
autorizează întreruperile procesorului apelant.

Structura prezintă avantajul unei creşteri substanţiale a vitezei prin executarea funcţiilor nucleului de un procesor specializat adresat ca un periferic de intrare / ieşire al sistemului mμP. Ca dezavantaje sînt de menţionat creşterea complexităţii dispozitivului de ar-

bitrare ca şi a faptului că scăderea traficului pe magistrala comună a sistemului este compensată în parte de contribuţia pe care o aduce la acest trafic adăugarea unui MP suplimentar. Acest ultim aspect constituie el însuşi o limitare în anumite condiţii,deoarece o serie de magistrale standard nu acceptă decît un număr limitat de MP care pot fi conectate; de exemplu,magistrala MULTIBUS permite conectarea unui număr maxim de trei MP de tip SBC 8o,în condiţiile unei arbitrări seriale [**77a,**86b] .

4.1.3. Nucleu monolitic de tip echipament de intrare/ieşire multiport.

Denumirea provine de la faptul că nucleul este localizat în memoria proprie a unui MP slave multiport [BB8o] care este privit ca un periferic de intrare/ieşire de fiecare MP al sistemului mμP,fig.4.3. Deosebirea faţă de structura prezentată anterior se referă la faptul că nucleul sistemului este localizat într-un MP slave multiport,în care fiecare MP master dispune de cîte un port de intrare şi de o linie de întrerupere proprie.

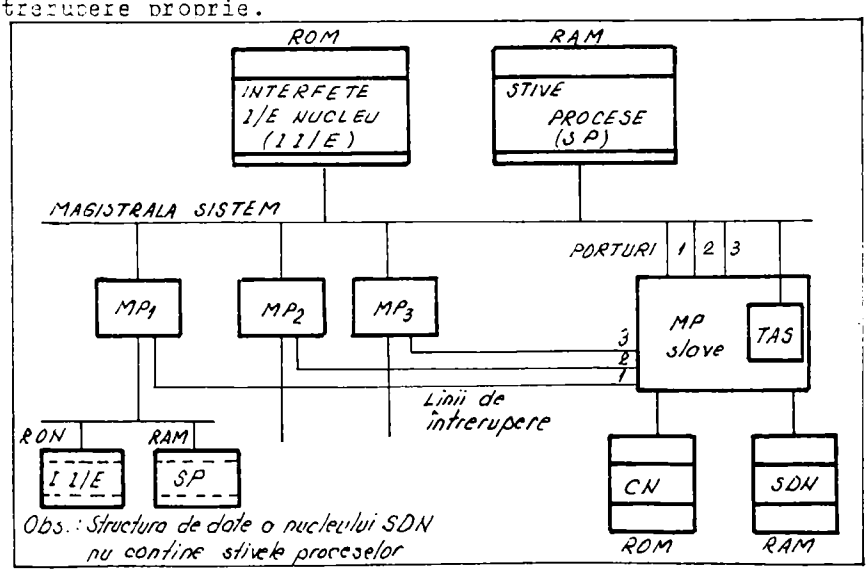


Fig.4.3. Structura sistemului cu nucleu monolitic de tip echipament de intrare/ieşire multiport.

Citind ciclic starea fiecărui port de intrare în cadrul unei discipline de planificare,nucleul poate detecta prezenţa unui apel la una din funcţiile sale. In cazul detectării unui apel se efectuează ci-

tirea parametrilor utilizînd acelaşi port de intrare după care se trece la executarea operaţiilor corespunzătoare primitivei solicitate, timp în care MP apelant rămîne într-o stare de aşteptare din care este scos printr-o întrerupere emisă de MP slave,în momentul terminării execuţiei. Din acest moment MP apelant va efectua citirea rezultatelor execuţiei realizate de nucleu şi va trece la restaurarea stării procesului. Faţă de structura precedentă apare o creştere a vitezei de răspuns a sistemului datorată scăderii traficului pe magistrala comună ca urmare a prezenţei porturilor de intrare separate pentru MP master. Dezavantajele rămîn aceleaşi (metoda aşteptării active a MP-oare la intrarea în nucleu fiind înlocuită de explorarea ciclică a porturilor de intrare),la care se adaugă o creştere a complexităţii hard prin apariţia de porturi de intrare şi linii de întrerupere suplimentare.

4.2. Arhitectura de tip nucleu partiţionat a unui sistem de operare multimicroprocesor.

Din analiza diagramei de stare a automatului care descrie funcţionarea unui nucleu,rezultă că stările po fi caracterizate prin primitive şi structuri de date care sînt proprii numai stărilor în cauză.

Această constatare a condus la definirea categoriei de nucleu partiţionat a unui sistem muP. In cadrul unei astfel de structuri automatul este fragmentat astfel încît să fie posibilă apariţia unei activităţi concurente pe fragmentele în cauză ca şi reducerea zonei de memorie comună utilizată. Caracteristica esenţială a acestei structuri este dată de extinderea mecanismului de excludere mutuală pe fragmentele nucleului,aspect care impune definirea a mai multor proceduri TAS. Realizarea acestora din urmă este efectuată în cele mai diverese moduri: utilizînd instrucţii TAS,prin instrucţii de intrare/ieşire care modelează o operaţie TAS,sau de un echipament periferic TAS [KN78,BB8c

4.2.1. Nucleu partiţionat orizontal.

Denumirea provine din cauza faptului că nucleul are codul reentrant,aspect care dă posibilitatea împărţirii concurente a acestuia de către diversele MP-oare apelante. Structura şi localizarea nucleului prezentată în fig4.4.,prezintă o serie de caracteristici. Prima se referă la faptul că regiunile critice de cod ale nucleului ca şi fi

care şir de aşteptare de tip GDE şi semafor,din cauza statutului de resursă comună sînt protejate TAS. Cea de a doua caracteristică se referă la faptul că toate informaţiile de stare referitoare la ansamblul sistemului (şirurile de aşteptare care conţin BCP la care se adaugă cîte o stivă nucleu pentru fiecare MP în parte) sînt prezente în memoria comună. In cadrul unei astfel de organizări,apelul unei primitive de exemplu SIGNAL,presupune realizarea următoarei secvenţe de operaţii,evident după ce MP apelant a cîştigat accesul pe magistrală:

```
Procedure SIGNAL (index semafor)
    Begin
      dezautorizează întreruperile procesorului apelant;
      test TAS de protecţie a semaforului;
      operează asupra semaforului pentru obţinerea adresei
          BCP a procesului care urmează să fie activat;
      resetează variabila TAS a semaforului;
      test TAS de protecţie a şirului GDE;
      mută BCP în şirul GDE;
      resetează variabila TAS a şirului GDE;
      autorizeaza întreruperile procesorului apelant;
    End
End SIGNAL
```

Avantajul metodei constă într-o creştere a vitezei de execuţie a sistemului mμP,dacă aplicaţia solicită un număr mare de apeluri a nucleului (acestea pot fi executate concurent,cu excepţia cazului în care unul sau mai multe MP-oare aşteaptă în acelaşi moment de timp pe aceiaşi variabilă TAS care protejează un semafor sau un şir GDE).Un alt avantaj se referă la faptul că nucleul care are o dimensiune mică poate fi optimizat ca urmare a utilizării reentranţei. Dezavantajul se referă pe de o parte la spaţiul mare de memorie RAM cerută,iar pe de altă parte ca o consecinţă directă apare o creştere considerabilă a traficului pe magistrala comună. De asemenea existenta unui număr mare de proceduri TAS conduce la complicarea soft sau hard a sistemului funcţie de modul de realizare a acestora. Structura se pretează pentru sistemele mμP cu două sau mai multe magistrale comune la care sînt conectate două sau mai multe memorii RAM comune.

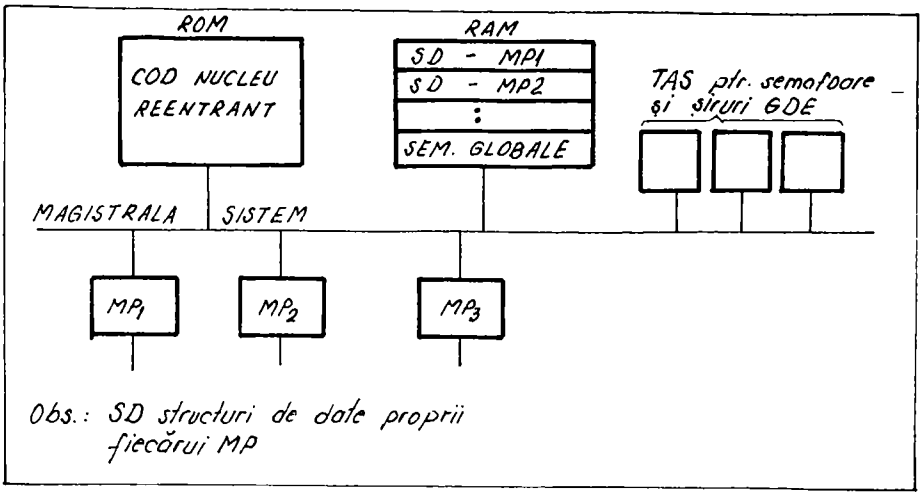


Fig.4.4. Structura sistemului cu nucleu partiţionat orizontal.

4.2.2. Nucleu partiţionat vertical.

Aspectul care a impus denumirea de nucleu structurat vertical a apărut ca urmare a reducerii la maximum a memoriei comune cerute. In ultimă instanţă s-a urmărit reducerea cererilor de acces la memoria comună ceea ce a impus reorganizarea fizică a structurilor prezente, atît ca număr de tipuri,cît şi ca volum ocupat. Operaţia este însoţită de reamplasarea codului nucleului în memoria ROM a fiecărui MP în parte [BB8o] ,fapt care face ca structura să poarte numele de nucleu partiţionat vertical,deoarece partiţionarea este făcută la nivelul structurilor de date şi nu a MP-oare.

In cadrul acestui concept,un semafor global este format dintr-o variabilă de contorizare şi un şir de aşteptare a proceselor,în care un proces se identifică nu prin adresa BCP ci prin intermediul unui identificator de proces,acesta reprezentînd o combinaţie de index procesor şi index proces,reprezentabilă pe un octet. Sirurile de aşteptare de tip GDE aferente fiecărui MP în actuala structură conţin identificatorii proceselor. Similar şirurile de aşteptare utilizate de un MP pentru a identifica zonele de date din memoria proprie sau comună este înlocuit prin şiruri de indici sau contoare. Corespondenţa dintre indexul unui proces şi BCP aferent ca şi cea dintre indexul unei zone de date şi adresa sa este realizată prin liste de corespondenţe păstrate în memoria proprie fiecărui MP. Ca şi în cazul nucleului partiţionat orizontal,protecţia semafoarelor globale ca şi a şirurilor de aşteptare GDE prezente în memoria comună se realizează prin variabile sau proceduri TAS. Deoarece în cadrul acestui concept,memoria comună

atîta cît a mai rămas,conţine numai semafoare globale şi şiruri de aşteptare în care sînt prezenţi pointeri spre informaţiile reale din memoriile proprii ale MP-oare,operaţiile de intrare/ieşire dintr-o regiune critică se modifică. Practic apare necesitatea ca şirurile GDE ca şi semafoarele aferente unui MP (din structura de date a nucleului prezent în respectivul MP) să devină accesibilă din nucleul unui alt MP al sistemului. Este evident că accesul trebuie să se facă în ambele sensuri între toate MP-oare ale sistemului. Singurele mecanisme care sînt afectate de această restructurare sînt cele corespunzătoare primitivelor SIGNAL şi WAIT,fapt care a dus la înlocuirea lor cu două noi primitive denumite CROSS-SIGNAL şi respectiv CROSS-WAIT,denumire care a fost adoptată pentru a sublinia faptul că operaţia lansată de un MP se adresează prin intermediul memoriei comune structurilor de date prezente în memoria altui MP. Datorită deosebirilor semnificative faţă de situaţia standard în continuare se dă secvenţa de operaţii realizate de fiecare din cele două primitive ale nucleului.

```
Procedure CROSS-SIGNAL (index semafor global)
    Begin
      dezautorizează întreruperile procesorului apelant;
      test TAS de protecţie a semaforului;
      operează asupra semaforului pentru obţinerea identificatorului procesului;
      resetează variabila TAS a semaforului;
      scrie identificatorul în şirul GDE cerut;
      autorizează întreruperile;
    End
End CROSS-SIGNAL
Procedure CROSS-WAIT (index semafor global)
    Begin
      dezautorizează întreruperile procesorului apelant;
      test variabilă TAS de protecţie a semaforului;
      operează asupra semaforului prin amplasarea indexului procesului în şirul de aşteptare ataşat acestuia;
      resetează variabila TAS a semaforului;
      salvează registrele; citeşte indexul noului proces din şirul GDE; restaurează registrele procesului nou şi procedează la lansarea în execuţie a acestuia;
    End
End CROSS-WAIT
```

Avantajele structurii prezentată în fig.4.5.,rezidă din dimensionarea redusă a memoriei comune,ceea ce implică un trafic redus pe magis-

trala comună. Aceasta se obţine printr-o creştere substanţială a mărimii memoriilor proprii a MP-oare ca şi printr-o complicare serioasă a structurii care se duplică în fiecare MP al sistemului.

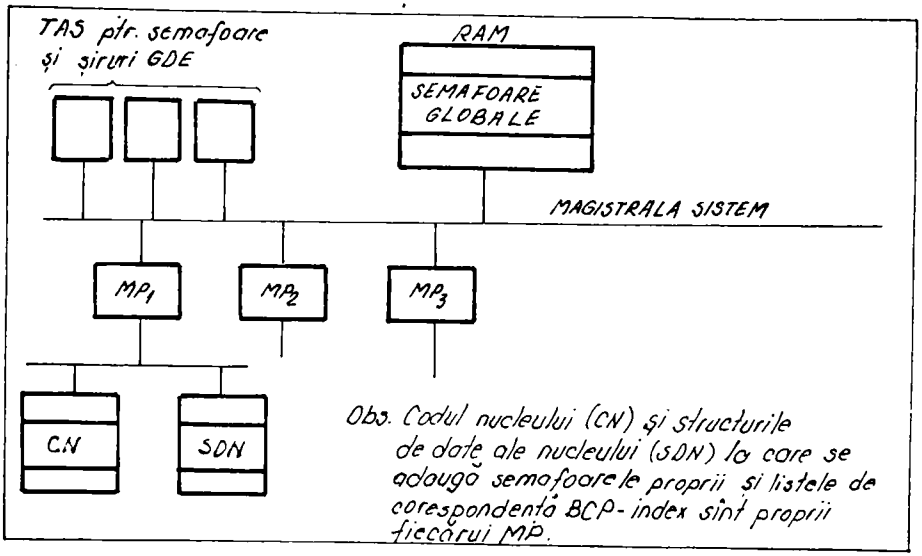


Fig.4.5. Structura sistemului cu nucleu partiţionat vertical.

4.3. Arhitectura de tip nucleu distribuit a unui sistem de operare multimicroprocesor.

Intr-o structură cu nucleu distribuit,nucleul împreună cu structurile de date aferente este localizat în totalitate în memoria proprie a fiecărui MP prezent în sistemul muP. Deoarece în acest caz nu mai sînt prezente nici măcar versiunile simplificate de structuri de date existente în memoria comună (ca în cazul structurii de nucleu partiţionat vertical),nu mai este posibil accesul bidirecţional de la un MP la alt MP,la semafoarele şi şirurile GDE prezente în fiecare din memoriile proprii în scopul sincronizării activităţilor în întregul sistem. Deoarece însă în cursul execuţiei necesitatea sincronizării face ca stările unui nucleu prezent într-un MP să fie obligatoriu afectate de modul în care se desfăşoară execuţia proceselor pe un alt MP al sistemului s-a impus schimbarea mecanismelor de comunicaţie între nucleele prezente în MP-oare,pentru asigurarea coordonării execuţiei acestora. Această schimbare a implicat modificări,în modul de realizare a primitivelor de tip WAIT şi SIGNAL şi a condus în final la două tipuri de nucleu: cu restricţii de sincronizare şi fără restricţii de sincronizare [Wi82,**86a] .

4.3.1. Nucleu distribuit cu restricţii de sincronizare.

In cadrul acestei structuri logice nucleul sistemului de operare μP este format dintr-o multitudine de nuclee standard distribuite în memoria fiecărui MP împreună cu structurile de date aferente [BB8o]. Fiecare nucleu în parte este responsabil de coordonarea şi sincronizarea execuţiilor proceselor din MP în care este prezent,gestionînd în modul cunoscut transferul unui proces din şirul de aşteptare a unui semafor,în alt şir de aşteptare sau în şirul GDF,dacă cererile de transfer sînt cerute de procesele rezidente în memoria proprie. Fiecare nucleu nu este însă o entitate singulară în cadrul sistemului μP, deoarece însăşi noţiunea de sistem implică existenţa unei comunicaţii interprocesoare în scopul sincronizării activităţilor,în care un proces în execuţie poate cere modificarea stării unui proces prezent în alt MP. Este de subliniat că în cadrul acestui tip de organizare,singurul mod de comunicare constă în activarea unui proces în urma cererii efectuate de un proces prezent în alt MP,nefiind posibil schimbul de date,deoarece memoria comună în accepţiunea adevărată a cuvîntului "nu există". Acest aspect a implicat definirea unor mecanisme speciale prin care să se realizeze tipul de comunicaţie specificat anterior, care prin implementarea sa a condus la definirea structurii de nucleu distribuit cu restricţii de sincronizare.

Ceea ce este caracteristic acestui tip de structură,fig.4.6.,îl constituie introducerea conceptului de stare a unui semafor. In cadrul acestui concept,un semafor este o entitate formată dintr-un:

-şir de aşteptare care conţine o listă înlănţuită de BCP corespunzătoare proceselor prezente în memoria proprie a fiecărui MP, şirul fiind prezent în memoria RAM a fiecărui MP;

-o variabilă de stare formată dintr-o pereche de octeţi în care fiecare octet reprezintă un contor a cărui semnificaţie depinde de poziţia octetului în cadrul variabilei.

Variabila de stare corespunzătoare fiecărui semafor utilizat de fiecare MP al sistemului sînt organizate matricial şi localizate într-o memorie comună a cărei dimensiune este neglijabilă,motiv pentru care poate fi realizată şi printr-un set de registre. Semnificaţia atribuită fiecărui octet din perechea ce defineşte variabila de stare este următoarea:

-un octet care poate lua inclusiv valori negative va indica numărul de procese posibil de a fi activate aflate în stare de aşteptare; contorul va corespunde variabilei de contorizare ataşară unui semafor standard în care prezenţa unei valori negative luată

în modul va indica numărul de procese în şirul de aşteptare;

-cel de al doilea octet care poate lua numai valori pozitive va indica numărul procesului care urmează să fie activat,apărut ca urmare a unei cereri efectuate de execuţia unui alt proces prezent în alt MP. Valoarea corespunzătoare stării "proces cu activare cerută dintr-un alt MP" este necesară deoarece valoarea negativă a contorului corespunzător furnizat de primul octet indică cîte procese sînt prezente în şirul de aşteptare corespunzător semaforului care asigură sincronizarea execuţiilor.

Acest mod de rezolvare a coordonării activităţilor realizate de procese prezente în MP diferite,cere pe de o parte definirea unei primitive prin care să se amplaseze în variabila de stare corespunzătoare semaforului global cerut a numărului procesului a cărui activare este cerută de execuţia unui proces prezent în alt MP; iar pe de altă parte,a realizării unei explorări a conţinutului respectivului octet la perioade egale de către nucleul MP,căruia îi aparţine semaforul în cauză pentru a vedea dacă în el nu este prezentă o cerere de activare pentru unul din procesele proprii.

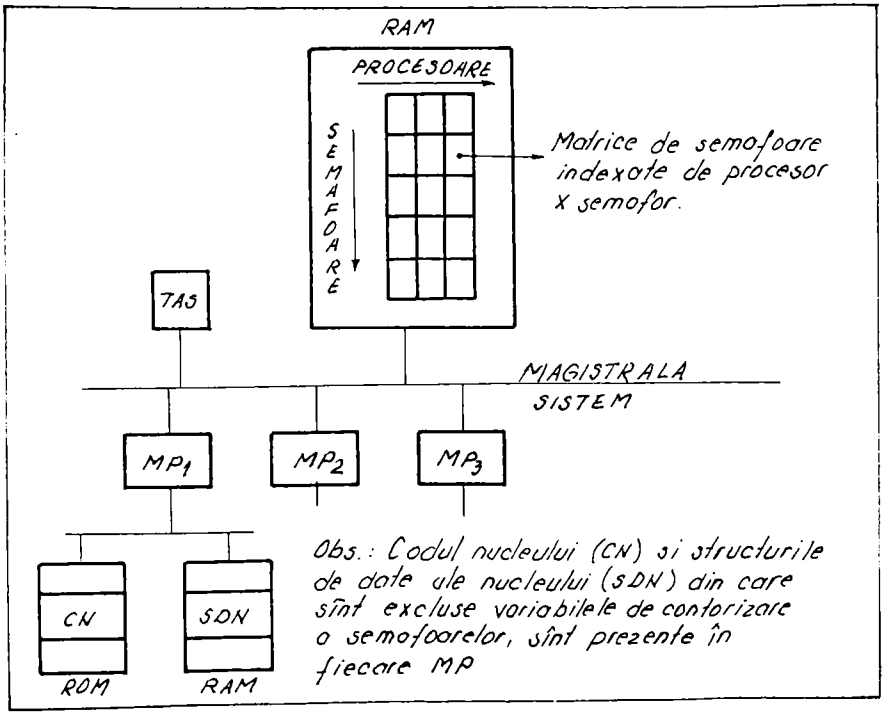


Fig.4.6. Structura sistemului cu nucleu distribuit cu restricţii de sincronizare

Operaţia de explorare se realizează de către un STI lansat de întreruperile de ceas ale MP în cauză. Datorită modului diferit de sincronizare realizat de această structură faţă de cele analizate anterior,în continuare se prezintă în detaliu în pseudocod primitive implicate în coordonarea activităţilor. In cele ce urmază se va nota cu Contor.WAIT şi respectiv Contor.WAKEUP primul şi cel de al doilea octet din variabila de stare ataşată unui semafor. Principalele primitive a căror structură şi mod de funcţionare sînt diferite de formele standard sînt:

- Primitiva WAKEUP are rolul de a testa prezenţa unei cereri de activare emisă de un alt MP fiind lansată periodic de un STI. Corespondenţa dintre numărul procesului şi BCP corespunzător se asigură prin tabele de corespondenţă prezente în memoria proprie a MP în cauză. Primitiva este de forma:

```
Procedure WAKEUP (index procesor,index semafor)
    Begin
      Dezautorizează întreruperile;
      Test variabila TAS;
    If Contor.WAKEUP > 0 then
       Begin
         Var:=Contor.WAKEUP (index procesor,index semafor);
         Contor.WAKEUP (index procesor,index semafor):=0;
         Resetează variabila TAS;
         Transferă Var în şirul GDE;
       End
    else Resetează variabila TAS;
      Autorizează întreruperile;
    End
End WAKEUP
```

- Primitivele WAIT şi SIGNAL se deosebesc de variantele standard întîlnite pînă în prezent prin aceea că operează în mod obligatoriu asupra unui semafor prezant în rudimentul de memorie comună în care sînt prezente toate semafoarele sistemului mμP,indiferent dacă se solicită activarea unui proces local MP din care s-au lansat primitivele în cauză sau se cere activarea unui proces prezent în memoria altui MP. Acest aspect impune obligatoriu efectuarea unei proceduri TAS prin care să poată fi cîştigat accesul pe magistrala sistemului mμP. După obţinerea accesului la matricea de semafoare cele două primitive vor opera în modul cunoscut asupra primului octet din variabila ataşată semaforului (respectiv asupra lui Contor.WAIT),aspect care impune ca în momentul lansării

în execuţie a primitivelor să se specifice indexul procesorului şi semaforului apelat. Ca urmare cele două primitive vor avea forma:

SIGNAL (index procesor,index semafor)

WAIT (index procesor,index semafor)

- Pentru a permite lansarea în execuţie a unui proces existent în alt MP,diferit de cel în care este prezent procesul solicitant este utilizată o primitivă de tip CROSS-SIGNAL. Aceasta are rolul de a amplasa numărul procesului a cărui lansare în execuţie se cere în octetul Contor.WAKEUP corespunzător semaforului şi MP în care este prezent procesul respectiv. Informaţia mai urmează să fie utilizată de primitiva WAKEUP prezentată mai sus. Ca şi în cazul celor două primitive anterioare se impune specificare indexului procesorului şi a semaforului apelat astfel că primitiva este de forma:

CROSS-SIGNAL (index procesor,index semafor)

Avantajul structurii se datoreşte în principal dimensiunii neglijabile a memoriei comune: aceasta putînd fi realizată cu o memorie tip RAM,sau cu un număr de registre,aspect care conduce la un trafic redus pe magistrala comună. Dezavantajele constau în restricţiile severe impuse procesului de execuţie pe ansamblul sistemului ,deoarece sincronizarea activităţilor apare ca un ansamblu de lansări în execuţie a unor procese de către alte procese (nefiind posibilă comunicarea de informaţii între acestea la nivelul diverselor MP-oare),blocarea sau întîrzierea unui proces de către alt proces în contextul în care mărimea memoriilor proprii ale MP-oare este mai mare ca urmare a duplicării nucleului în fiecare dintre acestea. Structura este acceptabilă pentru sistemele în care problemele de sincronizare nu sînt deosebit de complicate şi în care nu apar necesităţi legate de schimbul de mesaje între procesele rezidente în MP-oare diferite.

4.3.2. Nucleu distribuit fără restricţii de sincronizare.

Faţă de nucleul distribuit cu restricţii de sincronizare în care este posibilă numai activarea unui proces de către alt proces ambele fiind prezente în MP-oare diferite,printr-o mărire a zonei de memorie RAM aflată în regia fiecărui nucleu rezident este posibilă îmbunătăţirea coordonării activităţilor efectuate la nivelul întregului sistem mμP prin faptul că un proces are posibilitatea nu numai să activeze c şi să blocheze sau să întîrzie execuţia altui proces prezent în alt MP. Aceasta presupune că semafoarele globale prin care se realizează

sincronizarea şi coordonarea proceselor care cooperează să fie la fel organizate şi localizate ca în cazul precedent cu excepţia faptului că matricea de semafoare nu mai este bidimensională (indexarea efectuîndu-se după numărul procesorului şi numărul semaforului) ci monodimensională,indexarea efectuîndu-se numai după numărul semaforului. Această structură a devenit posibilă deoarece în memoria proprie a fiecărui MP se vor menţine copii după fiecare şir de aşteptare ataşat la un semafor global,fig.4.7. Astfel un proces poate cere activarea altui proces sau poate aştepta terminarea execuţiei unui proces prezent în alt MP. Ceea ce se modifică faţă de cazul precedent,constă în faptul că se adresează secvenţial toate semafoarele globale utilizate şi nu un singur semafor,caracterizat de un anumit indice procesor şi indice semafor. Prin urmare primitiva WAKEUP va trebui să se adreseze tuturor şirurilor de aşteptare,deoarece poate fi necesară coordonarea activităţii a mai mult de două procese care sînt prezente în MP diferite. In pseudocod primitiva WAKEUP care este lansată periodic de un STI de către oricare din MP-oare are următorul aspect:

```
Procedure WAKEUP
    Begin
      Dezautorizează întreruperile;
      For i:=1 to număr variabile globale;
        Test variabilă TAS;
        If Contor.WAKEUP (i)> 0 then
           Begin
           Var.=Contor.WAKEUP (i);
           Contor.WAKEUP (i):=0;
           For k:=1 to număr procesoare;
               Transferă Var în şirul GDE (k);
        End For
      else Resetează variabila TAS;
    End For
    Autorizează întreruperile;
End WAKEUP
```

O consecinţă a acestui mod de organizare este dat de faptul că disciplina şirului de aşteptare ataşat unui semafor global este impusă de necesităţile de sincronizare a activităţilor care se desfăşoară asincron în sistemul μP.

In principiu structura asigură o coordonare şi sincronizare normală a proceselor prezente în sistemul μP. Dezavantajul rezidă din prezenţa unei dimensiuni mari a memoriei proprii fiecărui MP în parte,în numărul relativ mare de cereri de acces la magistrala sistemului ca şi

din faptul că însăşi modul de funcţionare a primitivei WAKEUP introduce întîrzieri în procesul de coordonare.

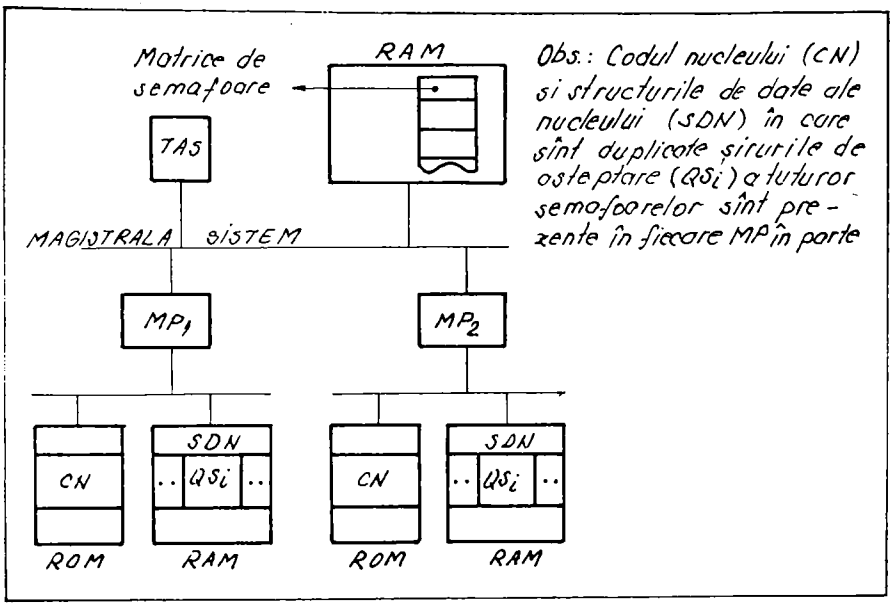


Fig.4.7. Structura sistemului cu nucleu ditribuit fără restricţii de sincronizare.

4.4. Arhitectura de tip nucleu cu interogare a unui sistem de operare multimicroprocesor.

Prezintă o alternativă a nucleului cu şiruri de aşteptare în cadrul căreia se utilizează metodă sincronizării directe. Folosirea acestei metode în care sincronizarea este realizată de relaţiile logice care apar între procesele aflate în execuţie în diversele MP-oare dă posibilitatea înlocuirii mecanismului de sincronizare în care primitivele nucleului utilizează şiruri de aşteptare printr-un mecanism de interogare a proceselor [BB8o].

Ceea ce este caracteristic acestui tip de nucleu poate fi rezumat astfel:

- Un semafor este format dintr-o pereche de variabile de contorizare cu aceiaşi structură ca cea utilizată de nucleul distribuit (pe care în cele ce urmează le vom denumi Contor.WAIT şi respectiv Contor.SIGNAL)dar cu atribuţiuni şi semnificaţii diferite ca urmare a absenţei şirurilor de aşteptare. Operaţiile legate de introducerea/scoaterea unui proces dintr-un şir de aşteptare care

care aparţine unui semafor s-au înlocuit cu operaţii efectuate asupra celor două variabile la care s-a asociat un algoritm de interogare. Semafoarele astfel definite sînt localizate în memoria comună a sistemului mμP fiind protejate fie printr-o singură variabilă TAS, aceiaşi ca cea utilizată pentru protecţia memoriei comune.

- Operaţia de semnalizare a îndeplinirii unei condiţii de către un proces se face prin incrementarea variabilei Contor.WAIT care aparţine semaforului corespunzător condiţiei de sincronizare în cauză. In pseudocod primitiva are următorul aspect:

```
Procedure SIGNAL (index semafor)
    Begin
      Dezautorizează întreruperile;
      Test variabilă TAS;
      Incrementează Contor.SIGNAL (index semafor);
      Resetează variabila TAS;
      Autorizează întreruperile;
    End
End SIGNAL
```

- Operaţia de punere în aşteptare a unui proces, de scoatere din şirul de aşteptare şi de lansare în execuţie se face prin incrementarea variabilei Contor.WAIT care aparţine semaforului pe care se efectuează operaţia de sincronizare, la care se adaugă un algoritm de interogare a proceselor susceptibile de a fi lansate în execuţie. In pseudocod primitiva WAIT şi algoritmul de interogare au următorul aspect:

```
Procedure WAIT (index semafor, tabel BCP)
    Begin
      Dezautorizează întreruperile;
      Test variabilă TAS;
      Incrementează Contor.WAIT (index semafor);
      Var:=Contor.WAIT (index semafor) —
           Contor.SIGNAL (index semafor);
      Resetează variabila TAS;
      Autorizează întreruperile;
      If Var > 0 then apelează algoritmul de interogare a
         proceselor;
                 else RETURN;
    End
End WAIT
```

- Din modul în care se efectuează incrementarea celor două contoare care formează un semafor, apare faptul că o valoare pozitivă

a diferenţei dintre cele două contoare va semnala fie disponibilitatea resursei protejată de semaforul în cauză,fie necesitatea unei operaţii de sincronizare a execuţiei a două sau mai multe procese,dacă semaforul este utilizat pentru coordonarea activităţilor. In acest caz apare necesitatea apelării unui algoritm de interogare a proceselor care îndeplinesc condiţiile de activare şi de selectare a unuia dintre acestea. Dacă valoarea diferenţei dintre cele două contoare a semaforului este negativă sau zero,procesul rămîne în aşteptare. In pseudocod,algoritmul de interogare are următorul aspect:

```
Procedure INTEROGARE (index semafor,tabelă BCP)
    Begin
      Dezautorizează întreruperile;
      Salvează Contor.WAIT şi indexul semaforului în BCP-ul
      procesului care a apelat primitiva WAIT;
    Test:
      Adresează şi încarcă BCP corespunzător indexului semafo-
      rului din tabela BCP;
      Var:=Proces.Contor.WAIT-Contor.SIGNAL
      If Var ≥ 0 then go to Test;
      Activează procesul detectat;
    End
End INTEROGARE
```

- Din modul în care se efectuează interogarea,criteriul prin care un proces rămîne în aşteptare deşi condiţiile de activare sînt îndeplinite,este dat de valoarea pozitivă a diferenţei dintre variabila de contorizare Contor.WAIT,care s-a memorat în momentul introducerii procesului în şirul de aşteptare şi valoarea curentă a variabilei Contor.SIGNAL a semaforului din momentul lansării procedurii de interogare. Acest mod de lucru presupune păstrarea în memoria proprie a fiecărui MP care aparţine sistemului mμP a unei tabele care să conţină BCP locale fiecărui MP în parte. In cadrul acestor tabele,BCP formează de obicei liste înlănţuite pentru fiecare semafor utilizat. Se observă din modul de lansare în execuţie,că primul proces care îndeplineşte condiţiile de lansare în execuţie reprezintă şi primul prezent în "şirul de aşteptare",aspect care în anumite situaţii reprezintă o restricţie care impune cunoaşterea modului în care se coordonează în timp activităţile declanşate de procese şi care în multe situaţii impune complectarea în mod corespunzător a algoritmului de interogare cu

diverse discipline de planificare a lansării în execuţie a proceselor.

In concluzie,în cadrul nucleelor cu interogare,introducerea în şirurile de aşteptare asociat cu primitiva WAIT şi scoaterea din şirul unui semafor a unui proces,urmată de introducerea lui în şirul proceselor GDF asociată cu primitiva SIGNAL,s-a înlocuit printr-un algoritm de interogare şi printr-o utilizare particulară a celor două variabile care alcătuiesc un semafor. Eficienţa nucleului depinde de eficienţa algoritmului de interogare şi de măsura în care este necesară cunoaşterea exactă a coordonării activităţilor în timp în sistemul mμP. Faptul că nucleul nu necesită structuri de date complicate în memoria comună şi că permite existenţa proceselor de tip STI care pot fi lansate şi executate în intervale scurte de timp ca urmare a faptului că execuţia primitivelor nucleului nu reprezintă un factor consumator de timp,face ca acest tip de structură să fie eficientă. Dezavantajul rezidă din faptul că în cazul unui număr mare de procese,algoritmul de interogare devine o componentă consumatoare de timp ca şi faptul că din motive de eficienţă este distribuit în fiecare memorie proprie a MP-oare componente ale sistemului mμP,aspect care duce la o creştere a cantităţii de memorie utilizată de fiecare MP în parte.

5. MODELE ALE GESTIUNII RESURSELOR INTR-UN SISTEM MULTIMICROPROCESOR.

5.1. Tehnici de analiză.

Gestiunea resurselor unui sistem mμP reprezintă un fenomen deosebit de complex,în care interfaţa dintre resursele hard şi soft este puternic influenţată de aplicaţia particulară căreia îi este dedicat sistemul. Acest aspect a condus la apariţia unui număr mare de tehnici de analiză,cu un caracter mai mult sau mai puţin general,în care sînt utilizate atît metode numerice cît şi de simulare. In cele ce urmează,se va proceda la o trecere în revistă a acestora,cu scopul definirii caracteristicilor de bază ca şi a domeniului de aplicabilitate.

O primă caracteristică comună tuturor acestor tehnici se referă la faptul că analiza este făcută nu prin urmărirea comportării unei anumite componente hard sau soft sau a interacţiunii dintre acestea în timp,ci prin urmărirea modului în care aceste componente intră în conflict. Acest aspect,a deplasat domeniul de analiză numai asupra acelor componente care reprezintă o sursă de conflict în desfăşurarea activităţilor într-un sistem mμP. Mai concret,alocarea unei resurse soft este analizată din punct de vedere a conflictelor care apar la alocarea memoriei comune,iar alocarea unei resurse hard este analizată din punct de vedere a conflictelor care apar la alocarea magistralelor sistemului.

O a doua caracteristică se referă la faptul că în majoritatea tehnicilor utilizate,aplicaţia căreia îi este destinat sistemul mμP este caracterizată de una sau mai multe variabile aleatoare cu diverse distribuţii sau valori medii. Datorită complexităţii fenomenului fenomenului de gestiune,în foarte puţine cazuri se pleacă de la un graf de proces în care apare explicit modul în care este necesară sincronizarea diverselor procese în timpul execuţiei,durata execuţiei fiecăruia în parte ca şi momentul în care este necesară lansarea lor în execuţie. Deşi acest mod de reprezentare este cel mai apropiat de realitate,rareori este posibilă obţinerea mărimilor mai sus menţionate, fapt care face ca aplicaţia şi întregul sistem de operare responsabil de coordonarea activităţilor în timp să fie definit prin variabile aleatoare reprezentînd intervale de timp,timpi de execuţie,lungime de mesaje,legate fiecare dintre ele de cererile de acces la resursele comune.

O a treia caracteristică,reprezintă un rezultat al primelor două prezentate anterior. Astfel gestiunea resurselor în sistemul mμP,este privită numai din punctul de vedere a conflictelor ce apar ca urmare a cererilor care se adresează memoriei comune şi alocării magistralelor. Ca urmare,rezultatele care se obţin în urma analizelor efectuate se vor referi la indici de performanţă ca de exemplu: capacitatea de trecere a magistralelor sistemului,timpul mediu de aşteptare a unei cereri,numărul mediu de cereri în aşteptarea alocării resursei. Foarte multe din analizele efectuate,urmăresc determinarea unui optim,care se referă de obicei la numărul de MP-oare care pot fi conectate în contextul prezenţei unui anumit număr de magistrale şi blocuri de memorie comune. Dintre tehnicile generale de analiză,în prezent,se pot distinge un număr de patru modalităţi de abordare. Prima se referă la definirea unor modele analitice a gestiunii resurselor comune care fac apel la elemente din teoria şirurilor de aşteptare. Aceste modele care au fost utilizate printre primii de Kleinrock şi Fung [Kl75,Ft79], pleacă de la graful de proces al aplicaţiei,în care fiecare proces este caracterizat de numărul de instrucţii care fac referiri la memoria proprie MP în care este rezident procesul şi numărul de instrucţii care fac referiri la memoria comună. Ca urmare este posibilă,ţinînd cont de interacţiunea în timp a proceselor ca şi de localizarea acestora,să se determine numărul de cereri care se adresează în fiecare ciclu procesor memoriei comune. In final se obţin relaţii analitice care permit determinarea capacităţii de trecere a sistemului sau a numărului optim de blocuri în care trebuie împărţită memoria comună [MH86a,MH87b]. Tehnica conduce la relaţii finale simple de calcul care însă în cazul în care se renunţă la definirea grafului de proces şi la înlocuirea lui printr-o variabilă aleatoare cu o valoare medie şi distribuţie dată (rezultate din analiza globală a proceselor care se execută în întregul sistem mμP),conduce la apariţia de erori care pot face neutilizabile rezultatele obţinute. Pe de altă parte definirea unui graf de proces reprezintă o operaţie laborioasă,care în cazul unor aplicaţii complexe devine practic imposibilă. Cea de a doua tehnică,se referă la utilizarea lanţurilor Markov în analiza conflictelor apărut la alocarea magistralelor şi memoriei comune. Modelele în care se utilizează această tehnică sînt deosebit de numeroase,pe de o parte datorită faptului că procesele Markov dau posibilitatea introducerii implicite a factorului timp şi a descrierii structurii sistemului (prin definirea matricii probabilităţilor de tranziţie dintr-o stare în alta a sistemului) ca şi descrierea modului de interacţiune a proceselor la alocarea resurselor comune.

Ca urmare a acestui aspect,pe de altă parte,este posibilă caracterizarea ansamblului de procese care se execută prin una sau mai multe variabile aleatoare,cu o anumită valoare medie şi distribuţie. Deoarece însă acest tip de analiză solicită definirea stărilor în care poate fi prezent un ansamblu de procese în execuţie,în contextul unui hard dat şi a probabilităţilor de tranziţie dintr-o stare în alta,tehnica lanţurilor Markov s-a utilizat cu precădere în analiza performanţelor unor sisteme deja existente,pentru care a fost posibilă obţinerea acestor date. Exemple clasice de utilizare a acestei tehnici sînt cele prezentate de Benjamin Wah şi Briggs în [Be83,BD81] . Cu toate dificultăţile legate de definirea matricii probabilităţilor de tranziţie,care constituie în multe cazuri un obstacol greu de depăşit,sînt prezente încercări de definire a unor modele Markov generale şi nu dedicate unei anumite aplicaţii. Dintre ultimele sînt de amintit cele prezentate în [AG82,To86] .Cea de a treia tehnică,se referă la utilizarea reţelelor cu şiruri de aşteptare,care a cunoscut în ultima perioadă o dezvoltare deosebită ca urmare a facilităţilor pe care le pune la dispoziţie. Metoda cere definirea unei matrici a probabilităţilor de tranziţie,dar nu între stările sistemului ca în cazul proceselor Markov,ci între diversele componente hard sau soft a sistemului mµP. Ca urmare,este posibilă pe de o parte definirea modului în care se desfăşoară în timp execuţia proceselor,iar pe de altă parte,definirea modului de interacţiune a proceselor la alocarea resurselor comune este legat de o anumită structură hard/soft a sistemului. Acest ultim aspect nu apare explicit în cazul lanţurilor Mrkov. Ca şi în cazul acestora din urmă,tehnica s-a aplicat îndeosebi pentru construirea unor modele pentru sisteme de calcul existente,în care a fost posibilă definirea matricii probabilităţilor de tranziţie între componentele sistemului ca şi calculul timpului mediu de execuţie şi a distribuţiei acestuia în fiecare centru de servire a reţelei. Datorită informaţiilor suplimentare pe care analiza cu reţele de şiruri de aşteptare le pune la dispoziţie,ca de exemplu: timp mediu de aşteptare,grad de utilizare,lungime medie a şirurilor de aşteptare, s.a.m.d.,studiile de gestiune a resurselor în care această tehnică este utilizată sînt deosebit de numeroase. Contribuţii importante atît la definirea unor tehnici de calcul a indicilor de performanţă a unui model descris cu reţele cu şiruri de aşteptare, cît şi în aplicarea acestui mod de analiză la studiul gestiunii resurselor în sistemele de calcul le-au adus Chandy K.M. în [CH75a,CH75b] ,Sauer C. şi Reiser M. în [Re81,SC81] ,Buzen J. în [Bu73] ca şi Lavenberg S. în [La83] . Cu toate facilităţile enumerate mai sus,tehnica utilizării reţelelor

cu şiruri de aşteptare la analiza gestiunii resurselor în sistemele mμP este limitată pe de o parte la construirea unor modele numai pentru sisteme deja existente,pentru care pot fi culese datele de intrare necesare procesului de modelare,nefiind posibilă definirea unor modele cu caracter general. Pe de altă parte,tehnica nu este aplicabilă decît pentru cazul în care timpul de serviciu în centrele de servire a reţelei este dat de o variabilă cu distribuţie exponenţială,aspect care limitează serios domeniul de aplicabilitate. Este de remarcat de asemenea faptul că algoritmii de calcul sînt complicaţi,fiind dificil de aplicat în cazul unor modele descrise prin reţele cu multe centre de servire. Cea de a patra tehnică de analiză,se referă la construirea unor modele de simulare a gestiunii resurselor într-un sistem mμP. Tehnica are un caracter de aplicabilitate foarte general,fiind posibilă descrierea exactă a procesului de coordonare a execuţiei proceselor dintr-un sistem mμP de către un anumit tip de sistem de operare,în contextul unei structuri hard date. In cadrul acestei tehnici,modelul care se construieşte,reproduce funcţiile unui sistem care urmează să fie construit sau care este deja prezent,utilizînd ca date de intrare atît mărimi care caracterizează sistemul mμP din punct de vedere a structurii hard,cît şi un set de variabile aleatoare a căror semnificaţie,distribuţie şi valor medii sînt impuse de aplicaţia căreia îi este destinat sistemul. Rezultatele obţinute în urma simulării sînt dintre cele mai diverse înglobînd practic totalitatea indicilor care pot caracteriza comportarea unui sistem mμP. Descrierea modelelor poate fi făcută utilizînd limbaje specializate de nivel înalt ca de exemplu,GPSS [Go76,DO66] sau limbaje cu caracter general ca de exemplu Pascal concurent [Vac6] . Ca şi în cazul altor tehnici,nici construirea modelelor de simulare nu conduce la definirea unor modele cu caracter general,tehnica fiind îndeosebi utilizată pentru studii de îmbunătăţire a performanţelor unor sisteme deja existente [CM81,Sc81,JL81] . Utilizarea ei,la construirea unor modele pentru sisteme care urmează să fie realizate,trebuie făcută cu multă atenţie,deoarece se elimină etapa de validare a rezultatelor obţinute prin simulare,aspect care în multe cazuri face imposibilă utilizarea tehnicii în cauză. Aceasta este motivul pentru care construirea modelelor de simulare se utilizează în general împreună cu tehnicile prezentate anterior în scopul validării rezultatelor obţinute de acestea. In plus,tehnica prezintă dezavantajul unei complexităţi relativ mari a descrierii modelului printr-un limbaj specializat,în contextul unui timp mare de simulare, ceea ce în multe cazuri constituie un impediment care limitează utilizarea tehnicii în cauză.

Din cele prezentate anterior,rezultă că la analiza procesului de gestionare a resurselor dintr-un sistem μP,sînt utilizate o multitudine de tehnici,a căror complexitate este mai mare sau mai mică,funcţie de nivelul de detaliere a anlizei care se efectuează,ca şi de scopul urmărit:proiectare sau îmbunătăţirea performanţelor unui sistem deja prezent. Acest număr mare de tehnici,este pe de altă parte o consecinţă firească a complexităţii fenomenului de gestiune a resurselor soft/hard dintr-un sistem μP care reprezintă practic un unicat pentru fiecare aplicaţie căreia îi este destinat sistemul în cauză.

5.2. Modelul gestiunii resurselor într-un sistem multimicroprocesor.

5.2.1. Formularea problemei.

Din analiza tehnicilor utilizate în prezent la modelarea fenomenului de gestiune a resurselor în sistemele μP se desprind un număr de trei concluzii şi anume:

- In marea majoritate a cazurilor,tehnicile de analiză urmăresc construirea unor modele din care să rezulte fie capacitatea de trecere a sistemului,fie determinarea numărului optim de magistrale,blocuri de memorie şi MP-oare.
- Sistemul μP este privit ca un ansamblu în care fenomenul de gestiune este văzut ca un conflict apărut ca urmare a cererilor adresate memoriei comune condiţionat de existenţa unui anumit număr de magistrale.
- Complexitatea calculelor impuse de modelele analitice ca şi de definirea unor modele de simulare este relativ ridicată,în contextul în care rezultatele obţinute au un grad de incertitudine destul de mare.

Cele trei concluzii,în contextul analizei efectuate anterior scot în evidenţă o serie de lipsuri prezente în modalităţile de analiză a fenomenului de gestiune a resurselor în sistemele μP existente pînă în prezent. Acestea s-ar putea rezuma în cele ce urmează:

- Determinarea numărului optim de MP-oare pentru o aplicaţie dată este cu mici excepţii inutilă,deoarece numărul de MP-oare este impus de aplicaţie,mai ales în cazurile în care sistemul μP este destinat unor aplicaţii de tip timp-real.
- Este inutilă determinarea unui graf de proces,în care pentru fiecare proces să se calculeze timpul de execuţie şi momentul în care acesta urmează să fie executat după cum se procedează în

[Hi81,Fu79] ,deoarece aceste mărimi sînt rezultate din execuţia proceselor în cauză,în contextul unei anumite situaţii. Ca urmare valoarea lor numerică se modifică de la o execuţie la altă execuţie,aspect care impune caracterizarea prin valori medii a unor distribuţii de valori,ca fiind mult mai apropiată de realitate decît în cazul caracterizării aceloraşi mărimi prin nişte valori fixe.

- In nici unul din modelele utilizate nu apare aportul adus la operaţia de gestiune a resurselor de sistemul de operare,deşi aceasta are o contribuţie esenţială în coordonarea execuţiei proceselor în sistemul mµP. Incorporarea acestei contribuţii în valorile medii a variabilelor aleatoare ce definesc ansamblul proceselor care se execută, constituie o aproximaţie foarte generală care nu ţine seama de fapt, de structura sistemului de operare.

Luînd în considerare cele menţionate,rezultă necesitatea remedierii acestei deficienţe prin definirea unui model care să ia în considerare şi caracteristicile de bază ale sistemului de operare care este implementat şi să arate că prin implementarea unui anumit tip sau altul de sistem de operare,performanţele în ansamblu ale sistemului mµP se modifică. Prin definirea acestui model se urmăreşte de asemenea stabilirea unui criteriu de alegere,funcţie de tipul aplicaţiei a celui mai potrivit sistem de operare.

5.2.2. Definirea modelului.

Stabilirea caracteristicilor modelului a impus rezolvarea unui număr de trei probleme privind: arhitectura pentru care se construieşte modelul,tehnica de modelare care se utilizează şi modul în care se reflectă în cadrul acestei tehnici structura sistemului de operare.

Din punct de vedere constructiv se consideră că sistemul multimicroprocesor este format din N procesoare $MP_1, MP_2, .. MP_N$,fiecare avînd propria sa memorie. Fiecare procesor este conectat la o memorie comună printr-un număr de B magistrale notate cu $B_1, B_2, .. B_B$. Memoria comună constă din M module de memorie notate cu $M_1, M_2, .. M_M$,pentru a permite cererilor simultane de memorie accesul concurent la memoria comună. Fiecare procesor este conectat la fiecare magistrală (fig.5.1.) şi prin intermediul acestora la fiecare modul al memoriei comune [MH87b] . Se consideră că magistralele de date şi adrese operează asincron,aspect care permite ca operaţia de arbitrare a cererilor adresate memoriei comune ca şi a cererilor de ocupare a unei magistrale să se efectueze în paralel cu transferul datelor. In cadrul arhitecturii luate în considerare pot apare două tipuri de conflicte da-

torate cererilor adresate memoriei:

- Primul apare ca urmare a faptului că numai o cerere poate fi adresată la un moment dat unui acelaşi bloc de memorie;
- Cel de al doilea,apare ca urmare a faptului că este posibilă depăşirea capacităţii magistralelor sistemului.

Cele două tipuri de conflicte impun modelului un număr de două restricţii:

C1) Un MP nu reprezintă o sursă a mai mult de un mesaj la un moment dat;

C2) Un bloc a memoriei comune nu reprezintă destinaţia decît a unui singur mesaj în curs de transmisie la un moment dat.

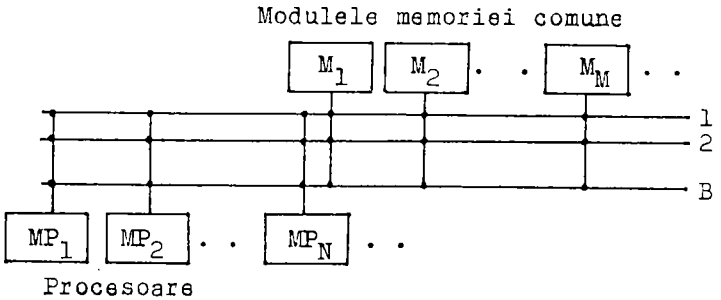


Fig. 5.1. Arhitectura sistemului multimicroprocesor.

Din punct de vedere al tehnicii de construire a modelului s-a adoptat tehnica lanţurilor Markov. Această alegere s-a datorat faptului că procesele Markov dau posibilitatea introducerii implicite a factorului timp,ca şi a descrierii structurii sistemului (prin definirea matricii probabilităţilor de tranziţie dintr-o stare în alta a sistemului),ca şi a descrierii implicite a modului de interacţiune a proceselor la resursele comune (prin definirea stărilor sistemului).

Din punct de vedere al categoriilor de sisteme de operare utilizate în cadrul sistemelor de operare μP în prezent se cunosc două mari categorii [BB8o] . Prima categorie cuprinde sistemele de operare a căror nucleu realizează sincronizarea şi coordonarea utilizînd tehnica semafoarelor,categorie din care fac parte nucleele distribuite şi de tip interogare. Cea de-a doua categorie cuprinde acele sisteme de operare în care sincronizarea este realizată printr-o tehnică de schimb de mesaje,categorie din care fac parte nucleele monolitice şi partiţionate. Evident că delimitarea dintre cele două tipuri nu este riguroasă,în realitate cele două tipuri de sincronizare şi coordonare fiind prezente în anumite cazuri concomitent.

Pentru a caracteriza cele două categorii de sisteme de operare, s-a făcut observaţia că în cadrul sistemelor care utilizează tehnica semafoarelor, cererile de acces la memoria comună, respectiv la nucleul sistemului de operare, sînt formate din mesaje care au o lungime fixă. In schimb, în sistemele în care se utilizează o tehnică de sincronizare bazată pe mesaje, cererile de acces sînt formate din mesaje a căror lungime este variabilă, putînd fi reprezentate printr-o variabilă aleatoare cu distribuţie exponenţială.

In literatură, pînă în prezent s-au dezvoltat numai modele pentru sistemele μP care răspund numai la primele două probleme. Plecînd de la un astfel de model de tip P*M*B dezvoltat în [To86, CG87], şi completîndu-l pentru a răspunde şi la cererile impuse de a treia problemă, se propune un model de tip P/D*M*B pentru definirea gestiunii resurselor în sistemele μP în care coordonarea şi sincronizarea se face utilizînd tehnica semafoarelor şi un model de tip P/E*M*B pentru definirea gestiunii resurselor în sistemele μP în care se utilizează tehnica mesajelor pentru realizarea operaţiilor de sincronizare şi coordonare a activităţilor.

5.3. Model de tip P/D*M*B

In cadrul acestui tip de model [Ro87] considerăm că operaţiile de coordonare şi sincronizare a activităţilor în sistemul μP impun sistemului de operare operaţii de transfer pe magistralele sistemului, în care mesajele au o lungime constantă. Acest aspect este subliniat prin notaţia /D, în care D specifică caracteristica de constanţă a mesajului care se transmite, notaţia fiind identică cu cea utilizată în teoria aşteptării [MC73, Al75]. In general utilizarea mesajelor de lungime constantă este caracteristică sistemelor de operare μP cu restricţii de sincronizare, categorie care înglobează acele sisteme de operare care implementează numai funcţiile de coordonare şi sincronizare a proceselor prin intermediul semafoarelor. Astfel, un proces poate cere lansarea în execuţie sau blocarea altui proces, fără a exista posibilitatea ca acestea să-şi transmită reciproc mesaje. Exemple din această categorie sînt date de sistemele de operare μP distribuite, cu interogare şi în general de toate celelalte tipuri, dacă în cadrul acestora operaţiile de cooperare nu implică mecanisme în care criteriul de coordonare şi sincronizare este dat de schimbul de mesaje dintre procese.

5.3.1. Indici de performanţă.

In cazul general,operaţiile de transfer pe magistralele sistemului μP a cărui arhitectură s-a prezentat în fig.5.1.,se desfăşoară asincron. Prin aceasta se va înţelege că operaţiile de alocare a resurselor comune,blocuri de memorie şi magistrale,se efectuează secvenţial în fiecare moment în care una dintre aceste resurse devine disponibilă,în condiţiile impuse de cele două restricţii specificate anterior C1 şi C2. Deoarece definirea unui model analitic asincron este dificilă de realizat,în cele ce urmează se va utiliza un model sincron,considerînd că operaţiile de alocare a cererilor au loc la începutul unui interval de lungime fixă. Pentru a obţine o soluţie viabilă a comportării asincrone a sistemului real,prin utilizarea unui model sincron este necesar ca mărimile cu care se operează să ţină cont de acest aspect. Din acest motiv,mărimile caracteristice modelului utilizat în calculul indicilor de performanţă se definesc după cum urmează:

D.5.3.1. Timpul mediu de aşteptare (W) este intervalul de timp dintre momentul generării mesajului şi momentul în care transmisia sa este complectă.

D.5.3.2. Intervalul de timp (T) la începutul căruia se efectuează operaţiile de alocare a resurselor este egal cu durata de transmisie a unui mesaj.

D.5.3.3. Frecvenţa de generare a mesajelor (λ) este raportul dintre unitate şi durata intervalului de timp cuprins între momentul terminării transmisiei mesajului şi momentul în care s-a generat următorul mesaj.

D.5.3.4. Timpul mediu de răspuns a unui modul procesor (R) este intervalul de timp între două generări consecutive de cereri spre un bloc de memorie.

Utilizînd mărimile definite anterior,rezultă imediat că timpul mediu de răspuns (ciclul unui MP) este format din timpul mediu de aşteptare şi timpul mediu necesar generării următoarei cereri,fiind deci de forma:

$$R = W + 1/\lambda \qquad (5.1)$$

Prezenţa timpului de aşteptare în cadrul ciclului unui MP,care înglobează timpii de aşteptare impuşi de fenomenul de alocare a unei magistrale sau bloc de memorie,face ca numărul mediu real de cereri efectuate şirezolvate de un MP la unul din blocurile memoriei comune ale

sistemului mμP să fie dat de raportul $1/(W + 1/\lambda)$. Ca urmare, rezultă imediat că numărul mediu de cereri care sînt efectuate şi rezolvate pe ansamblul întregului sistem mμP este dat de raportul $N/(W + 1/\lambda)$, în care N specifică numărul de MP-oare ale sistemului. Dacă reprezentăm activităţile din sistemul mμP impuse de fenomenul de gestiune a resurselor printr-o reţea cu şiruri de aşteptare, atunci aceasta are aspectul din fig.5.2.

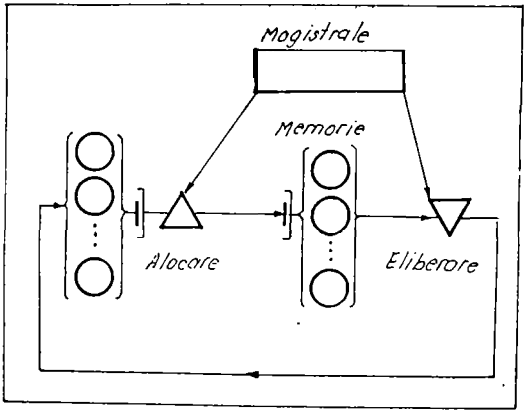


Fig.5.2. Modelul cu reţele de şiruri de aşteptare a gestiunii resurselor dintr-un sistem multimicroprocesor.

Deoarece în regim staţionar numărul de cereri care se transmit poate fi considerat constant, este posibilă definirea capacităţii de trecere a sistemului mμP.

D.5.3.5. Capacitatea de trecere a unui sistem mμP (Λ) este dată de numărul de cereri adresate blocurilor de memorie comună, care se pot transmite în unitatea de timp.

Utilizînd mărimile definite anterior, rezultă pentru capacitatea de trecere următoarea relaţie:

$$\Lambda = N/(W + 1/\lambda) \qquad (5.2)$$

de unde pentru timpul mediu de aşteptare se obţine:

$$W = N/\Lambda - 1/\lambda \qquad (5.3)$$

Pentru a putea calcula timpul mediu de aşteptare, devine necesar în acest context definirea unei proceduri care să conducă la obţinerea capacităţii de trecere a sistemului mμP.

Calculul capacităţii de trecere

Se face, considerînd că cererile efectuate de un MP al sistemului adresate memoriei comune, pot fi modelate ca o secvenţă de experienţe Bernoulli [MH78a, La83]. Pentru aceasta se consideră că intervalul pentru care se face studiul indicilor de performanţă este împărţit în cuante de timp a căror lungime este egală cu T, respectiv cu durata de transmisie a unui mesaj, fig.5.3.

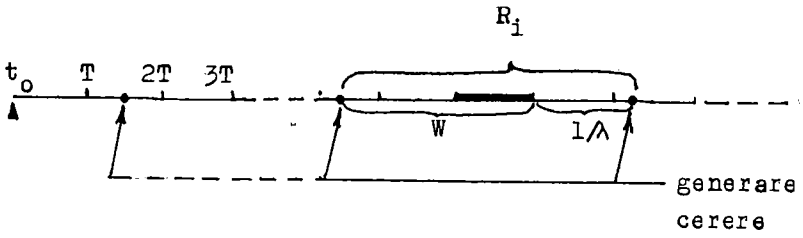


Fig. 5.3. Generarea cererilor efectuate de modulul procesor cu numărul i.

Dacă frecvenţa de generare a unei cereri de transmisie a unui mesaj efectuată de către un MP este egală cu λ cereri în unitatea de timp, atunci probabilitatea de apariţie a unei cereri în intervalul T este de $1 - e^{-\lambda T}$ [K78] . In această ipoteză, rezultă imediat că probabilitatea ca la începutul unui interval de lungime T să avem cereri prezente de la unul din cele N module procesoare ale sistemului mμP, este dată de relaţia $N(1 - e^{-\lambda T})$.

D.5.3.6. In regim staţionar, p_n este probabilitatea ca în sistemul mμP să fie prezente un n cereri cu $n = \overline{0,N}$ în aşteptarea rezolvării , la începutul unui interval de lungime T.

Din modul în care s-a definit p_n, rezultă că numărul de cereri n încorporează atît cererile de transmisie a mesajelor care nu s-au putut rezolva în intervalul precedent datorită conflictelor de alocare a resurselor comune, cît şi cele care s-au generat şi au apărut pentru prima dată în sistem, urmînd să concureze în prezentul interval la alocarea unui bloc de memorie. Utilizînd p_n, $n = \overline{0,N}$, se poate calcula numărul mediu de cereri (L) prezente la începutul unui interval de lungime T. Dacă considerăm numărul de cereri prezente în sistem la începutul fiecărui interval de lungime T ca fiind realizările unei variabile aleatoare discrete X (număr de cereri) atunci L reprezintă valoarea sa medie şi este dată de relaţia:

$$L = \sum_{n=1}^{N} np_n \qquad (5.4)$$

Cu alte cuvinte L reprezintă numărul mediu de MP-oare din sistem a căror cereri nu s-au rezolvat, fie datorită faptului că sînt nou apărute, fie din cauză că resursa solicitată nu este disponibilă. In aceste condiţii probabilitatea de a avea cereri de la cele L module procesoare ale sistemului este dată de $L(1 - e^{-\lambda T})$. Diferenţa dintre cererile care pot fi generateîn cursul unui interval T şi cererile pre-

zente la sfîrşitul aceluiaşi interval T care nu au fost rezolvate, reprezintă numărul de cereri care pot fi transmise şi ca urmare deci, reprezintă capacitatea de trecere a sistemului

$$\Lambda = N(1 - e^{-\lambda T}) - L(1 - e^{-\lambda T})$$

de unde

$$\Lambda = (N - L)(1 - e^{-\lambda T}) \qquad (5.5)$$

5.3.2. Structura modelului.

Plecînd de la definiţiile făcute anterior, se poate construi un model Markov observînd starea sistemului mμP la începutul unui interval de timp T. Modelul este definit de un vector de forma $(n_0, n_1, .. n_N)$ în care o stare n_i este dată de numărul cererilor prezente în sistem în aşteptare la începutul unui interval de lungime T. Astfel numărul de stări devine egal cu N + 1, număr rezultat din faptul că la începutul unui interval de lungime T în sistemul mμP pot fi prezente o, 1, 2, .. N cereri adresate blocurilor memoriei comune.

Calculul probabilităţilor de tranziţie

Pentru a calcula probabilitatea de tranziţie dintr-o stare în alta a modelului Markov fără a cunoaşte exact care este MP care emite cererea şi către ce bloc de memorie se adresează cererea în cauză se face presupunerea [Ko78] că cererile pot proveni cu egală probabilitate de la oricare din cele N module procesoare şi se pot adresa cu egală probabilitate oricăruia din cele M blocuri de memorie comună. Cu alte cuvinte se face presupunerea că distribuţia sursei şi destinaţiei unui mesaj la începutul unui interval T este dată de o distribuţie uniformă.

Pentru a obţine elementele matricii [Q] a probabilităţilor de tranziţie dintr-o stare în alta a modelului Markov se are în vedere faptul că la calculul unui element intervin un număr de două evenimente.

Evenimentul E1: La începutul intervalului de lungime T o parte din cererile emise pot fi rezolvate ca urmare a eliberării blocului de memorie solicitat sau a faptului că o magistrală devine disponibilă;

Evenimentul E2: La începutul intervalului T sînt generate cereri noi de acces la resursele sistemului mμP.

Calculul probabilităţii de realizare a evenimentului E1 se face plecînd de la următoarele definiţii:

D.5.3.7. Probabilitatea de realizare a evenimentului E1 în condiţiile impuse de restricţiile C1 şi C2 este $R_{i,j}$ în care i reprezintă numărul total de cereri în aşteptare prezent la începutul intervalului de lungime T din care j cereri pot fi rezolvate.

D.5.3.8. Probabilitatea ca o cerere generată la începutul intervalului T poate fi rezolvată, în condiţiile impuse de restricţiile C1 şi C2 atunci cînd sînt j cereri în curs de rezolvare este r_j.

Pentru a-l putea calcula pe r_j se definesc următoarele evenimente

Evenimentul A: cererea provine de la unul din MP-oare la care nu este în curs de rezolvare nici o altă cerere;

Evenimentul B: cererea de transmitere a unui mesaj se adresează unui bloc de memorie care nu prezintă obiectul nici unei din cererile în curs de rezolvare;

Din regula lui Bayes se obţine că:

$$P(A \cap B) = P(A) \times P(B|A) \qquad (5.6)$$

Probabilitatea de apariţie a evenimentului A rezultă imediat în felul următor. Dacă în sistem sînt prezente N module procesoare, probabilitatea ca un MP să fie activ este dată de raportul 1/N şi deci probabilitatea ca cererea să provină de la unul din cele j MP-oare implicate în transmisie, va fi dată de j/N. Prin urmare probabilitatea ca cererea să parvină de la restul de MP-oare care nu sînt implicate în transmisie, este dată de 1 - j/N, relaţie care defineşte probabilitatea evenimentului căutat, deci pe P(A). La calculul probabilităţii de apariţie a evenimentului B condiţionat de A se procedează similar. Probabilitatea ca un bloc de memorie din cele M prezente să reprezinte destinaţia unei cereri de transmisie este dată de 1/M, şi ca urmare probabilitatea ca cererea nou generată să se adreseze la unul din cele j blocuri de memorie deja implicate în procesul de transmisie va fi dată de j/M. Rezultă imediat că probabilitatea ca cererea nou generată să nu se adreseze nici unuia din blocurile de memorie utilizate este dată de 1 - j/M. Inlocuind în relaţia 5.6 se obţine:

$$r_j = \begin{cases} (1 - j/N)(1 - j/M) & \text{dacă } j < B \leqslant N \\ 0 & \text{în caz contrar} \end{cases} \qquad (5.7)$$

D.5.3.9. Probabilitatea de apariţie a evenimentului E1 este $R_{i,j}$ unde i reprezintă numărul total de cereri în aşteptare prezente la începutul intervalului T, iar j numărul de cereri care

pot fi rezolvate.

Ca urmare a modului în care este definit, $R_{i,j}$ apare ca fiind dat de o sumă de doi termeni. Primul, defineşte probabilitatea ca j cereri de transmisie a unui mesaj pot fi rezolvate din numărul total de i-1 cereri aflate în aşteptare, la care se adaugă condiţia ca cererea nou generată la începutul intervalului nu poate fi rezolvată. Cel de al doilea termen va reprezenta probabilitatea că j-1 cereri pot fi rezolvate din cele i-1 aflate în aşteptare, la care se adaugă condiţia că cererea nou apărută la începutul intervalului T se poate rezolva. Ca urmare, avem:

$$R_{i,j} = R_{i-1,j} \times (1 - r_j) + R_{i-1,j-1} \times r_{j-1} \qquad (5.8)$$

pentru care condiţiile iniţiale sînt date de:

pentru $j = 1$ — datorită faptului că prezenţa unei singure cereri
$R_{1,j}^{\Downarrow} = 1$ în sistem reprezintă evenimentul sigur (cererea poate fi întotdeauna rezolvată);

pentru $j > 1$ — datorită faptului că reprezintă evenimentul im-
$R_{1,j}^{\Downarrow} = 0$ posibil (nu pot fi rezolvate j cereri atunci cînd este prezentă una singură în sistem);

pentru $i \geqslant 1$ — datorită de asemenea faptului că reprezintă eveni-
$R_{i,o}^{\Downarrow} = 0$ mentul imposibil, deoarece întotdeauna din cele i cereri în aşteptare există întotdeauna cel puţin o cerere care poate fi rezolvată.

Inlocuind relaţia 5.7 în 5.8 se obţine relaţia finală pentru $R_{i,j}$:

$$R_{i,j} = \begin{cases} R_{i-1,j}(1 - (1 - j/N)(1 - j/M)) + \\ R_{i-1,j-1}(1 - (j - 1)/N)(1 - (j - 1)/M) \\ \qquad \text{dacă } 1 \leqslant j < B \leqslant N \text{ cu } j \leqslant i \\ R_{i-1,j} + R_{i-1,j-1}(1 - (B - 1)/N)(1 - (B - 1)/M) \\ \qquad \text{dacă } j = B \end{cases} \qquad (5.9)$$

D.5.3.1o. Probabilitatea de apariţie a evenimentului E2 este $e_{i,j}$ unde i reprezintă numărul de cereri generate la începutul intervalului T, atunci cînd la începutul aceluiaşi interval sînt prezente j cereri în aşteptare.

Calculul lui $e_{i,j}$ presupune parcurgerea următoarelor etape:

- Deoarece emiterea unei cereri de către un MP într-un interval de lungime T s-a considerat ca reprezentînd un succes al unui experiment Bernoulli, atunci probabilitatea de apariţie a cererii în ca-

uză este dată de:

$$1 - e^{-\lambda T}$$

unde $1/\lambda$ reprezintă valoarea medie a intervalului dintre două generări consecutive.

- Probabilitatea ca în intervalul T un număr de i module procesoare să emită cereri va fi dată evident de:

$$(1 - e^{-\lambda T})^i$$

- probabilitatea ca în același interval restul de MP-oare care nu sînt implicate deja în procesul de transmitere a unui mesaj să nu emită cereri (sînt în număr de N-j-i unități) este dată de

$$(e^{-\lambda T})^{N-j-i}$$

- Deoarece cele i cereri pot proveni de la un număr de i module procesoare din cele N-j care pot emite astfel de cereri, rezultă că cele i module procesoare pot fi distribuite în combinări de N-j luate cîte i moduri printre cele N-j module procesoare disponibile.

Cumulînd rezultatele discutate anterior se obține că:

$$e_{i,j} = \binom{N-j}{i} (1 - e^{-\lambda T})^i (e^{-\lambda T})^{N-j-i} \qquad (5.10)$$

Relația comportă o singură restricție:

pentru $N < i + j$ $\Downarrow$ $e_{i,j} = 0$ datorită faptului că în sistemul mpP nu pot fi prezente cereri în așteptare și noi generate care să depășească numărul MP-oare.

Utilizînd ecuațiile 5.9 și 5.10 în condițiile de regim staționar se poate calcula $p_n, n = \overline{0,N}$. Deoarece din modul de definire a modelului Markov rezultă că în starea n se poate ajunge din oricare altă stare a sistemului, p_n va apare ca o sumă de termeni în care fiecare termen definește probabilitatea de a trece dintr-o stare i, $i = \overline{0,N}$ în starea n. In acest context prezența în sistem a unui număr de n cereri la începutul unei perioade de lungime T apare ca o reuniune de N + 1 evenimente incompatibile. Analiza lor se va efectua grupînd numărul de stări din care se face tranziția în patru categorii fig.5 4., criteriul de grupare fiind mărimea stării din care se face tranziția în starea n, aspect care conduce în final la apariția unui număr de patru categorii de probabilități de tranziții.

Cele afirmate anterior conduc la o relație pentru calculul lui

p_n de forma:

$$p_n = p_o q_{o,n} + p_1 q_{1,n} + \sum_{i=2}^{B} p_i q_{i,n} + \sum_{i=B+1}^{N} p_i q_{i,n} \qquad (5.11)$$

în care $q_{i,n}$ reprezintă elementul corespunzător liniei i şi coloanei n din matricea probabilităţilor de tranziţie $[Q]$,definind prin urmare probabilitatea de tranziţie din starea i în starea n.

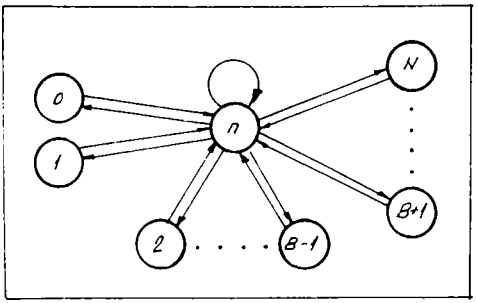


Fig.5.4. Graful tranziţiilor din starea i,$i=\overline{0,N}$ în starea n.

Apar patru categorii de tranziţii:

1) din starea 0 în starea n;

2) din starea 1 în starea n;

3) din stările $i=\overline{2,n}$ în starea n;

4) din stările $i=\overline{n+1,N}$ în starea n.

Cele patru categorii de probabilităţi de tranziţie se calculează după cum urmează:

1) Sistemul este iniţial în starea 0. Dacă în sistemul mμP nu este prezent nici un mesaj,pentru a ajunge în starea n este necesară generarea unui număr de n cereri de transmisie a unui mesaj la începutul intervalului T. Ca urmare probabilitatea de tranziţie din starea 0 în n este de forma:

$$q_{o,n} = e_{n,o} \qquad \text{pentru } n = \overline{0,N} \qquad (5.12)$$

2) Sistemul este iniţial în starea 1. Dacă în sistemul mμP este prezentă o singură cerere de transmisie,aceasta poate fi întotdeauna rezolvată. Prin urmare este de asemenea necesar ca un număr de n module procesoare să genereze cereri de transmisie la începutul intervalului T,aspect care face ca probabilitatea de tranziţie din starea 1 în starea n să fie de forma:

$$q_{1,n} = R_{1,1} e_{n,1} \qquad \text{pentru } n = \overline{0,N} \qquad (5.13)$$

3) Sistemul este iniţial prezent în una din stările i=2,3,.. n-1,n,n+1,..B. Tranziţia din una din stările i în starea n comportă în acest caz două situaţii:

- Prima apare atunci cînd i=2,3,.. n-1,n,deci cînd se efectuează trecerea dintr-o stare în care numărul de cereri nerezolvate prezent la începutul intervalului T este mai mic sau egal cu n. Acest aspect impune ca un număr n-i module procesoare să genereze cereri de transmisie la începutul intervalului

T. Pe de altă parte însă,la începutul intervalului T pot fi rezolvate un număr de k cereri cu $k=\overline{1,i}$ fapt care impune mărirea numărului de MP-oare care trebuie să emită cereri cu un număr egal de k. Deci numărul total de MP-oare care trebuie să emită noi cereri de transmisie la începutul intervalului T pentru a asigura tranziţia în starea n este dat de n-i+k. Probabilitatea de tranziţie din i în n este în acest caz de forma:

$$q_{i,n} = \sum_{k=1}^{i} R_{i,k} e_{n-i+k,i} \qquad \text{pentru} \begin{cases} n=\overline{0,N} \text{ şi} \\ i \leqslant n \end{cases} \qquad (5.14)$$

- Cea de a doua situaţie,apare atunci cînd i=n+1,..n+k,..B, deci se efectuează trecerea dintr-o stare în care numărul de cereri nerezolvate la începutul intervalului T este mai mare ca n. Pentru ca sistemul să ajungă în starea n,în acest caz,este necesar ca numărul de k=i-n cereri din cele i prezente să poată fi rezolvate. Pe de altă parte însă,mai pot fi rezolvate pe lîngă cele i-n cereri şi un număr de 1,2,.i cereri,caz în care pentru a se asigura trecerea în starea n este necesar ca un număr de n+k-i module procesoare să genereze noi cereri care să compenseze numărul de cereri rezolvate suplimentar. Ca urmare,probabilitatea de tranziţie din starea i în n apare şi în acest caz ca o sumă de forma:

$$q_{i,n} = \sum_{k=i-n}^{i} R_{i,k} e_{n-1+k,i} \qquad \text{pentru} \begin{cases} n=\overline{0,N} \\ i > n \end{cases} \qquad (5.15)$$

Cele două relaţii 5.14 şi 5.15 pot fi cumulate în una singură:

$$q_{i,n} = \sum_{k=1}^{i} R_{i,k} e_{n-1+k,i} + \sum_{k=i-n}^{i} R_{i,k} e_{n-i+k,i}$$

$$\text{pentru } n=\overline{0,N} \text{ şi } i=\overline{2,B}$$

Făcînd observaţia că cele două valori iniţiale k=1 şi k=i-n pot fi concatenate în una singură sub forma k=max(1,i-n) se obţine relaţia finală pentru probabilitatea de tranziţie din starea $i=\overline{2,B}$ în starea n:

$$q_{i,n} = \sum_{k=\max(1,i-n)}^{i} R_{i,k} e_{n-i+k,i} \qquad \text{pentru} \begin{cases} n=\overline{0,N} \\ i=\overline{2,B} \end{cases} \qquad (5.16)$$

4) La începutul intervalului T sistemul este prezent în una din stările i=B+1,B+2,..N. Este evident că în această situaţie apare

o condiţie suplimentară la tranziţia din starea i în starea n legată de numărul magistralelor sistemului. Faptul că la un moment dat numărul de cereri care pot fi rezolvate este limitat superior de numărul de magistrale disponibile,impune apariţia unui număr de două restricţii.

- Prima restricţie se referă la numărul maxim de cereri care pot fi rezolvate la începutul unui interval de lungime T pentru a se asigura tranziţia din starea i în starea n. Numărul acestor cereri nu poate depăşi numărul de magistrale pe care le are sistemul/mµP. Ca urmare probabilitatea de tranziţie din starea i în starea n care se calculează similar ca în cazul anterior este de forma:

$$q_{i,n} = \sum_{k=\max(1,i-n)}^{B} R_{i,k} e_{n-i+k,i} \qquad \text{pentru} \begin{cases} n=\overline{0,N} \\ i > B \end{cases} \quad (5.17)$$

- Cea de a doua restricţie se referă la faptul că numărul de stări i posibile din care se poate face tranziţia este dependent de numărul magistralelor. Este evident că numărul de stări i posibile teoretic din care se poate face tranziţia în starea n este dat de i=B+1,B+2,.,B+n,..N. Din punct de vedere însă practic,tranziţiile din stările cuprinse în domeniul B+n,B+n+1,..N nu au sens,deoarece nu este posibilă ajungerea în starea n,deoarece numărul de cereri care ar trebui rezolvate B+1,B+2,.. nu depăşeşte numărul magistralelor sistemului. Pe de altă parte B+n nu poate depăşi valoarea N (numărul de MP-oare din sistem). Cele două situaţii fac ca stările i posibile pentru care i>B din care se pot efectua tranziţiile în starea n să fie cuprinse numai în intervalul $[B+1, \min(B+n,N)]$.
  Cele două restricţii conduc la forma finală a probabilităţii de tranziţie din stările i>B în starea n care apare ca fiind:

$$q_{i,n} = \sum_{k=\max(1,i-n)}^{B} R_{i,k} e_{n-i+k,i} \qquad \text{pentru} \begin{cases} n=\overline{0,N} \text{ şi} \\ B+1 \leqslant i \leqslant \min(B+n,N) \end{cases} \quad (5.18)$$

5.3.3. Calculul distribuţiei de probabilitate.

Inlocuind probabilităţile de tranziţie definite prin 5.12,5.13,5.16 şi 5.18 în 5.11 pentru $n=\overline{0,N}$ se obţine un sistem liniar de N+1 necunoscute de forma:

$$[p] = [p]\,[Q] \qquad (5.19)$$

In relaţie $[p]$ este o matrice linie cu N+1 elemente,în care un element $p_n, n=\overline{0,N}$ defineşte probabilitatea ca în sistemul mμP să fie prezente un număr de n cereri de transmisie a unui mesaj de lungime constantă care aşteaptă să fie rezolvate. Matricea este de forma:

$$[p] = [p_o, p_1, \dots, p_n, \dots p_N] \qquad (5.2o)$$

cu

$$\sum_{n=0}^{N} p_n = 1 \qquad (5.21)$$

Cu $[Q]$,în relaţia 5.19 s-a notat matricea pătrată de dimensiune (N+1)x (N+1) a probabilităţilor de tranziţie dintr-o stare în alta a modelului Markov. Matricea este de forma:

$$\begin{array}{|c|c|c|c|}
\hline
e_{n,o} & e_{n,1} & \sum_{k=\max(1,i-n)}^{i} R_{i,k} e_{n-i+k,i} & \sum_{k=\max(1,i-n)}^{B} R_{i,k} e_{n-i+k,i} \\
n=\overline{0,N} & n=\overline{0,N} & n=\overline{0,N} & n=\overline{0,N} \\
 & & i=\overline{2,B} & i=\overline{B+1,\min(N,n+B)} \\
\hline
\end{array} \qquad (5.22)$$

Sistemul 5.19 este un sistem singular deoarece determinantul acestuia este întotdeauna egal cu zero. Acest aspect rezultă imediat din observaţia că în matricea $[Q]$ elementele de pe o coloană a matricii definesc probabilităţile de tranziţie din toate stările sistemului într-o stare a acestuia şi deci suma lor este întotdeauna egală cu 1

$$\sum_{i=o}^{N} q_{i,j} = 1 \qquad \text{pentru } j=\overline{0,N}$$

Pe de altă parte,forma sistemului 5.19 face ca pe diagonala principală a matricii sistemului să avem elemente de forma q-1. Ca urmare, suma elementelor de pe fiecare coloană a determinantului sistemului este egală cu zero,ceea ce face ca întotdeauna valoarea acestuia să fie zero [KM8o] . Pentru ca sistemul 5.1o să admită soluţie unică este obligatorie şi suficientă [KM8o] prezenţa condiţiei dată de relaţia 5.21. Prin rezolvarea sistemului liniar de N+1 ecuaţii necunoscute

$$[p] = [p][Q] \qquad (5.23)$$

cu condiţia

$$\sum_{n=0}^{N} p_n = 1$$

se obţin valorile pentru $p_n, n=\overline{0,N}$ care înlocuite în 5.4,5.5 şi 5.2 con-

duc la obţinerea capacităţii de trecere şi a timpului mediu de aşteptare pentru un sistem μP de tip P/D*M*B.

5.4. Model de tip P/E*M*B

In cadrul acestui tip de model [Mo87] considerăm că operaţiile de coordonare şi sincronizare în sistemul μP impun sistemului de operare operaţii de transfer pe magistralele sistemului în care lungimea mesajelor este exponenţial distribuită. Acest aspect este subliniat prin notaţia /E, în care E specifică caracteristica de distribuţie exponenţială a lungimii mesajelor care se transmit, fiind identică cu cea utilizată în teoria aşteptării [MC73] . In general utilizarea mesajelor cu lungime variabilă este caracteristică sistemelor de operare μP fără restricţii de sincronizare, categorie care înglobează acele sisteme în care mecanismele de sincronizare şi coordonare a execuţiei proceselor prezente în diversele MP-oare utilizează mesajele care se schimbă între aceste procese. Exemple din această categorie sînt date de sistemele de operare μP monolitice, partiţionate şi în general de toate celelalte tipuri, dacă în cadrul acestora sînt utilizate mecanisme de coordonare care utilizează mesaje de lungime variabilă.

5.4.1. Indici de performanţă.

Sistemul pentru care se construieşte modelul este cel a cărui caracteristici s-au prezentat în 5.2.2. şi a cărui arhitectură este dată în fig.5.1. Pentru acest μP se consideră că sistemul de operare implementat utilizează în cadrul mecanismelor de coordonare şi sincronizare a activităţilor o tehnică de tip mesaj în care lungimea mesajelor schimbate între procesele rezidente în MP-oare diferite este exponenţial distribuită. Datorită acestui aspect, nu mai este posibilă construirea unui model sincron, fiind obligatorie definirea unui model asincron. Modelul a cărui reprezentare cu reţele de şiruri de aşteptare este dată de asemenea de fig.5.2. este în acest caz mai complicat deoarece conduce la un număr mult mai mare de stări în lanţul Markov corespondent. P_e de altă parte însă el este în concordanţă cu situaţia reală prin aceea că alocarea unei resurse (bloc de memorie sau magistrală) se face secvenţial în fiecare moment de arbitrii memoriei comune sau magistralelor sistemului, în fiecare moment în care una din resursele în cauză devine disponibilă.

Mărimile care intervin în calculul indicilor de performanţă ai acestui model sînt:

- λ, frecvenţa de generare a cererilor de transmisie a unui mesaj, are aceiaşi semnificaţie şi definiţie D.5.3.3. ca în cazul modelului de tip P/D*M*B.

- p_n, probabilitatea ca în sistemul mμP în regim staţionar să fie prezente n cereri de transmisie a unui mesaj. Are aceiaşi semnificaţie cu cea dată de D.5.3.6., cu observaţia că de această dată p_n se calculează pentru fiecare moment în care una din resurse devine disponibilă.

Spre deosebire de modelul precedent în care durata de transmisie a unui mesaj era constantă, în prezentul model, datorită faptului că lungimea mesajelor este exponenţial distribuită rezultă că şi timpul necesar transmiterii fiecărui mesaj în parte este exponenţial distribuit. Acest aspect impune necesitatea definirii valorii medii a acestei distribuţii exponenţiale.

D.5.4.1. Valoarea medie a distribuţiei exponenţiale a timpului necesar transmiterii unui mesaj este dată de raportul dintre unitate şi frecvenţa (μ) de transmitere a mesajelor.

Indicii de performanţă prin care se caracterizează comportarea modelului sînt: numărul mediu de cereri prezente în sistem (L), timpul mediu de aşteptare (W) şi capacitatea de trecere (Λ). Toţi aceşti indici au aceiaşi semnificaţie ca în cazul modelului de tip P/D*M*B fiind definiţi de D.5.3.1., D.5.3.2. şi de relaţia 5.4. Deoarece timpului de transmisie a unui mesaj îi este caracteristică o distribuţie exponenţială cu media $1/\mu$, este aplicabilă relaţia lui Little [Ko78] pentru stabilirea legăturii dintre acesti indici de performanţă. Se obţine:

$$W = L/\Lambda \qquad (5.24)$$

Calculul capacităţii de trecere

In cadrul sistemului mμP fiecare MP în parte generează cereri de transmitere a unui mesaj cu o frecvenţă de λ cereri în unitatea de timp. Ca urmare numărul maxim de cereri care pot fi generate în unitatea de timp de către toate MP-oare ale sistemului mμP va fi dat de produsul dintre N, numărul MP-oare din sistem şi λ, deci de $N\lambda$. Pe de altă parte însă, deoarece L reprezintă numărul mediu de MP-oare a căror cereri urmează să fie rezolvate, produsul $L\lambda$ va reprezenta numărul mediu de cereri nerezolvate în unitatea de timp. Diferenţa dintre cererile care pot fi generate în unitatea de timp şi cererile care nu pot fi rezolvate în unitatea de timp, va reprezenta conform definiţiei D.5.3.5., capacitatea de trecere a sistemului mμP. Se obţine:

$$\Lambda = N\lambda - L\lambda$$

de unde:

$$\Lambda = (N - L)\lambda \qquad (5.25)$$

5.4.2. Structura modelului.

Plecînd de la definiţiile făcute anterior,se poate construi un Model Markov observînd starea sistemului în fiecare din momentele în care una din resursele sistemului (bloc de memorie sau magistrală) devine disponibilă. Modelul este definit de un vector de forma $(n_0,n_1,..,n_k,..n_N)$ în care un element n_k reprezintă mulţimea tuturor perechilor de forma (i,j) care îndeplinesc condiţia i+j=k,unde:

- i defineşte numărul de cereri în curs de rezolvare,reprezentînd deci numărul mesajelor în curs de transmisie;
- j defineşte numărul de cereri care aşteaptă eliberarea unei resurse,reprezentînd deci numărul de mesaje care nu pot fi transmise deoarece blocul de memorie căruia i se adresează este ocupat,sau o magistrală nu este disponibilă;
- i+j=k defineşte condiţia care impune ca numărul cererilor k care definesc un element n_k este dat de numărul cererilor în execuţie şi în aşteptare. Deoarece nu pot fi prezente situaţii în care sînt prezente cereri în aşteptare fără să existe nici o cerere în execuţie se procedează la eliminarea stărilor de forma (0,j) şi deci la impunerea condiţiei ca i> 0. Este evident de asemenea că suma celor două variabile nu poate depăşi valoarea N,numărul de MP-oare din sistem.

Rezumînd,modelul este definit de vectorul

$$(n_0,n_1,..,n_k,...n_N)$$

în care un element este dat de:

$$n_k = \left\{(i,j)\right\}_{i+j=k} \quad \text{şi } i>0$$

Numărul de stări pentru modelul astfel definit,a cărui graf a tranziţiilor între stări este prezentat în fig.5.4.,este egal, datorită condiţiilor impuse de i>0 şi i+j≤N, cu Nx(N+1)/2.

Calculul probabilităţilor de tranziţie

Pentru a calcula probabilitatea de tranziţie dintr-o stare în alta a modelului Markov fără a cunoaşte exact care este MP care emite cererea şi către ce bloc de memorie se adresează aceasta,se face presupunerea [Ko78] că distribuţia sursei şi destinaţiei este dată de o distribuţie uniformă.Deoarece observarea stării sistemului se face în momen-

tele în care o cerere s-a rezolvat (ceea ce atrage după sine eliberarea resurselor alocate) se va proceda la definirea acesteia prin perechea (s,d),în care s defineşte MP sursă a cererii,iar d,blocul de memorie comună căruia i se adresează cererea în cauză.

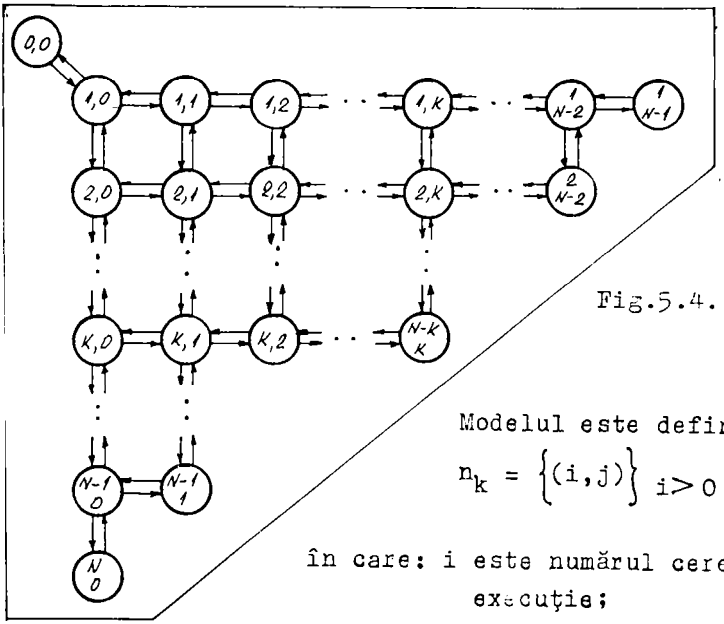


Fig.5.4. Graful tranziţiilor între stările modelului P/E*M*B.

Modelul este definit de:

$$n_k = \{(i,j)\} \quad i>0 \qquad \text{cu } i+j=k \text{ pentru } k=\overline{1,N}$$

în care: i este numărul cererilor în curs de execuţie;
j este numărul cererilor în aşteptare.

Calculul elementelor matricii $[Q]$ a probabilităţilor de tranziţie impune efectuarea următoarelor definiţii:

D.5.4.2. Probabilitatea ca o cerere din cele j prezente în aşteptare,se poate rezolva în condiţiile impuse de restricţiile C1 şi C2,ca urmare a terminării rezolvării cererii definite de (s,d) este m_i,atunci cînd în sistem sînt prezente i-1 cereri în curs de rezolvare.

D.5.4.3. Probabilitatea că există printre cele j cereri care aşteaptă să fie rezolvate,cel puţin o cerere care se poate rezolva,ca urmare a rezolvării cererii definite de (s,d) este $n_{i,j}$ atunci cînd în sistem sînt prezente i cereri în curs de rezolvare.

Pentru a-l putea calcula pe m_i se definesc următoarele evenimente

Evenimentul A: destinaţia cererii de transmisie a mesajului este d;
Evenimentul B: cererea provine de la unul din MP-oare care nu este în transmisie;
Evenimentul C: sînt prezente i-1 cereri în curs de rezolvare şi cererea de transmisie a mesajului definit de (s,d) s-a re-

zolvat.

Probabilitatea de realizare a evenimentului A,referitoare la faptul că un bloc de memorie din cele M prezente în sistem reprezintă destinaţia cererii de transmisie a mesajului este dată de 1/M. Pentru calculul probabilităţii de realizare a evenimentului B condiţionat de C,se procedează similar. Astfel,probabilitatea ca cererea să provină de la unul din cele N module procesoare în condiţia de distribuţie uniformă a mesajului de transmis este 1/N şi deci probabilitatea ca cererea să provină de la unul din cele i-1 module procesoare implicate în transmisie este (i-1)/N. Ca urmare,probabilitatea ca cererea să parvină de la restul MP-oare care nu sînt implicate în transmisie este dată de 1-(i-1)/N. Deoarece evenimentele A şi B C sînt independente pentru calculul intersecţiei celor două,se aplică regula produsului probabilităţilor,obţinîndu-se că:

$$m_i = P(A \cap B|C) = P(A) \times P(B|C) = (1 - (i - 1)/N)/M$$

şi deoarece pentru i=1 se obţine evenimentul sigur,întrucît întotdeauna dacă există o singură cerere de transmisie a unui mesaj,aceasta poate fi rezolvată întotdeauna pentru m_i,avem:

$$m_i = \begin{cases} (1 - (i - 1)/N/M & \text{dacă } i > 1 \\ 1 & \text{dacă } i = 1 \end{cases} \qquad (5.26)$$

Pentru a-l putea calcula pe $n_{i,j}$ se definesc următoarele evenimente:

Evenimentul D: nu există nici o cerere care se poate rezolva printre cele j în aşteptare;

Evenimentul E: sînt prezente i cereri în curs de rezolvare.

Utilizînd cele două evenimente se face observaţia că :

$$n_{i,j} = 1 - P(D|E)$$

Este evident că probabilitatea de realizare a evenimentului "nu există o cerere care să se poată rezolva atunci cînd sînt deja i cereri în curs de rezolvare" este dată de $1-m_i$. Ca urmare,probabilitatea de realizare a evenimentului D|E este dată de $(1-m_i)^j$ ceea ce face ca pentru $n_{i,j}$ să se obţină relaţia:

$$n_{i,j} = 1 - (1 - m_i)^j \qquad (5.27)$$

După cum s-a specificat,o stare a modelului Markov este definită de perechea (i,j) în care i specifică numărul de cereri în curs de rezolvare,iar j numărul de cereri în aşteptarea eliberării unei resurse. In acest context,trecerea dintr-o stare în alta a modelului este rezul-

tatul apariţiei a două evenimente. Primul eveniment,este dat de generarea unei cereri noi de transmisie a unui mesaj. Numărul maxim de cereri noi care pot fi generate atunci cînd sistemul este prezent în starea (i,j) este dat de produsul dintre numărul de MP-oare neimplicate în procesul de transmisie,deci de N-(i+j) şi frecvenţa de generare în unitatea de timp a unei cereri de către un MP. Cel de al doilea eveniment,este dat de terminarea transmisiei unui mesaj. Numărul maxim de terminări care pot apare în starea (i,j) este dat de produsul dintre i,numărul de cereri în curs de rezolvare şi frecvenţa de transmisie a unui mesaj în unitatea de timp. Dacă se notează cu t(i,j) numărul maxim de cereri care pot cauza trecerea în altă stare,avem:

$$t(i,j) = (N - i - j)\lambda + i\mu \qquad (5.28)$$

Presupunînd că sistemul este în starea (i,j),trecerea în altă stare conform modului în care s-a definit modelul,are loc numai atunci cînd apare o nouă cerere de transmisie sau cînd s-a terminat de rezolvat o cerere. Cele două evenimente în funcţie de posibilitatea rezolvării cererii de transmisie a mesajului conduc fiecare la alte două situaţii,aspect care face ca tranziţia din (i,j) să se poată face numai pentru stările imediat vecine (i+1,j);(i,j+1);(i-1,j);(i,j-1),ca în fig.5.4. Analizate pe rînd,probabilităţile de tranziţie apar după cum urmează.

1) Tranziţia din starea (i,j) în (i+1,j),apare atunci cînd cererea nou generată poate fi rezolvată imediat. Situaţia este rezultatul a două evenimente: primul,este dat de generarea unei noi cereri,iar cel de al doilea,de faptul că cererea în cauză poate fi rezolvată. Probabilitatea de realizare a primului eveniment este dată de raportul dintre numărul cererilor care se pot genera în starea respectivă $(N-i-j)\lambda$ şi numărul total de cereri care pot cauza trecerea în altă stare dat de t(i,j). Probabilitatea de realizare a celui de al doilea eveniment este dată de r_i care se calculează conform relaţiei 5.7. Se obţine că:

$$q((i,j) \rightarrow (i+1,j)) = \begin{cases} (N - i - j)\lambda r_i / t(i,j) & \text{dacă } i < B \\ 0 & \text{dacă } i \geq B \end{cases} \qquad (5.28)$$

2) Tranziţia din starea (i,j) în (i+1,j),apare cînd cererea nou generată nu poate fi rezolvată imediat. Situaţia este rezultatul a două evenimente: primul este dat de generarea unei noi cereri iar cel de al doilea,de faptul că cererea generată nu poate fi rezolvată,rămînînd în aşteptare. Probabilitatea de realizare a pri-

mului eveniment,se calculează similar ca în cazul anterior,iar a celui de al doilea eveniment este dată de $(1-r_i)$. Se obţine că:

$$q((i,j) \to (i,j+1)) = \begin{cases} (N-i-j)\lambda(1-r_i)/t(i,j) & \text{dacă } i<B \\ (N-i-j)\lambda/t(i,j) & \text{dacă } i=B \end{cases} \quad (5.29)$$

3) Tranziţia din starea (i,j) în (i-1,j) apare cînd s-a terminat transmisia unui mesaj. Situaţia este rezultatul a două evenimente: primul,este dat de terminarea cererii de transmisie a unui mesaj din cele i cereri în curs de transmisie,iar cel de al doilea,de faptul că nici una din cele j cereri prezente în aşteptare nu îndeplineşte condiţiile de rezolvare. Probabilitatea de realizare a primului eveniment,este dată de raportul dintre numărul maxim de terminări care pot apare în starea (i,j) dat de $i\mu$ şi numărul total de cereri t(i,j) care pot cauza trecerea în altă stare. Probabilitatea de realizare a celui de al doilea eveniment este dată de $(1-n_{i,j})$. Pentru i=B probabilitatea de realizare a celui de al doilea eveniment prezintă o formă particulară. Astfel, $n_{B,j}$ reprezintă probabilitatea de realizare a evenimentului: este prezentă o cerere în aşteptare care se poate rezolva dacă există o magistrală disponibilă. Este evident că $1-n_{B,j}$,va reprezenta evenimentul complementar: nu este prezentă o cerere în aşteptare care se poate rezolva dacă o magistrală este disponibilă. Deoarece toate j cereri prezente în aşteptare trebuie să nu poată fi transmise în condiţiile existenţei unei magistrale disponibile,rezultă o reuniune de j evenimente independente. Ca urmare forma finală a evenimentului căutat pentru i=B este $(1-n_{B,j})j$. Se obţine că:

$$q((i,j) \to (i-1,j)) = \begin{cases} i\mu(1-n_{i,j})/t(i,j) & \text{pentru } 0<i<B \\ B\mu(1-n_{B,j})j/t(i,j) & \text{pentru } i=B \text{ şi } j>0 \\ i\mu/t(i,j) & \text{pentru } i\leq B \text{ şi } j=0 \end{cases} \quad (5.30)$$

4) Tranziţia din starea (i,j) în (i,j-1) apare cînd s-a terminat transmisia unui mesaj şi o cerere din cele j în aşteptare îndeplineşte condiţiile de a fi rezolvate. Probabilitatea de realizare a primului eveniment se calculează ca în cazul precedent, iar a celui de al doilea este dată de $n_{i,j}$. O formă particulară o prezintă pentru i=B,probabilitatea de realizare a celui de al doilea eveniment. Deoarece $(1-n_{B,j})j$ determină probabilitatea inexistenţei printre cele j cereri în aşteptare a uneia care să poată fi transmisă,rezultă imediat că $1-(1-n_{B,j})j$ va defini pro-

babilitatea căutată. Se obţine:

$$q((i,j) - (i,j-1)) = \begin{cases} i\mu n_{i,j}/t(i,j) & \text{pentru } 0<i<B \\ B\mu(1-(1-n_{B,j})j)/t(i,j) & \text{pentru } i=B \end{cases} \quad (5.31)$$

5.4.3. Calculul distribuţiei de probabilitate.

Pentru a calcula distribuţia de probabilitate se face presupunere că sistemul este în regim staţionar,deci numărul de cereri rezolvate în unitatea de timp,cît şi numărul de cereri generate în acelaşi interval este constant [Ko78] . Această presupunere ne permite să definim probabilitatea ca sistemul să fie prezent în una din stările modelului Markov astfel:

D.5.4.4. Probabilitatea ca în sistem să existe i cereri în curs de rezolvare şi j cereri în aşteptarea eliberării unei resurse este dată de $\pi_{i,j}$.

Ca urmare distribuţia de probabilitate a stării staţionare este dată de elementele matricii monodimensionale $[\pi]$,în care un element al matricii defineşte probabilitatea ca sistemul să se găsească în starea $n_k=(i,j)$ cu $k=i+j$.

Pentru un model dat $n_k=\{(i,j)\}_{i+j=k}$,unde $k=\overline{0,N}$ şi $i>0$,utilizînd relaţiile 5.28-5.31 pentru calculul elementelor matricii probabilităţilor de tranziţie,se poate determina [St76,Io77] ,distribuţia de probabilitate a stării staţionare plecînd de la sistemul liniar de ecuaţii cu $N(N+1)/2$ necunoscute de forma:

$$[\pi] = [\pi][Q] \quad (5.32)$$

cu

$$\sum_i \sum_j \pi_{i,j} = 1 \quad (5.33)$$

Deoarece sistemul este singular,din aceleaşi considerente ca şi cele prezente în 5.3 3.,pentru a-l rezolva este suficientă prezenţa condiţiei 5.3.3.,care dă posibilitatea obţinerii unei soluţii unice. Utilizînd probabilităţile astfel calculate se poate calcula probabilitatea ca în sistem să fie prezente k cereri (în rezolvare şi în aşteptare). Este evident dă deoarece n_k este dat de toate stările (i,j) posibile pentru care sînt îndeplinite condiţiile $i+j=k$ şi $i>0$,rezultă că p_k va apare ca o sumă de probabilităţi de forma:

$$p_k = \sum_{\substack{i+j=k \\ i>0}} \pi_{i,j} \qquad \text{cu } k=\overline{0,N} \qquad (5.34)$$

Probabilităţile astfel determinate sînt utilizate în relaţia 5.4 pentru calcul numărului de cereri în aşteptare din sistem L,ca şi în relaţiile 5.24 şi 5.25 pentru calculul indicilor de performanţă W-timp mediu de aşteptare şi Λ -capacitatea de trecere a sistemului mμP.

5.5. Studiul comparativ a sistemelor multimicroprocesor de tip P/D*M*B şi P/B*M*B.

Pentru a evalua performanţele sistemelor mμP de tip P/D*M*B şi respectiv P/B*M*B,cu scopul de a emite recomandări privind alegerea unui latom de operare sau a altuia funcţie de aplicaţia căreia îi este destinat sistemul,s-au avut în vedere următoarele aspecte:

- în marea majoritate a aplicaţiilor numărul MP-oare nu depăşeşte cifra de 3 sau 4,existînd un număr mic de aplicaţii care să necesite depăşirea acestor cifre;
- din considerente constructive,de fiecare dată se urmăreşte obţinerea unei structuri hard cît mai simple,fapt care face ca în sistemele mμP utilizarea unei singure magistrale la care sînt conectate toate MP-oare să fie soluţia cea mai des întîlnită. Pentru aplicaţii de timp real,în care se pun condiţii severe privind timpul de răspuns a întregului sistem numărul de două magistrale este în foarte puţine cazuri depăşit;
- ca rezultat a existenţei unei singure magistrale,sistemele mμP utilizează o singură memorie comună. Pentru cazurile în care aplicaţia impune utilizarea unui număr de două magistrale,soluţia normală adoptată,constă în utilizarea unui număr de două blocuri de memorie,număr care în anumite situaţii poate fi mărit pînă la trei blocuri de memorie comună.

Toate considerentele expuse mai sus au impus ca studiul să fie efectuat pentru sisteme în care numărul MP-oare este de 3 sau 4 şi în care informaţia este vehiculată prin una,maxim două magistrale,fiecărei arhitecturi corespunzîndu-i unu sau două blocuri de memorie comună.

Pentru a asigura un caracter cît mai mare de generalitate,calculele s-au efectuat în unităţi de timp adimensionale. Astfel,funcţie de tipul aplicaţiei,unitatea de timp poate fi: una,două sau mai multe microsecunde,zeci de microsecunde,milisecunde,etc. Caracterul de adimen-

sionabilitate a unităţii de timp,este asigurat de faptul că numărul de cereri de alocare a unei resurse,memorie sau magistrală,se poate da în număr de cereri în unitatea de timp,indiferent de semnificaţia acordată unităţii de timp alese. De asemenea,faptul că timpul necesar transmiterii unui mesaj de lungime constantă sau variabilă spre blocul de memorie,sau de la un bloc de memorie,poate fi de asemenea dat în unităţi de timp,măreşte caracterul de generalitate a rezultatelor obţinute. In acest context,în calculele efectuate s-a considerat că frecvenţa de generare a cererilor pe întreg sistemul mμP este de 0.25 0.5,1,1,5,2,2.5 şi 3 cereri de alocare în unitatea de timp. Durata de Transmisie a unui mesaj este pentru mesajele de lungime constantă de o unitate de timp,iar pentru cele de lungime variabilă este dată de o distribuţie exponenţială cu media egală de asemenea cu o unitate de timp. Valorile numerice pentru frecvenţa de generare a cererilor şi pentru timpul mediu de transmitere a unui mesaj,s-au obţinut pe baza datelor prezente în literatură [Sw77,Ng86,LZ86] .

Pentru fiecare din cele două tipuri de sisteme mμP s-au construit cîte un program în limbajul Fortran 77,care s-a implementat pe un sistem de tip M118. Deoarece după calculul matricii probabilităţilor de tranziţie este necesară rezolvarea unui sistem liniar de ecuaţii a cărui dimensiune este relativ mare (de exemplu,pentru un model de tip 4/E*2*2,se obţine un sistem de 8 ecuaţii cu 8 necunoscute) a fost necesară utilizarea unei metode eficiente şi rapide de rezolvare. Soluţia aleasă a constat din preluarea şi adaptarea pentru sistemul M118 a unui subprogram prezent în biblioteca matematică a sistemelor de tip Felix [**73] .

Rezultate obţinute

Rezumînd,studiul s-a efectuat pentru următoarele tipuri de sisteme mμP:

1) Sisteme mμP în care sînt prezente un număr de 3 module procesoare,una,respectiv două magistrale şi în care suportul pentru schimbul de informaţii dintre MP-oare ale sistemului se efectuează prin intermediul a una sau două memorii comune. Pentru fiecare configuraţie s-au considerat prezente două sisteme de operare: primul,care utilizează mesaje de lungime constantă,iar cel de al doilea,mesaje de lungime variabilă. Ca urmare,au rezultat următoarele variante 3/D*1*1; 3/E*1*1; 3/D*2*2; 3/E*2*2,pentru care s-au obţinut,pentru timpul mediu de aşteptare şi capacitatea de trecere rezultatele prezentate în tabelele a şi b din fig.5.5.

(număr cereri/unitatea de timp)	(unităţi de timp)		(nr.cereri/u.de timp)	
	3/D*1*1	3/E*1*1	3/D*1*1	3/E*1*1
0.25	1.74	1.46	0.52	0.55
0.5	1.91	1.8	0.77	0.79
1	2.18	2.2	0.94	0.94
1.5	2.37	2.41	0.99	0.97
2	2.5	2.54	1	0.99
2.5	2.6	2.62	1	1
3	2.7	2.68	1	1

a) 3/D*1*1 şi 3/E*1*1

(număr cereri/unitatea de timp)	(unităţi de timp)		(nr.cereri/u.de timp)	
	3/D*2*2	3/E*2*2	3/D*2*2	3/E*2*2
0.25	1.65	1.31	0.35	0.56
0.5	1.75	1.54	0.8	0.85
1	1.89	1.84	1.04	1.05
2	2.03	2.12	1.18	1.14
2.5	2.08	2.19	1.21	1.16
3	2.11	2.25	1.23	1.17

b) 3/D*2*2 şi 3/E*2*2

Fig.5.5. Indici de performanţă pentru un sistem mμP în care P=3; M=1 şi 2; B=1 şi 2 în două variante /D şi /E a sistemului de operare.

2) Cel de al doilea tip de sisteme mμP s-a considerat că sînt formate din 4 module procesoare interconectate prin una,respectiv prin două magistrale. Ca şi în cazul precedent schimbul de informaţii dintre MP-oare ale sistemului se efectuează prin intermediul a una sau două memorii comune. Pentru fiecare configuraţie s-au considerat de asemenea că sînt implementate două tipuri de sisteme

de operare: primul care utilizează mesaje de lungime constantă,iar cel de al doilea,mesaje de lungime variabilă în coordonarea şi sincronizarea activităţilor în sistemul mμP. Ca urmare,au rezultat următoarele variante 4/D*1*1; 4/E*1*1; 4/D*2*2; 4/E*2*2 pentru care s-au obţinut,pentru timpul mediu de aşteptare şi capacitatea de trecere,rezultatele prezentate în tabelele a şi b din fig.5.6.

(număr cereri/unitatea de timp)	(unităţi de timp)		(nr.cereri/unit.de timp)	
	4/D*1*1	4/E*1*1	4/D*1*1	4/E*1*1
0.25	1.93	1.8	0.67	0.69
0.5	2.37	2.42	0.92	0.9
1	3	3.1	0.99	0.98
1.5	3.33	3.35	1	1
2	3.5	3.51	1	1
2.5	3.6	3.62	1	1
3	3.7	3.7	1	1

a) 4/D*1*1 şi 4/E*1*1

(număr cereri/unitatea de timp)	(unităţi de timp)		(nr.cereri/unit.de timp)	
	4/D*2*2	4/E*2*2	4/D*2*2	4/E*2*2
0.25	1.72	1.46	0.7	0.73
0.5	1.88	1.84	1.03	1.04
1	2.09	2.3	1.3	1.21
1.5	2.2	2.54	1.39	1.25
2	2.28	2.67	1.44	1.26
2.5	2.34	2.76	1.46	1.27
3	2.38	2.82	1.47	1.28

b) 4/D*2*2 şi 4/E*2*2

Fig.5.6. Indici de performanţă pentru un sistem mμP în care P=4; M=1 şi 2; B=1 şi 2 în două variante /D şi /E a sistemului de operare.

Validarea rezultatelor

Pentru a verifica corectitudinea rezultatelor obţinute s-a utilizat o tehnică de simulare [Vă77,Răl81,D086] . In cadrul acesteia,s-a construit pentru fiecare din tipurile de sisteme mμP analizate,cîte un model de simulare utilizînd limbajul GPSS implementat în varianta SIMUB pe sistemele de calcul din gama Felix [Vă82] . După calibrarea generatoarelor de numere pseudoaleatoare utilizate în cadrul modelelor,care a constat în alegerea celei mai bune (din punct de vedere statistic) secvenţe de numere pseudoaleatoare care poate fi generată, s-a trecut la operaţia propriu zisă de simulare. Aceasta s-a efectuat pentru o frecvenţă de generare a cererilor de transmisie egală cu 0.25,0.5,1,1.5,2,2.5, şi 3 cereri în unitatea de timp,considerînd că durata de transmisie a mesajelor este constantă şi egală cu o unitate de timp sau exponenţială,cu media egală de asemenea cu o unitate de timp. Rezultatele obţinute pentru un interval de simulare de 1000 unităţi de timp,echivalentul unei durate egale a funcţionării unui sistem real mμP au confirmat rezultatele obţinute de modelele analitice şi deci implicit viabilitatea acestora. Este de remarcat faptul că execuţia unui singur model de simulare din cele 8 analizate,a durat aproximativ 7 minute,obţinîndu-se aceleaşi rezultate ca şi cele rezultate din execuţia modelului analitic corespunzător,dar cu observaţia că execuţia acestuia din urmă a durat mai puţin de lo secunde.

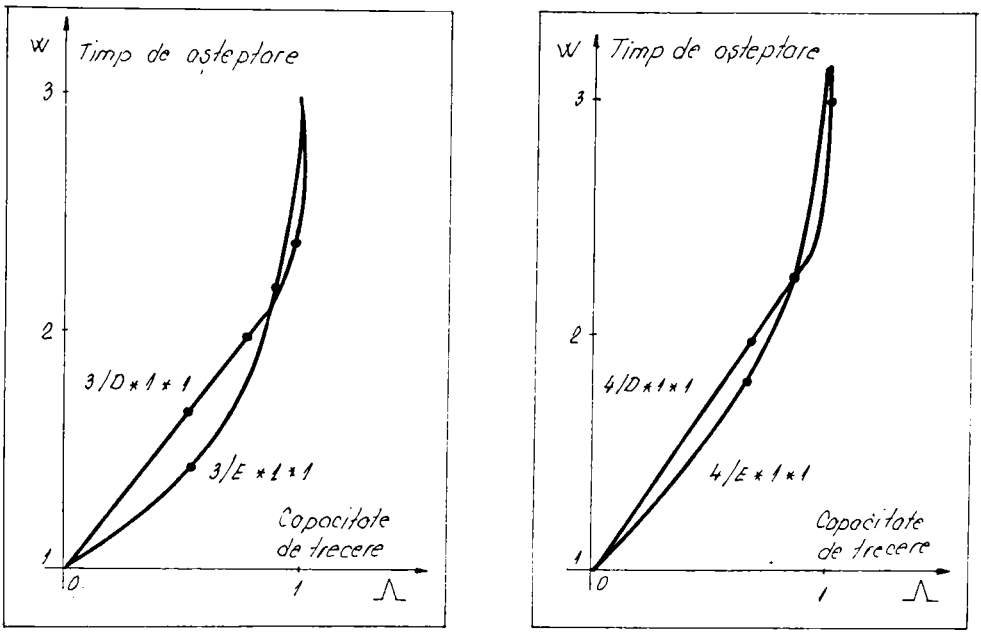
Interpretarea rezultatelor

Din analiza rezultatelor obţinute au rezultat următoarele observaţii:

- Performanţele sistemelor mμP care utilizează sisteme de operare de tip P/E*M*B este mai bună în cazul frecvenţelor mici de generare a cererilor de alocare a unui bloc de memorie (magistrală) decît în cazul în care acelaşi sistem utilizează un sistem de operare de tip P/D*M*B. Astfel pentru frecvenţe de generare a cererilor cuprinse între (0.25-1) cereri în unitatea de timp,pentru un sistem cu 3 module procesoare,1 magistrală şi un bloc de memorie comună,timpul de aşteptare a unei cereri pentru a fi rezolvată este cu 2o% pînă la 5% mai mic pentru un sistem de tip P/E*M*B faţă de unul de tip P/D*M*B. Aceasta în contextul în care capacitatea de trecere a întregului sistem este mai mare cu 6% pînă la 4%. Aceleaşi valori puţin mai mici,se păstrează şi în cazul în care numărul modulelor procesoare creşte la 4. Ambele afirmaţii rezultă atît din analiza valorilor numerice prezente în tabelele 5.5a şi 5.6a,cît şi din graficele în care s-a trasat timpul de aş-

teptare funcţie de capacitatea de trecere,fig.5.7a şi b.

- Pentru sistemele mμP în care frecvenţa de generare a cererilor de alocare a unui bloc de memorie este mare,comportarea sistemelor care au implementate sisteme de operare de tip P/D*M*B este mai bună decît cele care utilizează sisteme de tip P/E*M*B. Astfel pentru cazurile în care frecvenţa de generare a cererilor este mai mare de o cerere în unitatea de timp,timpul de aşteptare a unei cereri pentru a fi rezolvată este mai mic cu 2 pînă la 3%,pentru sistemele de tip P/D*M*B faţă de sistemele de tip P/E*M*B. Acest aspect rezultă de asemenea uşor din fig.5.7a şi b.



a) 3/D*1*1 şi 3/E*1*1 b) 4/D*1*1 şi 4/E*1*1

Fig.5.7. Modificarea timpului de aşteptare pentru un sistem mμP cu 3, respectiv 4 MP-oare,o magistrală şi o memorie comună,funcţie de capacitatea de trecere a sistemului.

- Pentru sistemele mμP în care sînt prezente două magistrale, indicii de performanţă pentru domeniul frecvenţelor mici de generare a cererilor cuprinse între (0.25-1) cereri în unitatea de timp,sînt mai buni pentru cazul în care sistemul de operare este de tip P/E*M*B,faţă de cazul în care se utilizează un sistem de tip P/D*M*B. Astfel,timpul de aşteptare este cu 25% pînă la 7% mai mic pentru un sistem P/E*M*B în contextul unei capacităţi de

trecere cu 4% pînă la 2% mai mare faţă de un sistem P/D*M*B. Ca şi în cazul sistemelor cu o singură magistrală,atunci cînd frecvenţa de generare a cererilor depăşeşte numărul de cereri în unitatea de timp,performanţele pentru sistemele de tip P/D*M*B devin mai bune decît în cazul sistemelor P/E*M*B. De data aceasta însă timpul de aşteptare este cu 21% pînă la 1o% mai mic pentru sistemele de tip P/D*M*B faţă de cazul în care se utilizează un sistem de tip P/E*M*B,în contextul unei creşteri de aproximativ 1o% a capacităţii de trecere,fig.5.5b şi 5.6b.

- O altă analiză care trebuie efectuată se referă la oportunitatea trecerii de la un sistem mμP cu o singură magistrală la un sistem mμP cu două magistrale. Din interpretarea rezultatelor prezentate în tabelele din fig.5.5a şi b,rezultă următoarele concluzii. Pentru sistemele de tip P/D*M*B trecerea de la o magistrală la două magistrale se traduce printr-o reducere (funcţie de frecvenţa de generare a cererilor) cuprinse între 12% pînă la 2o% a timpului de aşteptare cu o creştere corespunzătoare a capacităţii de trecere de la 1% pînă la 1o%. Pentru sistemele de tip P/E*M*B trecerea de la o magistrală la două magistrale conduce la o diminuare a timpului de aşteptare a unei cereri cuprinsă între 6% pînă la aproximativ 13%,în contextul unei îmbunătăţiri de la 2% pînă la 1o% a capacităţii de trecere. Se observă că per ansamblu, performanţele sistemelor de tip P/D*M*B sînt superioare celor de tip P/E*M*B,atunci cînd se trece de la o magistrală la două magistrale.

Concluzii

Din interpretarea rezultatelor se desprind următoarele concluzii practice în ceea ce priveşte utilizarea unui sistem de operare de tip P/D*M*B sau de tip P/E*M*B.

- Pentru frecvenţe mici de generare a cererilor de alocare a unei resurse (bloc de memorie sau magistrală) pînă la o limită superioară de o cerere în unitatea de timp,se vor utiliza sisteme de tip P/E*M*B fără restricţii de sincronizare. Practic aceasta impune ca mesajele care se vehiculează între memoria comună şi MP-oare să aibă lungime variabilă.

- Pentru frecvenţe care depăşesc valoarea de o cerere în unitatea de timp,sistemele care conduc la cele mai bune performanţe sînt cele de tip P/D*M*B cu restricţii de sincronizare. Practic aceasta impune ca mesajele care se transmit să aibă o lungime constantă.

- In cazul în care aplicaţia impune un schimb mare de informaţii între MP-oare prezente în sistemul mμP, aspect care conduce la o frecvenţă ridicată a cererilor de alocare a unei resurse de tip memorie comună sau magistrală,devine obligatorie utilizarea unui sistem de operare în care se utilizează pentru coordonarea activităţilor o tehnică de tip mesaj, deci implementarea unui sistem de operare fără restricţii de sincronizare de tip P/E*M*B. Deoarece în acest caz utilizarea unui astfel de sistem conduce la un timp de aşteptare mare la rezolvarea unei cereri,se recomandă ca transferul de informaţii să se facă prin mesaje de lungime fixă. Astfel apare o excepţie,prin faptul că deşi s-a afirmat că sistemelor de tip P/E*M*B le sînt caracteristice mesaje de lungime variabilă,în acest caz se va proceda la o uniformizare a acestora, asigurîndu-se tuturor informaţiilor care se transmit aceiaşi lungime.

- Ori de cîte ori este posibil se va proceda la concatenarea a două sau mai multe informaţii într-un singur mesaj. Prin aceasta se va asigura o frecvenţă mai mică de generare a cererilor de alocare a unei magistrale sau bloc de memorie,chiar dacă lungimea mesajelor care se transmit se măreşte. Această recomandare se impune deoarece mărirea frecvenţei de generare a cererilor,conduce la o înrăutăţire a performanţelor în ansamblu a unui sistem mμP,avînd o contribuţie mai mare în acest sens decît lungimea mesajelor care se transmit.

- Din cele prezentate,rezultă că performanţele unui sistem mμP în care sînt prezente două magistrale,sînt întotdeauna superioare unuia dotat cu o singură magistrală. Din analiza rezultatelor prezentate în tabelele din fig.5.5 şi 5.6,rezultă însă că într-o serie de cazuri se poate renunţa la cea de a doua magistrală (de exemplu din punct de vedere a timpului de aşteptare,un sistem de tip 3/E*1*1 pentru care frecvenţa de generare a mesajelor este egală cu 1 este echivalent cu un sistem de tip 3/E*2*2 în care frecvenţa este egală cu 2). Inlocuind sistemul de operare,modificînd lungimea şi deci implicit frecvenţa de generare a cererilor,se poate obţine astfel o simplificare considerabilă a arhitecturii unui sistem mμP prin trecerea de la un sistem cu două magistrale la un sistem cu o singură magistrală.

6. IMPLEMENTAREA UNEI ARHITECTURI DE SISTEM DE OPERARE PE UN SISTEM DE CALCUL MULTIMICROPROCESOR.

Sistemul de operare a cărui descriere se va prezenta în capitolele 6 şi 7,a reprezentat obiectul unui contract cu Institutul de Proiectări în Automatică Bucureşti [**87b] . Scopul contractului la constituit analiza posibilităţilor de configurare a structurilor de calcul mμP cu module din familia MULTIPROM,în jurul magistralei MULTIPROM şi a elaborării unui sistem de operare pe 8 biţi cu facilităţi de intercomunicaţie cu alte MP-oare prin intermediul unei memorii comune. Proiectul s-a încadrat în lucrarea "Echipament multiprocesor pentru conducerea şi supravegherea proceselor tehnologice ECAROM 886".

6.1. Descrierea arhitecturilor sistemelor multimicroprocesor realizate cu module MULTIPROM.

Familia MULTIPROM conţine din punct de vedere arhitectural,un set le module compatibile funcţional prin asigurarea comunicaţiei pe acelaşi magistrală. In prezent există două tipuri principale de module:

- universale: module procesoare pe 8/16 biţi,unităţi aritmetice,interfeţe periferice,intrări/ieşiri numerice;
- specifice: module dedicate unei anumite aplicaţii.

Toate modulele respectă protocolul de comunicaţie al magistralei MULTIPROM,putînd fi cuplate în diverse variante şi moduri la diferitele tipuri de magistrale prezente în clasa magistralei MULTIPROM. In acest mod se asigură compatibilitatea pe "verticală",fiecare modul putînd fi înlocuit cu altul mai performant,fără a mai fi necesare modificări în cadrul structurii sistemului mμP. Datorită importanţei pe care o prezintă la alegerea tipului de sistem de operare şi a modului de implementare,caracteristicile modulelor care intervin în arhitectura unui sistem mμP,cît şi mai ales modul în care aceste module se pot interconecta,se va face în cele ce urmează o prezentare a acestora. Prezentarea se va face din punct de vedere a acelor aspecte care impun adoptarea de decizii privind structura sistemului de operare, sintaxa şi modul de implementare a primitivelor utilizate,ca şi modul în care se realizează funcţia de coordonare şi sincronizare a activităţilor din întregul sistem mμP.

6.1.1. Magistralele familiei MULTIPROM.

Familia MULTIPROM cuprinde următoarele magistrale structurate ierarhic:

- Magistrala sistem (MS). Este de tip AMS-M derivată din MULTIBUS, îndeplinind toate specificaţiile principale, funcţionale şi de timp, definite de standardul IEEE 796. Magistrala permite funcţionarea asincronă a mai multor MP master cu viteze diferite de lucru. Comanda MS poate fi exercitată la un moment dat numai de un singur MP master care activează liniile de adrese şi comenzi. Modulele slave nu pot comanda magistrala. Cererile simultane de acces la magistrală sînt sincronizate cu tactul BCLK/ al MS care poate fi generat de oricare din MP master, dar care este independent de tactul propriu al masterului. Cînd MS a fost atribuită unui MP master, viteza de transfer depinde numai de modulele implicate în transfer, comunicaţia efectuîndu-se asincron sub controlul semnalelor de dialog. Ca o ultimă observaţie, MP master care a cîştigat MS poate efectua transferuri simple sau multiple.

- Magistrala rezidentă (MR). Este derivată din MS prin reducerea liniilor de arbitrare şi de control pentru accesul pe magistrală, fapt care are ca urmare că numai două module procesoare pot coexista la un moment dat pe MR (restul liniilor rămîn nemodificate faţă de MS). Magistrala rezidentă se taliază în funcţie de tipul procesorului I8086, I8080 sau Z80, prezentînd avantajul atribuirii unor resurse locale fiecărui modul procesor, ceea ce conduce la degrevarea MS şi la creşterea vitezei de comunicaţie între două module procesoare.

- Magistrala de proces (MP). Funcţionează pe principiul "master-slave" cu un singur modul procesor master şi cu comunicaţie asincronă, paralelă. MP cuprinde două tipuri de magistrale distincte: o magistrală de proces numerică şi o magistrală de proces analogică. Deosebirea dintre cele două constă în prezenţa în primul caz, a unui canal numeric cu 8 linii de date şi în cel de al doilea, a unui canal analogic.

- Magistrala serială (MS). Este utilizată ca un mijloc suplimentar de comunicaţie de mare viteză între două module procesoare.

6.1.2. Tehnici de rezolvare a priorităţilor.

Datorită faptului că la un moment dat, într-un sistem mμP pot coexista mai multe MP-oare master, este necesară prezenţa unei tehnici de rezolvare a cererilor de acces la MS, emise de aceşti masteri. In cadrul familiei MULTIPROM se pun la dispoziţie un număr de trei tehnici de rezolvare: în serie, în paralel şi cu rotirea priorităţilor.

Rezolvarea serială a priorităţilor.

Principiul rezolvării seriale a priorităţilor a fost prezentat în detaliu în paragraful 2.4.1. Faţă de cele prezentate,în cazul tehnicii utilizate de modulele familiei MULTIPROM,apar cîteva diferenţe notabile. Astfel,linia BABR (numele ei în rezolvarea serială a priorităţilor MULTIPROM este BUSY/),este utilizată de o funcţie de blocare a MS ce se poate implementa în exteriorul arbitrului 8218,sau crea prin program la I8o86 via 8289. Această funcţie permite unui MP master,ce deţine în mod curent controlul MS,să şi-l menţină prin poziţionarea independentă a liniei BUSY/,pînă cînd se furnizează o comandă de deblocare ce invalidează linia BUSY/. Scopul urmărit,constă în a permite unui MP master să obţină controlul exclusiv al MS pentru executarea unor funcţii critice de sistem (operaţii DMA,TAS,etc.). In acest caz,toate MP master vor găsi în starea "ocupat" MS dacă o vor solicita. Funcţia de blocare a MS permite efectuarea unor transferuri rapide de date multiple,cînd se solicită o astfel de operaţie. Mai este de menţionat faptul că în cazul rezolvării seriale a priorităţilor pentru o frecvenţă a tactului de magistrală BCLK de loMHz,numărul de MP master care se pot conecta la MS este în număr de 3. O dată cu scăderea frecvenţei acest număr poate creşte.

Rezolvarea paralelă a priorităţilor.

Tehnica a cărui principiu s-a prezentat în paragraful 2.4.2.,prezintă cîteva particularităţi care sînt specifice modulelor MULTIPROM. Astfel,în această tehnică fiecare linie separată BRQ de solicitare a magistralei (numele ei şi a semnalului corespunzător în rezolvarea paralelă a priorităţilor MULTIPROM este BREQ/) este conectată la fiecare din arbitrii conectaţi la MS (fig.2.lo). Fiecare din aceste linii intră într-un codificator de prioritate (de exemplu 74148) care generează la ieşire,adresa binară a liniei BRQ cea mai prioritară conectată la intrări. Adresa binară de la ieşire,după ce a fost decodificată (74138,82o5 etc.),selectează linia corespunzătoare de garantare a priorităţii BGI (linia şi semnalul echivalent MULTIPROM fiind BPRN/) ce este conectată la arbitrul cu cea mai mare prioritate. Arbitrul recepţionînd prioritatea (BPRN/=O) permite MP master asociat,accesul la MS imediat ce magistrala devine disponibilă. Se observă că integritatea transferului este asigurată prin faptul că atunci cînd un arbitru de magistrală cîştigă MS în detrimentul altui arbitru,nu poate ocupa imediat MS trebuind să aştepte pînă cînd prezentul ocupant al MS îşi complectează ciclul de transfer (predarea MS se face făcînd BUSY/=l). Intîrzierea în propagare în cazul rezolvării paralele MULTIPROM a pri-

orităţilor la un ansamblu codificator/decodificator de tip 74148/8205 este de aproximativ 40ns,ceea ce permite ca la o frecvenţă a tactului de magistrală BCLK/ de 10MHz să coexiste un număr de 8 MP-oare master

Rezolvarea prin rotirea priorităţilor.

Tehnica MULTIPROM de rezolvare prin rotirea priorităţilor este identică cu cea prezentată în paragraful 2.4.2. Această tehnică este similară cu cea de rezolvare în paralel a priorităţilor (prezentînd aceleaşi caracteristici din punct de vedere a modului de realizare MULTIPROM) cu excepţia faptului că priorităţile sînt reasignate dinamic. Codificatorul de prioritate este înlocuit în acest caz cu un circuit mai complex ce roteşte prioritatea la intervale standard de timp (de exemplu,perioada lui BCLK/) printre arbitrii,garantînd fiecărui arbitru acces egal la MS.

Toate cele trei tehnici,prezintă avantaje şi dezavantaje. Sub raport cost/performanţe,un compromis satisfăcător îl asigură tehnica de rezolvare paralelă MULTIPROM. Aceasta deoarece permite mai multor arbitrii (între 8 şi 16) să aibă acces la MS fără să pretindă o logică extensivă pentru implementare.

6.1.3. Necesitatea arbitrării din punct de vedere a magistralei MULTIPROM.

Arhitectura unui mμP bazat pe magistrala MULTIPROM permite fiecărui MP master să lucreze asincron. Prin urmare,un MP a cărui microprocesor este mai rapid,va opera la viteza care îi este caracteristică fără să ţină cont de viteza celui mai lent MP din configuraţie. Utilizarea acestei tehnici care permite variaţii în ciclul de lucru,asigură modularitatea unui sistem mμP realizat în jurul acestei magistrale, prin faptul că atunci cînd este necesară crearea unor funcţii suplimentare ale sistemului,este posibilă adăugarea de noi module fără ca aceqsta să afecteze localizarea iniţială a proceselor în diversele MP-oare.

Magistrala MULTIPROM implementează această structură asincronă de prelucrare prin sincronizarea cererilor de magistrală emise de MP-oare cu un ceas de referinţă al magistralei a cărei frecvenţă este de pînă la 10MHz. Cererile astfel sincronizate,sînt în continuare rezolvate de un codificator de prioritate. Rezultatul imediat obţinut în acest caz,constă în obţinerea unei minimizări a circuitelor de rezolvare a priorităţilor comune tuturor MP-oare ale sistemului.

Funcţia de arbitrare şi sincronizare a cererilor de acces este in-

tegrată în sistemul mμP MULTIPROM de arbitrii de magistrală 8218/9, pentru I8o8o şi 8289 pentru I8o86/8o88. Aceştia cooperează în conjuncţie cu un controlor de magistrală care generează semnalele de citire/scriere în memorie şi la porturile de I/E pentru a interfaţa MP-oare cu MS (un exemplu în acest sens îl constituie controloarele 8289 şi 8288).

Datorită importanţei pe care o prezintă la realizarea sistemului de operare modul de cooperare dintre MP-oare,controlorii de magistrală aferenţi şi arbitrul de magistrală,ca şi datorită particularităţilor care intervin,funcţie de tipul de magistrală utilizat,acest dialog se va detalia pentru sistemul MULTIPROM în cîteva cuvinte. Un μP prezent într-un MP al sistemului mμP,trimite semnale de comandă atîta timp cît deţine controlul MS fără să ţină cont de arbitru. Dacă însă MP nu deţine controlul exclusiv al MS,arbitrul de magistrală va preveni controlorul de magistrală,amplificatoarele de date şi adrese că μP nu deţine controlul MS,fapt care are ca urmare că toate ieşirile circuitelor de acţionare a magistralei să fie blocate. Deoarece nu se trimit semnale de comandă,μP prezent în MP nu recepţionează XACK/ şi ca urmare μP intră în WAIT. Ciclul de transfer al μP-rului se extinde pînă cînd arbitrul de magistrală cîştigă accesul la MS,după care acesta permite controlorului de magistrală şi circuitelor de acţionare să aibă acces la MS. După ce controlorul emite un semnal de comandă şi are loc un transfer de date,un XACK/ este returnat μP-rului care trece la complectarea ciclului de transfer. Prin această procedură,un arbitru asigură coordonarea şi sincronizarea accesului unui MP master la MS a unui sistem mμP MULTIPROM.

6.1.4. Caracteristicile modulelor MULTIPROM ce pot deveni componente ale unor arhitecturi multimicroprocesor.

In cadrul sistemului MULTIPROM s-a realizat un ansamblu tipizat de module şi subansamble pentru echipamente numerice de automatizare MULTIPROM. Dintre aceste module,cîteva îndeplinesc condiţiile de încorporare într-o structură de sistem mμP. Acestea sînt:

- Modulul unitate centrală MPCM-01. Este un MP realizat în jurul μP-rului pe 8 biţi I8o8o,fiind compatibil cu produsul iSBC 80/20 al firmei INTEL. Modulul este construit din blocuri funcţionale organizate în jurul a trei magistrale structurate ierarhic: locală (ML),rezidentă (MR) şi sistem (MS). Fiecare magistrală poate comunica cu cea adiacentă ei şi poate opera independent de cele-

lalte două,performanţele întregului sistem μP fiind influenţate de tipul de magistrală activat la un moment dat. Modulele funcţionale sînt reprezentate de: unitatea centrală de prelucrare,memoria EPROM şi RAM,logica de decodificare a adreselor şi de formare a comenzilor,sistemul de întreruperi,ceasul de timp real,ca şi din interfeţele cu magistrală rezidentă şi cea sistem. In jurul magistralei locale ML,sînt dispuse modulele funcţionale împreună cu logica de selecţie a acestora şi de generare a comenzilor pentru coordonarea transferurilor de date. Dintre celelalte două magistrale,singura care permite obţinerea unor arhitecturi μP este magistrala sistem; magistrala rezidentă nepermiţînd operaţii multiprocesor,deoarece pe această magistrală este posibilă întotdeauna numai prezenţa unui singur MP master.
Un aspect deosebit de important,legat de utilizarea modulului MPCM-01,într-o structură μP se referă la blocarea magistralei sistem,după ce a fost cîştigat accesul (de circuitul de control a magistralei 8218),în scopul efectuării unor operaţii indivizibile de tip TAS. Modulul de blocare se asigură printr-o secvenţă de program care se efectuează de utilizator. Secvenţele corespunzătoare blocării,respectiv eliberării magistralei sistem,sînt de forma:

```
OCUP:   MVI   A,1        ELIB:   XRA   A
        ØUT   OD5H               ØUT   OD5H
```

Mai este de menţionat faptul că blocarea magistralei sistem nu se poate efectua pe o perioadă mai lungă de 12μs (aspect care constituie o caracteristică a magistralei MULTIPROM).

- Modulul de supraveghere/diagnoză MPCB-03. In cadrul unei structuri μP,joacă un rol esenţial,prin faptul că asigură rezolvarea priorităţilor şi generarea întreruperilor. Modulul dispune de un bloc de rezolvare paralelă a priorităţilor cu rolul de a alege dintre cererile emise simultan de două sau mai multe MP-oare master,pe cea mai prioritară. In urma rezolvării,pe baza unor priorităţi fixate,se garantează accesul la MS a MP master cel mai prioritar. Un rol deosebit de important îl prezintă acest modul,prin faptul că în structura sa apare un bloc destinat generării întreruperilor în întregul sistem μP. Blocul asigură atît generarea prin program a întreruperilor inter MP-oare,cît şi a cunoaşterii la solicitarea utilizatorului a întreruperii generate. Modul de generare şi recunoaştere a întreruperilor,a cărui principiu de funcţionare s-a prezentat în paragraful 2.3.1. şi detaliat în fig.

2.7,este realizat prin intermediul unui registru de generare a întreruperilor realizat cu 8282,ca şi de un registru de memorare a cererilor de întrerupere,realizat cu 74LS373. Conform acestui principiu,o cerere de întrerupere INT_i se activează prin înscrierea unui 1 în poziţia i a registrului de generare a întreruperilor,iar resetarea ei,prin înscrierea unui 0 în aceiaşi poziţie. Ca urmare,de exemplu,lansarea şi resetarea lui INT3/ se va efectua sub forma:

```
LANS:   MVI   A,8        RESET:   XRA   A
        OUT   4                   OUT   4
```

Modul de alocare a celor 8 nivele de întrerupere a modulelor procesor master din sistemul mμP se face funcţie de tipul aplicaţiei căreia îi este dedicat sistemul. Pentru ca în cursul operaţiei de lansare a unei întreruperi spre un MP al sistemului să nu apară posibilitatea pierderii unei întreruperi,s-a prevăzut un registru separat destinat memorării întreruperilor care urmează să fie rezolvate,prezenţa unei întreruperi putînd fi sesizată prin citirea acestuia. Ca urmare,cînd un MP doreşte să lanseze o întrerupere către alt MP al sistemului testează în prealabil acest registru astfel:

```
CICLU:  IN    4          ; Test dacă există o INT pentru
        CPI   0          ; MP curent
        JNZ   CICLU      ; Dacă DA se aşteaptă tratarea ei
LANS:   MVI   A,CUVINT   ; Lansare INT
        OUT   4
```

In cadrul rutinei de tratare a întreruperilor,se resetează registrul de generare a întreruperilor după cum s-a prezentat anterior. Intreruperile sînt de tipul vectorizate pe magistrală şi se tratează ca atare.

O importanţă deosebită trebuie acordată următorului aspect; întrucît modulul MPCB-03 constituie din punct de vedere al unui mμP o resursă comună,procedura de citire a registrului de memorare a întreruperilor,urmată de generarea întreruperii,trebuie să fie precedată de o secvenţă de blocare a MS,urmînd ca după efectuarea operaţiei,aceasta să fie deblocată.

- Modulul RAM-CMOS MPMM-01. Reprezintă un bloc de memorie cu o capacitate de 16ko. In cadrul unei arhitecturi mμP reprezintă un modul "slave" putînd intra în alcătuirea unuia sau a mai multor blocuri de memorie comună. Un aspect care trebuie avut în vedere

la încorporarea acestui bloc într-o structură mμP se referă la necesitatea configurării de către utilizator a adresei de început a modulului MPMM-01,astfel încît această adresă să fie diferită de spaţiul de adrese al unui alt modul de memorie "slave" dispus pe MS.

- Modulul interfaţă disc flexibil MPP-02. Modulul realizează cuplarea la MS a unui număr de 4 unităţi de disc flexibil. Este utilizat în sisteme 8/16 biţi,fiind destinat implementării sistemului RMX 80. Incorporat într-o configuraţie mμP,reprezintă un modul master de tip DMA. Un MP master care doreşte să lucreze cu MPP-02 iniţializează dialogul cu acesta prin intermediul controlerului de magistrală 8218. In continuare MPP-02 programează parametrii operaţiei (la 8237 pentru transfer DMA,iar la 8272 pentru execuţia comenzii). După calcularea parametrilor ceruţi de controlerul de disc se iniţiază cererea de DMA,iar la obţinerea MS de către MPP-02 se execută operaţia de transfer cu discul. La epuizarea operaţiei de transfer,rezultatul este comunicat prin intermediul unei întreruperi.

Concluzii

In urma analizei [**87b] efectuate asupra capacităţii modulelor MULTIPROM de a fi integrate în arhitecturi mμP,au rezultat următoarele propuneri a căror aplicare poate face viabilă construirea unor sisteme mμP. Aceste propuneri se referă la:

1. Necesitatea ca modulul procesor MPCM-01 să fie prevăzut cu logică de detectare,semnalizare şi evitare a situaţiei de time-out.

2. Pentru a micşora traficul pe magistralele sistemului mμP, modulul procesor MPCM-01 şi echivalentul lui MPCM-03,realizat cu I8086,este necesar să fie prevăzute cu memorii RAM biport.

3. Logica de arbitrare a cererilor la MS,rezidentă pe MPCB-03 este necesar să fie prevăzută şi cu facilitatea de rotire a priorităţilor pentru a acorda Mp-oare master şanse egale de acces la resursele sistemului mμP.

4. In vederea reducerii timpului necesar efectuării operaţiilor TAS,este necesară implementarea unor fanioane realizate hard şi dispuse într-un registru TAS a cărui localizare s-ar putea efectua pe MPCB-03,sau alt modul aflat obligatoriu într-o configuraţie mμP.

6.2. Stabilirea arhitecturii sistemului de operare.

La stabilirea arhitecturii sistemului de operare destinat a fi implementat pe sisteme μP realizate cu module MULTIPROM, s-au avut în vedere următoarele considerente:

- Modulele realizate în cadrul sistemului MULTIPROM nu au fost concepute în ideea de a fi utilizate la construirea unor sisteme μP. Acest aspect a impus necesitatea ca sistemul de operare să realizeze suplimentar o serie de funcţii care în mod normal ar fi trebuit să fie realizate de modulele componente ale sistemului MULTIPROM.

- Datorită gamei largi de aplicaţii posibile, a devenit necesară asigurarea unei flexibilităţi deosebite, care să dea posibilitatea talierii simple a sistemului de operare pe orice configuraţie μP, cu efectuarea unui număr mic de modificări.

- Deoarece sistemele μP din această gamă urmează să fie utilizate în aplicaţii de tip timp real, s-a impus necesitatea asigurării unui timp de aşteptare mic a cererilor de alocare a unei resurse comune (memorie comună, echipamente de I/E, magistrale) în condiţiile realizării unui timp de răspuns rapid la evenimentele externe sistemului μP.

- Ca urmare a caracterului aplicaţiilor avute în vedere a apărut necesitatea sincronizării şi coordonării activităţilor nu numai la nivelul întregului sistem μP, ci şi la nivelul fiecărui MP în parte. De asemenea, datorită cooperării strînse care apar între procesele implicate într-o aplicaţie de timp real, devine obligatorie prezenţa unui mecanism prin care procesele să poată schimba mesaje între ele. Acest mecanism trebuie să fie suficient de flexibil ca să asigure dialogul nu numai între procesele rezidente în MP-oare diferite, dar şi între procesele prezente în memoria comună sau în acelaşi MP.

Din analizele efectuate în capitolele precedente, se poate considera că, condiţiile enumerate mai sus sînt realizate în bună măsură de un sistem de operare cu nucleu monolitic. Deoarece însă codul acestuia şi structurile de date aferente sînt prezente în totalitate în memoria comună a sistemului μP, aspect care atrage după sine o creştere substanţială a taficului pe magistrala sistemului μP şi deci o mărire a timpului de răspuns a sistemului, s-a recurs la realizarea unei variante modificate. Modificarea care s-a efectuat, a constat în distribuirea prin duplicare atît a nucleului, cît şi a structurilor de da-

te în fiecare MP a sistemului. Astfel în memoria EPROM a fiecărui MP este prezent codul nucleului,iar în memoria RAM,semafoarele şi structurile de date utilizate în activitatea locală de coordonare şi sincronizare. Memoria comună a sistemului mμP în acest caz,este atît de tip RAM,conţinînd semafoarele şi structurile de date implicate în coordonarea globală a activităţilor în întreg sistemul mμP,cît şi de tip EPROM,conţinînd monitoarele şi procesele comune tuturor MP-oare din sistem. Ca un rezultat al acestei opţiuni,a apărut o variantă nouă,de tip de sistem de operare şi anume: sistem de operare mμP cu nucleu monolitic distribuit. Incorporarea unui astfel de sistem într-o arhitectură mμP realizată cu module din familia MULTIPROM,este prezentată în fig.6.1. Singurul dezavantaj constă în mărirea capacităţii de memorie EPROM,utilizată de fiecare MP în parte. Dezavantajul este compensat în parte de dimensiunea mult mai redusă a memoriei EPROM comune.

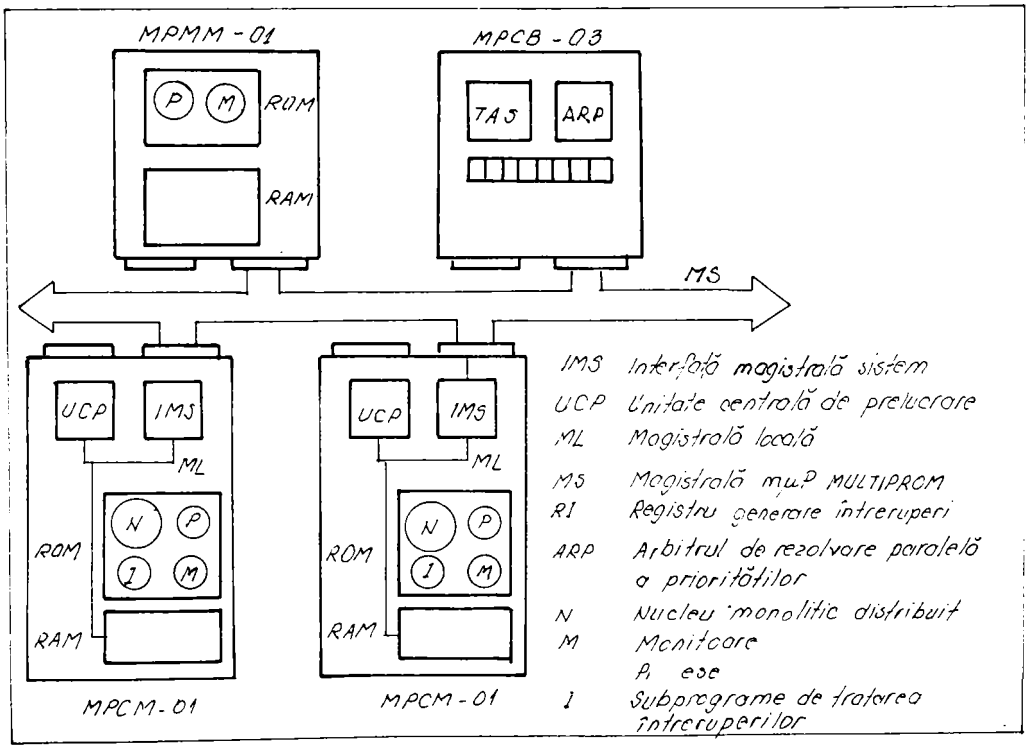


Fig.6.1. Arhitectura unui sistem mμP realizat cu module MULTIPROM şi localizarea componentelor sistemului de operare de tip nucleu monolitic distribuit.

7. IMPLEMENTAREA UNUI SISTEM DE OPERARE MULTIMICROPROCESOR DE TIP MONOLITIC DISTRIBUIT.

Pentru aplicaţiile timp real destinate sistemelor μP cele mai bune performanţe din punct de vedere al modului de organizare şi uşurinţei de implementare le prezintă un nucleu monolitic. Deoarece însă codul acestuia şi structurile de date aferente sînt prezente în totalitate în memoria comună, apar creşteri substanţiale în traficul desfăşurat pe magistralele sistemului μP care se traduc în ultimă instanţă printr-o mărire, care în multe cazuri nu poate fi acceptată, a timpului de răspuns în ansamblu al sistemului. In aceste situaţii se recurge la o modificare a nucleului care constă în distribuirea acestuia prin duplicare în fiecare din modulele procesoare aferente sistemului μP. Modificarea impune amplasarea în memoria EPROM a fiecărui MP a codului nucleului monolitic, proceselor şi monitoarelor locale, în memoria RAM locală rămînînd să fie prezente semafoarele şi structurile de date utilizate în activitatea locală de coordonare şi sincronizare. Memoria comună a sistemului μP, în acest caz, este atît de tip RAM, conţinînd semafoarele şi structurile de date implicate în coordonarea globală, cît şi de tip EPROM, conţinînd monitoarele şi eventual procesele comune tuturor MP-oare din sistem. Ca rezultat al acestei distribuiri apare o variantă nouă de tip sistem de operare μP monolitic şi anume: sistem de operare multimicroprocesor cu nucleu monolitic distribuit. Dezavantajul care în anumite cazuri poate fi substanţial (dacă sînt conectate într-o structură μP un număr mare de MP-oare), constă într-o mărire semnificativă a cantităţii de memorie EPROM utilizate. In principiu însă, acest dezavantaj este compensat printr-un trafic mult mai redus pe magistrala sistemului μP care se concretizează printr-o creştere adecvată a timpului de răspuns al sistemului, fapt care face ca în cadrul unor aplicaţii timp real μP în care numărul MP-oare din sistem nu este mare, structura de sistem de operare de tip monolitic distribuit să fie preferată. La realizarea prezentului sistem de operare μP şi îndeosebi la alegerea arhitecturii de tip nucleu monolitic distribuit, s-a ţinut cont de experienţa obţinută în scrierea unor sisteme de operare dedicate unor aplicaţii de proces [JH8o] şi reţele de calculatoare [Ho81, Sh84], cît şi de rezultatele teoretice expuse în lucrare privind alegerea arhitecturii optime a unui sistem de operare dedicat unei aplicaţii μP particulare.

7.1. Funcţiile unui sistem de operare de tip nucleu monolitic distribuit.

Un sistem de operare μP trebuie să implementeze un număr de trei funcţii, sistem care se referă la:

- sincronizare;
- comunicare;
- excludere mutuală.

Mecanismele care implementează aceste funcţii sînt ierarhizate pe două nivele. La primul nivel, cel inferior, acestea sînt responsabile cu coordonarea execuţiei proceselor, reprezentînd din acest punct de vedere mecanismele standard ale unui sistem de operare concurent. In cazul celui de al doilea nivel, funcţiile respective sînt responsabile cu coordonarea activităţii MP-oare din cadrul sistemului μP, privit de data aceasta ca un tot unitar. Intre cele două nivele există o legătură strînsă, o serie de funcţii specifice fiecărui mecanism în parte interferîndu-se cu funcţiile aceluiaşi mecanism situat la un nivel diferit. Din acest motiv, în cazul structurii adoptate şi realizate, analiza acestor mecanisme se va face începînd cu nivelul inferior responsabil cu concurenţa la nivelul unui procesor, urmînd ca în momentul analizei nucleului sistemului μP să se detalieze caracteristicile mecanismelor care implementează aceste funcţii ierarhic imediat superior.

7.2. Implementarea mecanismelor de coordonare şi sincronizare pe un sistem de calcul multimicroprocesor.

Aspectele de care trebuie ţinut cont în realizarea practică a mecanismelor de coordonare şi sincronizare pot fi rezumate în cîteva principii de bază:

- Operaţia de coordonare presupune implementarea obligatorie a două mecanisme: excludere mutuală şi sincronizarea pe condiţie.

- Excluderea mutuală presupune realizarea coordonării acceselor la o resursă comună, astfel încît ea să nu fie utilizată niciodată de mai mult de un proces la un moment dat. In acest scop excluderea mutuală trebuie să asigure accesul la o resursă comună ca şi o operaţie individuală. Secvenţa de instrucţii prin care se asigură accesul la resursa comună va defini prin urmare o regiune critică. In acest context excluderea mutuală se va referi la execuţia exclusiv separată a secţiunilor critice care se referă la aceiaşi resursă comună.

- Mecanismul sincronizării pe condiţie este destinat coordonării proceselor în cazul în care resursa comună la care procesele solicită accesul nu permite utilizarea ei. Ca urmare, mecanismul va trebui să asigure întîrzierea procesului pînă cînd starea resursei se va schimba, ca rezultat al acţiunii altor procese care execută operaţii asupra aceleiaşi resurse. După cum s-a specificat în capitolele anterioare, pentru implementarea coordonării activităţilor din sistemul μP se utilizează semaforul împreună cu mecanismele de excludere mutuală şi sincronizare pe condiţie aferente.

7.2.1. Definirea şi implementarea mecanismelor primare de sincronizare.

Mecanismele primare de sincronizare, rezolvă în principiu problema excluderii mutuale. Modul de definire şi de implementare diferă funcţie de modul de localizare a semafoarelor în memoria locală a unui modul procesor sau în memoria comună a sistemului μP.

Dacă semaforul este prezent în memoria locală a unui MP, fiind utilizat pentru sincronizarea proceselor prezente în memoria acestuia, tehnica utilizată în prezent, şi cea care s-a utilizat în cadrul prezentului sistem de operare, constă în dezactivarea întreruperilor. Conform acestei tehnici orice apel provenit dintr-un proces local adresat nucleului monolitic local modulului procesor, duce la dezactivarea întreruperilor. Ca urmare secvenţele de program care aparţin monitorului monolitic se execută într-un regim special cu întreruperile dezautorizate, fiind ferite astfel de posibilitatea de a fi întrerupte, aspect care conferă nucleului în întregime statutul de regiune critică. Reautorizarea întreruperilor este efectuată imediat ce se revine din nou într-un proces utilizator. Metoda dezautărizării şi autorizării întreruperilor este simplă şi avantajoasă, asigurîndu-se implementarea directă a excluderii mutuale şi realizarea indirectă a incronizării pe condiţie.

Dacă semaforul este prezent în memoria comună a sistemului μP, fiind utilizat pentru sincronizarea proceselor prezente în această memorie, metoda autorizării şi dezautorizării întreruperilor devine neoperaţională. Metoda utilizată în prezent constă în definirea unui mecanism TAS. Mecanismul al cărui principiu de funcţionare s-a prezentat în capitolele precedente, presupune atașarea unei variabile TAS la fiecare zonă partajată a memoriei în care este prezentă resursa asupra căreia se cer a fi efectuate operaţii de două sau mai multe MP-oare ale sistemului.

In cazul sistemului de operare realizat,datorită existenţei unei singure memorii comune,care se poate atribui la un moment dat unui singur MP al sistemului,s-a utilizat o singură variabilă TAS care protejează întreaga memorie comună. Variabila care este poziţionată de nucleul MP care a cîştigat magistrala este identică cu cea utilizată pentru blocarea şi deblocarea magistralei sistemului μP. Mecanismul implementat este de forma:

```
INLINE ($F3);                          (* DI  *)
1: IF FLAG = FALSE THEN
                BEGIN
                  FLAG:=TRUE;
                  INLINE ($FB);        (* EI *)
                END
      ELSE GOTO 1;
FLAG:= FALSE;
```

Mecanismul implementat este simplu avînd însă dezavantajul ca MP care a cîştigat accesul la magistrală intră într-un singur regim de testare repetată pînă în momentul în care variabila este resetată de MP care a ocupat anterior memoria comună.

7.2.1. Detalii de implementare a mecanismelor de coordonare. Primitiva de coordonare.

Mecanismul de coordonare utilizat în prezent în sistemele μP se bazează pe conceptul de semafor aşa cum s-a definit în capitolele anterioare. Acelaşi concept s-a utilizat şi în cadrul sistemului de operare μP realizat. Operaţiile asupra semafoarelor sînt realizate de două primitive care aparţin nucleului monolitic distribuit. Cele două primitive P şi V al căror mod de operare asupra unui semafor este cel clasic,sînt definite în cele ce urmează.

Primitiva P este implementată printr-o procedură cu acelaşi nume,realizînd suspendarea unui proces în şirul de aşteptare al unui semafor,pe de o parte,iar pe de altă parte,încercînd lansarea în execuţie a următorului proces din şirul Gata De Execuţie (GDE),ataşat MP curent,dacă contorul semaforului după decrementare este negativ. In cadrul acestei proceduri se acţionează asupra contorului BCPS[RUN].Q care este trecut pe 1 pentru a semnala că procesul a fost pus în aşteptare. Introducerea în şirul de aşteptare al semaforului se face în ordinea FIFO (se justifică astfel prezenţa cîmpului SF la un semafor care are rolul de a păstra ultimul nod al listei).

Primitiva implementată este de forma:

```
PROCEDURE P(VAR SEM: SEMAFOR);

(* P-REALIZEAZA DECREMENTAREA CONTORULUI SEMAFORULUI SEM.COUNT,
   PUNEREA IN ASTEPTARE A PROCESULUI CURENT SI LANSAREA IN
   EXECUTIE A URMATORULUI PROCES DIN GDE DACA CONTORUL ESTE
   NEGATIV *)
     VAR REV: INTEGER;
      C: CHAR;
BEGIN
 SEM.COUNT:= SEM.COUNT-1;
 IF SEM.COUNT<0 THEN
 BEGIN
   BCPS[RUN].Q:= 1;
   WITH SEM DO
   BEGIN
    IF (SF<>0) AND (SIR.INCEP<>0) THEN
      SIR.ZONA[SF].URM:= SIR.DISPON
    ELSE SIR.INCEP:=SIR.DISPON;
    SF:= SIR.DISPON;
    SIR.DISPON:= SIR.ZONA[SIR.DISPON].URM;
    SIR.ZONA[SF].NOD:= RUN;
    SIR.ZONA[SF].URM:= 0;
   END;
   RUN:= SUPRIM(RTR);
 END
END;
```

Primitiva V este implementată printr-o procedură cu acelaşi nume realizînd incrementarea contorului unui semafor. In cazul în care valoarea acestuia este mai mică sau egală cu zero se scoate primul proces din şirul de aşteptare al semaforului şi se inserează în lista GDE a UP curent în ordinea descrescătoare a priorităţii. Primitiva implementată este de forma:

```
PROCEDURE V(VAR SEM: SEMAFOR);

(* V-REALIZEAZA INCREMENTAREA CONTORULUI SEMAFORULUI SEM.COUNT,
   SUPRIMAREA PRIMULUI PROCES DIN SIRUL SEMAFORULUI SI INSERAREA
   LUI IN SIRUL GDE DACA CONTORUL NU ESTE POZITIV *)
VAR INDEX: INTEGER;
BEGIN
 SEM.COUNT:= SEM.COUNT + 1;
```

```
    IF SEM.COUNT <= 0 THEN
     BEGIN
     INDEX:= SUPRIM(SEM.SIR);
     INSERT(RTR,INDEX);
     END
   END;
```

Modul de implementare a celor două primitive responsabile de operaţia de coordonare este diferit funcţie de modul de localizare al semafoarelor asupra cărora operează,în memoria proprie a MP-oare execută procedurile asociate celor două primitive sau în memoria comună a sistemului multimicroprocesor. Ca urmare apar două tipuri de mecanisme:

- un mecanism responsabil de coordonarea locală a activităţilor în cadrul unui MP;
- un mecanism responsabil cu coordonarea activităţilor la nivelul întregului sistem multimicroprocesor.

Mecanism de coordonare locală

La nivelul unui MP mecanismul de coordonare bazat pe semafoare, utilizează cele două primitive P şi V în modul specificat mai sus. Problema esenţială care s-a pus în acest caz,a fost legată de necesitatea asigurării indivizibilităţii execuţiei primitivelor P şi V. Mecanismul care s-a implementat a constat în asigurarea statutului de regiune critică procedurilor care implementează cele două primitive,prin dezautorizarea întreruperilor în momentul lansării lor în execuţie şi respectiv,reautorizarea acestora în momentul terminării operaţiilor efectuate de aceste primitive pe semaforul asupra căruia au acţionat. In acest mod,s-a sigurat coordonarea proceselor care aparţin unui MP din sistemul mμP.

Mecanism de coordonare globală

Coordonarea execuţiei proceselor prezente în memoriile locale ale diferitelor MP-oare care fac parte din sistemul mμP,implică utilizarea aceloraşi primitive P şi V,cu observaţia că de data aceasta semafoarele asupra cărora se acţionează sînt prezente în memoria comună a sistemului (motiv pentru care sînt cunoscute sub denumirea de semafoare globale). La implementarea mecanismului de coordonare global s-au avut în vedere următoarele aspecte:

- Pentru asigurarea statutului de regiune critică procedurilor care implementează cele două primitive s-a procedat la dezautorizarea întreruperilor MP care a cîştigat accesul la memoria comună

reautorizarea acestora efectuîndu-se în momentul în care s-a terminat acţiunea asupra semaforului global;

- Deoarece la un moment dat mai multe MP-oare pot solicita accesul la memoria comună prin intermediul primitivelor P şi V,a devenit necesară utilizarea unui mecanism de excludere mutuală care s-a realizat printr-o funcţie TAS. Variabila TAS utilizată este aceiaşi cu cea utilizată pentru protecţia memoriei comune.

In ceea ce priveşte funcţiile primitivei P aplicată unui semafor global,acestea coincid cu cele ale primitivei P utilizată într-un mecanism de coordonare local. Deoarece o primitivă P are efect numai asupra procesului care o apelează,rezultă că primitiva P aplicată unui semafor global realizează actualizările de stare în structura de date proprie nucleului monolitic distribuit al MP pe care se execută,şi înlănţuirile semaforului global prezent în memoria comună.

Implementarea primitivei V este mult mai complexă deoarece execuţia ei asupra unui semafor global,are drept consecinţă eliberarea unui proces care aparţine unui MP diferit de cel care a executat operaţia. Ca urmare,actualizarea informaţiei privitoare la starea procesului eliberat nu poate fi efectuată de procedurile nucleului monolitic care asigură execuţia primitivei V,deoarece acesta nu are acces la memoria altui MP. Astfel a devenit necesar ca primitiva V să semnaleze modulului procesor căruia îi aparţine procesul,că acesta a fost scos din şirul de aşteptare al unui semafor global împreună cu informaţia necesară identificării procesului. Semnalarea este efectuată prin intermediul sistemului de întreruperi. Astfel,terminarea execuţiei primitivei se încheie cu lansarea unei întreruperi adresată MP căruia îi aparţine procesul eliberat. In urma tratării întreruperii de către nucleul monolitic prezent în respectivul MP se activează procedurile pentru actualizarea stării procesului eliberat. Transmiterea informaţiei privind identitatea procesului eliberat se face printr-un mecanism de tip cutie poştală.

La implementarea celor două primitive,s-a urmărit ca acestea să aibă acelaşi mod de utilizare,indiferent dacă semaforul asupra căruia acţionează este un semafor local unui MP sau este un semafor global prezent în memoria comună.

7.3. Detalii de implementare a cooperării prin mesaje.

Cooperarea prin intermediul mesajelor reprezintă în cadrul unui sistem de operare multiP forma cea mai înaltă de interacţiune între procesele prezente. După cum s-a arătat mesajul îşi are originea în con-

ceptul de semafor al cărui mecanism de sincronizare a fost completat cu un mecanism de transmitere a datelor. Astfel mesajul este o structură mai complexă,care prin operaţii specifice,permite atît transmiterea unei informaţii între procesele concurente prezente în acelaşi MP sau între MP-oare diferite,cît şi sincronizarea execuţiei acestora. Cooperarea în acest caz este strîns legată de relaţia producător/consumator.

Pentru a asigura un caracter unitar modului de cooperare prin mesaje,în cazul prezentului sistem de operare mμP,s-a adoptat soluţia utilizării unui mecanism de tip monitor pentru coordonarea interacţiunii proceselor care îşi transmit reciproc mesaje. Monitorul ales într-o primă versiune este de tip MONITOR,deoarece prezintă cel mai simplu şi mai rapid mecanism de coordonare în cazul unui număr relativ mic de procese.

7.3.1. Definirea structurilor de date şi a primitivelor monitorului.

Structura care s-a adoptat a constat dintr-o zonă de tip cutie poştală prin intermediul căreia se efectuează schimbul de mesaje şi a cărei dimensiune se stabileşte la implementarea sistemului de operare. In cadrul sistemului aceste mărimi sînt definite astfel:

```
CONST MAX= 4;  (* NR MAXIM DE PROCESE *)
      MAXT=2;  (* NR MAXIM DE TAMPOANE *)
TYPE  QS = 1;..MAXT;
```

Zona este împărţită în două sau mai multe subzone (tampoane) de lungime fixă,lungimea unei subzone fiind dată de mărimea maximă a unui mesaj care poate fi schimbat între două procese. Protecţia zonei este asigurată de un semafor. Pentru sincronizarea accesului proceselor la zonele care conţin mesaje s-au prevăzut alte două semafoare, unul pentru sincronizarea accesului la zonele care conţin mesajele care urmează să fie preluate,iar celălalt pentru sincronizarea accesului la zonele care sînt libere şi deci pot fi complectate cu un nou mesaj. Sirurile de aşteptare ale celor două semafoare vor conţine adresele zonelor completate,respectiv adresele zonelor libere. Intreaga structură de date: zone care conţin mesaje plus semafoarele aferente reprezintă cutia poştală prin care două procese implicate într-o relaţie producător/consumator îşi transmit reciproc mesaje. Aspectul unei cutii poştale în prezentul sistem de operare este deci de forma:

```
BUF = ARRAY[1..80] OF CHAR;
WQ = RECORD
       COUNT: 0..MAXT;
       QAREA: ARRAY[1..MAXT] OF INTEGER
     END;
COND = RECORD
         COUNT: INTEGER;
         SEM: SEMAFOR
       END;
STIVA = ARRAY [1..512] OF BYTE;
         END;
VAR WWORK,WRET: COND;
    GATE: SEMAFOR;
    FLAG: BOOLEAN;
    WORK,RETURN:WQ;
```

Structura unei astfel de cutii poştale s-a realizat pentru un număr de una sau mai multe perechi de procese implicate într-o relaţie producător/consumator.

Utilizînd această structură de date,implementarea unui monitor de tip MONITOR a condus la definirea unui set de primitive ale cărui proceduri au următorul aspect.

```
PROCEDURE DELAY(VAR C: COND);
(* DELAY-REALIZEAZA INCREMENTAREA CONTORULUI VARIABILEI DE CONDITIE
   SI INCEARCA SUSPENDAREA PROCESULUI CURENT IN SIRUL DE ASTEPTARE
   AL SEMAFORULUI VARIABILEI C *)
BEGIN
 C.COUNT:= C.COUNT+1;
 NUCLEU ('PV',GATE,C.SEM);
END;

PROCEDURE CONT (VAR C: COND);
(* CONT-DACA CONTORUL VARIABILEI C ESTE NENUL,SE DECREMENTEAZA ACEST
   CONTOR SI SE SCOATE PRIMUL PROCES DIN SIRUL SEMAFORULUI CORES-
   PUNZATOR *)
BEGIN
 IF C.COUNT<>0 THEN
 BEGIN
  NUCLEU ('V',C.SEM,C.SEM);
  C.COUNT:= C.COUNT-1;
 END
END;
```

```
  PROCEDURE ENTER;
(* ENTER-ASIGURA INTRAREA IN MONITOR *)
BEGIN
 NUCLEU ('P',GATE,GATE);
END;
  PROCEDURE EXIT;
(* EXIT-ASIGURA IESIREA DIN MONITOR *)
BEGIN
 NUCLEU (' V' ,GATE,GATE);
END;
```

7.3.2. Implementarea mecanismului de cooperare prin mesaje.

Utilizînd structura de cutie poştală şi primitivele mecanismului de cooperare realizate de primitivele monitorului implementat,a devenit posibilă definirea practică a unui mecanism de cooperare pentru un sistem de procese bazat pe transmiterea mesajelor. Cele două primitive realizate sînt SEND şi RECEIVE.

Primitiva SEND este implementată printr-o procedură cu acelaşi nume,avînd rolul de a completa o zonă tampon. Procedura asigură,în cazul în care nu mai sînt prezente zone tampon libere,punerea în aşteptare a procesului activ care a apelat primitiva printr-o procedură DELAY (aceasta efectuează o operaţie V asupra semaforului de intrare în monitor - GATE şi o operaţie P asupra semaforului de condiţie WRET - procese care aşteaptă o zonă tampon liberă). Cînd procesul suspendat este relansat (operaţie care se face prin execuţia unei primitive P asupra semaforului de condiţie WWORK -procese care aşteaptă o zonă tampon completată,apelată din DELAY),se va verifica dacă procesul a fost sau nu pus în aşteptare. Dacă acesta a fost suspendat se revine la testarea contorului de zone tampon libere. In cazul în care acest test nu s-ar efectua,procesul poate termina execuţia în monitor fără să fi completat nici o zonă tampon. Primitiva implementată este de forma:

```
  PROCEDURE SEND
(* SEND-REALIZEAZA TRANSMITEREA UNUI MESAJ IN TAMPONUL INDICAT
   PRIN POINT *)
VAR I,POINT:INTEGER;
```

```
LABEL 2;
BEGIN
 2: ENTER;
    IF RETURN.COUNT=0 THEN
    BEGIN
      DELAY (WRET);
      GOTO 2;
    END
    ELSE
    BEGIN
    POINT:= RETURN.QAREA [RETURN.COUNT] ;
    RETURN.COUNT:=RETURN.COUNT-1;
    WRITELN ('PROCES NR:',RUN);
     WRITELN ('INTRODUC DATE IN TAMPONUL NR:',POINT);
     WBUF [POINT,20 ]:= CHR (RUN+48);
     WBUF [POINT,53 ]:= CHR (POINT+48);
     WORK.COUNT:=WORK.COUNT+1;
     WORK.QAREA [WORK.COUNT] := POINT;
     CONT (WWORK);
     EXIT;
    END;
END;
```

Primitiva RECEIVE este implementată printr-o procedură cu același nume având rolul de a elibera o zonă tampon. Modul ei de funcționare reprezintă de fapt complementul primitivei SEND prin faptul că acționează asupra unei zone tampon completate. Primitiva implementată este de forma:

```
  PROCEDURE RECEIVE;
(* RECEIVE-REALIZEAZA RECEPTIONAREA MESAJULUI DIN TAMPONUL INDICAT
   PRIN TAMP *)
VAR T,J: INTEGER;
LABEL 3;
BEGIN
3: ENTER;
   IF WORK.COUNT=0 THEN
   BEGIN
     DELAY (WWORK);
     GOTO 3;
   END
   ELSE
```

```
      BEGIN
        TAMP:= WORK.QAREA [ WORK.COUNT ] ;
        WORK.COUNT:= WORK.COUNT-1;
        WRITELN ('PROCES NR:',RUN);
        WRITELN ('CONTINUTUL TAMPONULUI NR:',TAMP);
        FOR J:= 1 TO 80 DO
         WRITE (WBUF [TAMP,J] );
        WRITELN;
        RETURN.COUNT:= RETURN.COUNT+1;
        RETURN.QAREA [ RETURN.COUNT ] := TAMP;
        CONT (WRET);
        EXIT;
      END;
    END;
```

Ambele primitive astfel definite fac parte integrantă din nucleul monolitic al sistemului de operare muP fiind prezente în fiecare din MP-oare ale sistemului. In momentul apelării de către un proces, cele două primitive determină numărul zonei tampon asupra cărora vor acţiona,reprezentînd practic interfaţa dintre monitor şi procese.

Implementarea macanismului de intercomunicare utilizînd cele două primitive s-a efectuat ţinînd cont de două situaţii care pot apare

- Mecanismul este destinat intercomunicaţiei dintre procese care sînt locale unui MP. In acest caz,mecanismul de intercomunicare a impus definirea structurilor de date de tip cutie poştală şi a monitorului în memoria locală a MP. Utilizînd tehnica reţelelor de evaluare [**72] în fig.7.1.,s-a ilustrat modul în care funcţionează mecanismul de coordonare a proceselor pe un semafor S la nivelul unui singur MP. Din figură se observă că disciplina de planificare a proceselor în şirul de aşteptare al semaforului este de tip FIFO.

- Mecanismul este destinat schimbului de mesaje între procese care aparţin unor MP-oare diferite. Datorită caracterului distribuit al nucleului monolitic al sistemului de operare a fost posibilă simplificarea considerabilă a mecanismului de intercomunicare. Aceasta se datoreşte faptului că în memoria comună sînt prezente numai structurile de date comune proceselor prezente în MP-oare diferite, fără să fie necesară şi prezenţa codului monitorului în această memorie comună,ca şi în cazul în care s-ar fi recurs la o structură de nucleu monolitic localizat în memoria unui singur MP al sistemului multimicroprocesor,fig.7.2.

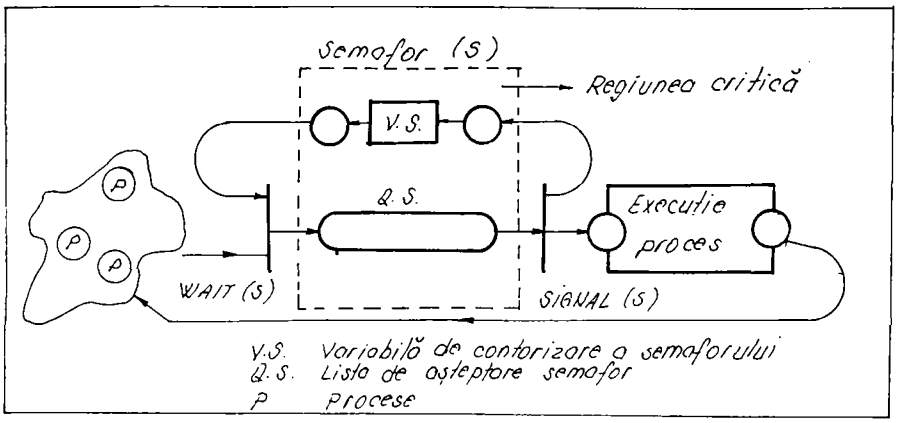


Fig.7.1. Mecanismul de coordonare a execuţiei proceselor care sînt locale unui MP.

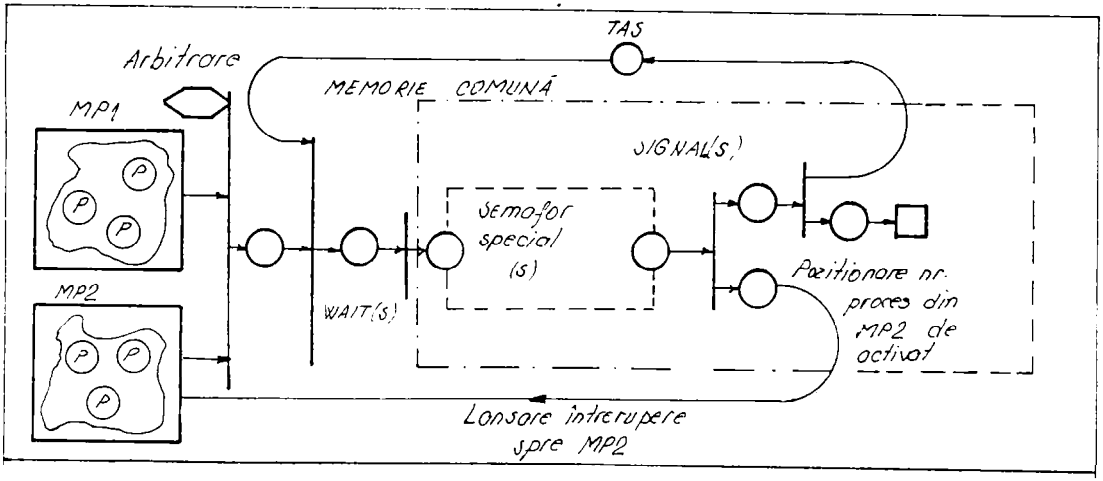


Fig.7.2. Mecanismul de coordonare globală a execuţiei proceselor care sînt prezente în diferitele MP-oare ale sistemului mpP. Situaţia corespunde cazului în care un proces prezent în MP_1 cere lansarea în execuţie a unui proces prezent în MP_2.

Definirea primitivelor pentru controlul proceselor de tip STI

Pentru controlul proceselor de tip subprograme de tratare a întreruperilor,în cadrul prezentului sistem de operare s-a procedat la utilizarea aceloraşi tipuri de primitive ca cele folosite pentru coordonarea execuţiei proceselor obişnuite.

Datorită caracterului distribuit a nucleului monolitic al sistemului de operare (care este practic prezent în fiecare MP al sistemului mμP) se asigură un grad mare de independenţă a fiecărui MP component. Un rezultat direct al acestei stări de fapt,timpul necesar executării regiunilor critice de cod (pentru care sistemul de întreruperi este dezautorizat),este mult mai mic decît în cazul în care sistemul de operare ar fi prezent în memoria comună a sistemului mμP. Acest aspect a făcut posibilă aplicarea unei proceduri simple de tratare a proceselor de tip STI,care este inoperantă în cazul sistemelor de operare mμP localizate în memoria comună a sistemului. Procedura constă în faptul că procesele sînt executate cu sistemul de întreruperi dezactivat,aspect care face ca un proces de tip STI să nu fie nevoit să aştepte înainte de a avea accesul la o regiune critică. Singurul dezavantaj al metodei,constă în faptul că procesele de tip STI trebuie să fie localizate în memoria proprie a fiecărui MP, deoarece prezenţa lor în memoria comună,conduce la apariţia unui timp suplimentar legat de fenomenul de aşteptare la cîştigarea magistralei. In anumite aplicaţii,această metodă de rezolvare face necesară duplicarea aceluiaşi proces de tip STI în două sau mai multe MP-oare,aspect care constituie de asemenea un dezavantaj. Singura primitivă utilizată de un proces de tip STI care s-a implementat,are rolul de a stopa execuţia procesului curent,care a fost întrerupt în momentul lansării în execuţie a procesului de tip STI şi de a emite lansarea în execuţie a altui proces,după terminarea execuţiei procesului de tip STI în cauză. Primitiva implementată este PREEMPT.

7.4. Detalii de implementare a nucleului sistemului de operare multimicroprocesor.

Nucleul reprezintă o regiune critică care controlează funcţiile sistemului legate de fenomenul de concurenţă. Conform acestei definiţii,nucleul este o regiune critică de cod ca şi un monitor reprezentînd astfel din punct de vedere logic monitorul fundamental al sistemului. Orice activitate în care este cerută schimbarea stării unui proces,va implica în mod obligatoriu nucleul. Această cerinţă implică şi justifică faptul că însuşi nucleul este la rîndul lui o regiune critică. Pe lîngă această funcţie,într-un sistem mμP nucleul trebuie să gestioneze structurile de date care identifică procesoarele în care se vor executa numai anumite procese şi să asigure în

fapt izolarea structurii hard de structura logică a sistemului.

Pentru a putea gestiona şi cotrola concurenţa dintre procese şi monitoare,în contextul unei anumite structuri hard,o serie de aspecte legate de modul de acces în monitor,de execuţia unui proces ca şi de asigurarea unei structuri optime a sistemului de operare,trebuie să îşi găsească răspuns în cadrul fiecărui nucleu.

Din punct de vedere logic,nucleul unui sistem de operare multimicroprocesor este divizat în trei părţi:

- nucleul principal;
- primitive şi structuri de date;
- interfaţa de intrare/ieşire din nucleu.

Din punct de vedere structural,nucleul apare ca o procedură care este apelată de procese şi monitoare în care primitivele solicitate sînt definite de către parametrii.

7.4.1. Definirea structurilor de date utilizate de nucleu.

Principalele structuri de date utilizate de nucleu cît şi modul în care acestea s-au definit în prezentul sistem de operare muP apar după cum urmează.

Stive

Se rezervă cîte una pentru fiecare proces,stiva nucleului fiind identică cu cea a programului principal. Pentru implementarea prezentă pentru un număr de patru procese,declararea stivei se face sub forma:

```
TYPE  STIVA = ARRAY [1..512] OF BYTE;
      STIV4: STIVA;
      STIV3: STIVA;
      STIV2: STIVA;
      STIV1: STIVA;
```

Semafoare

Se defineşte pentru fiecare variabilă de condiţie utilizată cît şi pentru fiecare din stările unui monitor. Structura unui semafor este de forma:

```
  LISTA = RECORD
(* LISTA-IMPLEMENTATA CA O LISTA INLANTUITA UTILIZIND
   STRUCTURA DE TABLOU *)
            ZONA: ARRAY [1..MAX] OF ART;
            DISPON,INCEP: INTEGER
(* DISPON-INDICA SPRE PRIMUL NOD AL LISTEI LIBERE
```

```
        INCEP-INDICA SPRE PRIMUL NOD AL LISTEI OCUPATE *)
              END;
      SEMAFOR = RECORD
                  COUNT: INTEGER;
(*  COUNT-CONTORUL SEMAFORULUI  *)
                  SF: INTEGER;
(*  SF-INDICA SPRE ULTIMUL NOD AL LISTEI OCUPATE  *)
                  SIR: LISTA
(*  SIR-REPREZINTA SIRUL DE ASTEPTARE AL SEMAFORULUI CARE
    UTILIZEZA DISCIPLINA DE PLANIFICARE FIFO  *)
                END;
```

Blocuri de control ale proceselor

La nivelul nucleului fiecare proces este identificat printr-un Bloc de Control al Procesului (BCP) care să conţină suficiente informaţii pentru a permite punerea în aşteptare sau restaurarea procesului din orice şir de aşteptare în care acesta a fost suspendat. Pentru prezenta realizare BCP are următorul aspect:

```
      BCP = RECORD
(*  BCP-BLOCUL DE CONTROL AL PROCESULUI  *)
            NUME: ARRAY [1..8] OF CHAR;
            Q: INTEGER;
(*  Q-VARIABILA CARE INDICA DACA PROCESUL ESTE LANSAT IN EXECU-
    TIE PENTRU PRIMA DATA (0),SAU A MAI FOST PUS IN ASTEPTARE
    (1)  *)
            PC: INTEGER;
(*  PC-CONTINE ADRESA DE LA CARE SE EXECUTA UN PROCES  *)
            ADRST: INTEGER;
(*  ADRST-CONTINE INDICATORUL DE STIVA A PROCESULUI  *)
            PRIOR: INTEGER;
(*  PRIOR-CONTINE PRIORITATEA PROCESULUI  *)
          END;
```

Siruri de aşteptare ale proceselor care solicită lansarea în execuţie

Pentru fiecare MP în parte se defineşte cîte un şir de aşteptare care conţine procesele,identificate prin BCP aferente,ce sînt gata de execuţie. In literatură,şirurile de acest tip sînt cunoscute sub denumirea de RTR (Ready To Run). In prezentul sistem de operare multimicroprocesor, un astfel de şir este definit sub forma:

```
        RUN,PM: INTEGER;
        RTR: LISTA;
(*RTR-REPREZINTA LISTA PROCESELOR GATA DE EXECUTIE.
  LISTA ESTE ORDONATA IN SENS DESCRESCATOR AL PRIORITATII  *)
```

7.4.2. Detalii de implementare a interfeţei de intrare şi ieşire a nucleului.

Interfaţa de intrare/ieşire a nucleului apare ca parte constitutivă a acestuia,reprezentînd interfaţa cu limbajul de nivel înalt în care sînt scrise procesele şi respectiv nucleul. Ambele interfeţe trebuie să apară structurate astfel ca apelarea nucleului să apară sub forma apelului unui subprogram sau proceduri cu parametrii,a cărei sintaxă.să fie conformă cu cea a limbajului în care este scrisă procedura (proces,monitor) din care se efectuează apelul. Acest aspect asigură transparenţa nucleului faţă de limbajele în care sînt scrise procedurile din care este chemat.

Interfaţa de intrare

Realizarea depinde de modul de localizare a nucleului,fie în memoria locală a unui MP destinat exclusiv realizării funcţiilor acestuia, fie în memoria comună,caz în care funcţiile sînt realizate de MP-oare care solicită accesul în nucleu.

In cazul prezentei implementări,datorită caracterului distribui al nucleului,interfaţa de intrare trebuie să realizeze următoarele:

- Dezactivarea întreruperilor MP care a cerut execuţia unei primitive a nucleului,deoarece nucleul reprezintă o regiune critică protejată la întreruperile care pot apare de la diversele echipamente de I/E aparţinînd MP în cauză. Dezactivarea întreruperilor este urmată de salvarea registrelor procesului,deoarece parametrii funcţiei apelate sînt transferaţi prin intermediul lor,cît şi de operaţia de comutare pe stiva nucleului. Operaţia de comutare a stivei este realizată de către procedura SALV. Aceasta realizează în limbaj de asamblare [##86b]comutarea stivei,impunînd încărcarea pointerului noii stive în variabila ADSP. După comutarea stivei la reîntoarcerea din SALV,în ADSP va fi prezent pointerul stivei vechi care se va memora după caz fie în variabila MEM,fie în cîmpul ADRST,dacă este vorba de comutare de stivă pentru un proces. Procedura implementată este:

```
PROCEDURE SALV;
(*  SALV-SALVEAZA POINTERUL STIVEI CURENTE SI COMUTAREA ACESTUIA
```

```
    PE ZONA INDICATA DE VARIABILA ADSP.VALOAREA SALVATA ESTE
     MEMORATA TOT IN VARIABILA ADSP  *)
BEGIN
 INLINE ($C1/          (* POP     B     *)
        $2A/ZERO/      (* LHLD    ZERO  *)
        $39/           (* DAD     SP    *)
        $EB/           (* XCHG          *)
        $2A/ADSP/      (*LHLD    ADSP  *)
        $F9/           (* SPHL          *)
        $C5/           (* PUSH B        *)
        $EB/           (* XCHG          *)
        $22/ADSP);     (* SHLD          *)
END;
```

- Deoarece nucleul este o regiune critică interfaţa trebuie să asigure apelul propriu-zis al acestuia,care din punct de vedere logic este similar cu accesul în monitor. Datorită caracterului distribuit al implementării realizate dispare necesitatea arbitrării cererilor de acces în nucleu provenite de la alte MP-oare ale sistemului mP deoarece aceste cereri,care se traduc prin apariţia unor întreruperi,sînt evaluate implicit de sistemul de întreruperi al fiecărui MP în parte (întreruperile în acest caz au rolul numai de a semnala apariţia unei modificări în zona de date comune ce interesează MP în cauză -în ultimă instanţă modificarea unui semafor global). Este evident însă că se impune un procedeu de tip TAS pentru a proteja nucleul faţă de cererile simultane provenite de la procesele locale MP în cauză. Se observă că în prezenta realizare,asigurarea statutului de regiune critică pentru nucleu este realizată la nivelul fiecărui MP printr-o astfel de procedură,aspect care impune utilizarea metodei aşteptării active a cererilor locale în faţa intrării nucleului. Soluţia aleasă reprezintă o soluţie optimă deoarece întîrzierile introduse în execuţia primitivei solicitate sînt nesemnificative.

Interfaţa de ieşire

Asigură eliberarea nucleului,operaţie care se realizează prin:

- resetarea dispozitivului TAS;
- autorizarea întreruperilor şi restaurarea registrelor.

7.4.3. Detalii de implementare a nucleului principal.

Datorită aspectului distribuit al soluţiei de implementare ale-

se,nucleul este prezent în fiecare MP al sistemului,fiind localizat între cele două interfeţe descrise anterior. Atribuţiile acestuia se referă la:

- funcţie de informaţiile transmise determină tipul primitivei solicitate;
- lansează în execuţie procedura corespunzătoare primitivei solicitate.

Primitivele nucleului

Procedurile care implementează primitivele unui nucleu reprezintă componenta structurală şi funcţională cea mai importantă a unui sistem de operare multimicroprocesor. Funcţie de finalitatea funcţională şi structurală pe care o prezintă în cadrul unui nucleu apar următoarele clase de primitive responsabile cu:

- controlul execuţiei unui proces;
- intercomunicarea dintre procese;
- iniţializarea nucleului;
- planificarea proceselor care se execută.

In sistemul de operare realizat,nucleul principal este definit ca o procedură cu numele NUCLEU care are următorii parametrii:

- APEL pentru transmiterea numelui primitivei ce se doreşte a fi apelată de către procedura nucleu. Aceste primitive pot fi:'P',' V', 'PV'. Se face specificarea că dacă la apel se va specifica 'P'sau 'V'dec se cere executarea unei singure proceduri semafoarele se vor dubla,în sensul că se transmite acelaşi semafor de două ori; dacă însă se specifică 'PV',se va realiza mai întîi operaţia 'V'asupra semaforului P1, după care se va lansa o operaţie asupra semaforului P2 (realizîndu-se astfel o operaţie echivalentă lui WAIT-AND-SIGNAL).

Procedura implementată în care se observă interfaţa de intrare şi cea de ieşire care delimitează operaţiile de determinare a tipului primitivei solicitate şi de lansare în execuţie are următorul aspect:

```
PROCEDURE NUCLEU (APEL: SR; VAR P1,P2: SEMAFOR);
(* NUCLEU-REALIZEAZA COMUTAREA PE STIVA NUCLEULUI,APELUL PRIMITI-
   VEI SPECIFICATE PRIN PARAMETRUL APEL,REVENIREA PE STIVA
   PROCESULUI CURENT SI RELANSAREA ACESTUIA.PENTRU ASIGURAREA
   ACCESULUI IN REGIM DE EXCLUDERE RECIPROCA LA STRUCTURILE DE
   DATE DIN MEMORIA COMUNA S-A FOLOSIT FANIONUL FLAG *)
LABEL 1;
VAR AREV:INTEGER;
BEGIN
INLINE (*F3);                    (* DI        *)
```

```
1: IF FLAG = FALSE THEN
                BEGIN
                  FLAG: = TRUE;
                  INLINE ($FB);          (* EI          *)
                END
     ELSE GOTO 1;
  INLINE ($E1/                           (* POP   H     *)
          $22/AREV);                     (* SHLD AREV   *)
  ADSP:= MEM;
  SALV;
  BCPS [RUN] .ADRST:= ADSP;
  BCPS [RUN] .PC:= AREV;
  IF APEL='P'THEN P(P1)
  ELSE
   IF APEL='V'THEN V(P1)
    ELSE
     IF APEL='PV'THEN
                  BEGIN
                    V(P1); P(P2);
                  END;
ADSP:= BCPS [RUN] .ADRST;
SALV;
MEM:= ADSP;
AREV:= BCPS [RUN] .PC;
FLAG:= FALSE;
INLINE ($2A/AREV/                        (* LHLD AREV   *)
        $E5);                            (* PUSH H      *)
 END;
```

Primitive care controlează execuţia proceselor

Sînt cele care s-au discutat anterior: P,V şi PV. Acestea s-au definit în urma construirii automatului cu stări finite care specifică stările nucleului şi modul în care se efectuează tranziţia dintr-o stare în alta. Automatul se construieşte plecînd de la automatul care defineşte modul de execuţie al unui proces la care se adaugă un număr de stări suplimentare. Stările automatului sînt: EXECUTIE,ASTEPTARE PE SEMAFOR,BLOCAT şi GATA DE EXECUTIE. Evenimentele care cauzează trecerea dintr-o stare în alta,vor defini setul de primitive responsabile cu stările unui nucleu,acestea fiind cele specificate anterior.

La acest set de primitive se mai adaugă o procedură cu numele LANS, avînd rolul de lansare în execuţie a primului proces din şirul GDE. Procedura dispune de următorii parametrii:

- X prin intermediul căruia se specifică dacă s-a apelat data (0) sau a mai fost apelată (1) procedura LANS pentru un proces. Acest parametru a devenit necesar deoarece la prima apelare se introduce în stivele tuturor proceselor declarate adresa următoare apelului lui LANS, la care procesele vor trebui să revină după terminarea execuţiei. Memorarea adresei de revenire se va face în variabila AREV, prin intermediul unei secvenţe de cod.

- AREV reprezintă variabila locală care conţine adresa de revenire care se va introduce în baza stivei fiecăruia din procesele declarate.

Procedura implementată are următorul aspect:

```
PROCEDURE LANS (X: INTEGER);
VAR AREV: INTEGER;
    PR: INTEGER;
BEGIN
 INLINE ($E1/                          (* POP H     *)
        $22/AREV);                     (* SHLD AREV *)
 IF X=0 THEN
  FOR PR=1 TO PM DO
   BEGIN
    ADSP:= BCPS [PR ].ADRST;
    SALV;
    MEM:= ADSP;
    INLINE ($2A/AREV/                  (* LHLD AREV *)
           $E5);                       (* PUSH H    *)
    SALV;
    BCPS [PR ].ADRST:= ADSP;
   END;
  ADSP:= BCPS [RUN ].ADRST;
  SALV;
  MEM:= ADSP;
  AREV:= BCPS [RUN ].PC;
 INLINE ($2A/AREV/                     (* LHLD AREV *)
        $E5);                          (* PUSH H    *)
END;
```

Pimitive care realizează inițializarea nucleului

Procedura care asigură inițializarea sistemului de operare este amplasată fie în memoria locală a unui MP destinat special operației de inițializare a sistemului, fie în memoria comună. Procedura de inițializare impune utilizarea unui număr de două primitive:

- BEGINN care realizează inițializarea variabilelor de condiție și a semafoarelor. Se inițializează contorul corespunzător numărului de tampoane libere (în cazul nostru 2) și cel al tampoanelor ocupate (0). Primitiva este proprie fiecărui MP în parte avînd rolul de a inițializa structurile de date locale, una dintre primitive avînd rolul de a inițializa și structurile de date globale prezente în memoria comună. Primitiva realizată este de forma:

```
PROCEDURE BEGINN;
VAR NP,I,PRI: INTEGER;
    C: CHAR;
BEGIN
 FLAG:= FALSE;
 PM:= 0;
 INISEM (GATE);
 WWORK.COUNT:= 0;
 INISEM (WWORK.SEM);
 WWORK.SEM.COUNT:= 0;
 WRET.COUNT:= 0;
 INISEM (WRET.SEM);
 WRET.SEM.COUNT:= 0;
 WORK.COUNT:= 0;
 RETURN.COUNT:= 2;
 RETURN.QAREA [ 1 ] := 1;
 RETURN.QAREA [ 2 ] := 2;
 INIL (RTR);
 WRITELN ('DECLARATI PROCESELE:');
 REPEAT
  WRITELN ('DATI INDEXUL PROCESULUI');
  READLN (NP);
  WRITELN ('DATI PRIORITATEA PROCESULUI',NP);
  READLN(PRI);
  DECLP(NP,PRI);
  WRITELN('TASTATI D DACA PROCESUL E GATA DE EXECUTIE');
  READLN(C);
```

```
    IF (C='D') AND (PM<>0) THEN INSERT (RTR,PM);
    WRITELN ('CONTINUATI ?');
    WRITWLN ('DACA DA,TASTATI D');
    READLN(C);
  UNTIL C <> 'D';
END;
```

- DECLP primitiva are rolul de a iniţializa un proces; se stabileşte stiva,adresa de început cît şi prioritatea de execuţie. De asemenea se efectuează trecerea procesului în şirul GDE aferent fiecărui MP în care este prezent procesul.

Se face observaţia că în cazul prezentei implementări declararea proceselor se face prin dialog cu consola,soluţia impusă de caracterul experimental al implementării. Primitiva implementată este de forma:

```
PROCEDURE DECLP (I,PRI: INTEGER);
BEGIN
PM:= PM+1;
WITH BCPS [ PM ] DO
BEGIN
 Q:= 0
 PRIOR:= PRI;
   CASE I OF
    1: BEGIN
         ADRST:= ADDR(STIV1)+511;
         PC:= ADDR(PROC1);
         NUME:='PROCES1';
       END;
    2: BEGIN
         ADRST:= ADDR(STIV2)+511;
         PC:= ADDR(PROC2);
         NUME:='PROCES2';
       END;
    3: BEGIN
         ADRST:= ADDR(STIV3)+511;
         PC:= ADDR(PROC3);
         NUME:='PROCES3';
       END;
    4: BEGIN
         ADRST:= ADDR(STIV4)+511;
         PC:= ADDR(PROC4);
```

```
        NUME:='PROCES4';
      END
    ELSE
      WRITELN('PARAMETRII ERONATI')
    END;
  END;
END;
```

Proceduri destinate planificării proceselor

Sînt utilizate pentru modificarea poziţiei unui proces aflat într-un şir de aşteptare. Deoarece acestea apar şi sînt folosite de către primitivele definite anterior,fiind incluse indirect în structura nucleului. Cu toate că prezintă structura şi funcţiile unor primitive,ele vor apare mai corect ca proceduri ale nucleului.

Procedurile utilizate intern sînt:

- INIL(VAR L: LISTA) cu rol de iniţializare a unei liste vide;
- INSERT(VAR L: LISTA; X: INTEGER) cu rol de înserare a nodului X în lista L în ordinea descrescătoare a priorităţii;
- SUPRIM(VAR L: LISTA) are rolul de suprimare a primului nod al listei L;

7.5. Detalii de construire a proceselor.

In cadrul sistemului pot apare două categorii de procese:

- independente,care nu cooperează cu nici un alt proces;
- concurente,care interacţionează unul cu celălalt în cadrul unor relaţii de tip producător/consumator.

Ambele categorii de procese se descriu prin proceduri PASCAL cu o structură organizatorică impusă de categoria căreia îi aparţine procesul în cauză. Pentru prima categorie nu există nici o restriţie în ceea ce priveşte modul în care se scrie procedura care implementează procesul respectiv. In schimb,pentru procesele care aparţin celei de-a doua categorii apar cîteva reguli privind modul de secvenţiere şi utilizare a primitivelor nucleului care sînt utilizate în procesul de intercomunicare. Aceste reguli sînt observabile în structura unui proces consumator,producător,prezentate în continuare:

```
PROCEDURE PROC1;
(* PROCES PRODUCATOR *)
```

```
VAR N: INTEGER;
BEGIN
 N:= 0;
 REPEAT
  SEND
  N:= N+1;
 UNTIL N=4;
END;
```

PROCEDURE PROC2;

```
(* PROCES CONSUMATOR *)
VAR R: INTEGER;
BEGIN
 R:= 0;
 REPEAT
  RECEIVE
  R:= R+1;
 UNTIL R=4;
END;
```

Se observă că atît procesele consumatoare cît şi cele producătoare s-au implementat ca nişte cicluri,în cadrul fiecărui ciclu eliberîndu-se,respectiv completîndu-se,cîte o zonă tampon. Cînd un proces a terminat de efectuat ciclurile (în exemplul dat 4) se va proceda la reîntoarcerea la adresa instrucţiunii care urmează după apelul procedurii LANS,adresa care a fost încărcată în baza stivei fiecărui proces la prima execuţie a lui LANS.

Testarea sistemului de operare multimicroprocesor pe un sistem monoprocesor

Deoarece sistemul s-a creat iniţial şi s-a testat pe un sistem monoprocesor,a fost necesară definirea unei secvenţe de lansare în execuţie a acestuia,care este specifică modului de realizare şi testare. Această secvenţă de lansare este de forma:

```
BEGIN
  X:= 0; ZERO:= 0;
  BEGINN;
  RUN:= SUPRIM(RTR);
  WHILE RUN<>0 DO
   BEGIN
    LANS(X);
```

```
        ADSP:= MEM;
        SALV;
        RUN:= SUPRIM(RTR);
        X:= 1;
      END
    END.
```

Ea reprezintă de fapt programul principal care se execută în contextul prezenţei unui număr variabil de procese care interacţionează.

8. CONCLUZII.

Contribuţii originale

Lucrarea abordează într-o manieră originală,problematica legată de stabilirea structurii sistemului de operare,funcţie de modul în care este rezolvată problema gestiunii resurselor întregului sistem μP. Pornind de la scopul şi obiectivele acestei lucrări se pot scoate în evidenţă următoarele contribuţii originale:

- Precizarea conceptului general de resursă a unui sistem μP prin stabilirea atributelor caracteristice. S-a propus introducerea noţiunii de resursă locală şi globală a unei arhitecturi μP,precum şi a noţiunii de resursă a unui sistem de operare μP. S-au studiat deosebirile care apar în cadrul mecanismelor implicate în gestiunea locală şi globală a acestora.
- Precizarea caracteristicilor arhitecturale care trebuie să stea în atenţia proiectantului de sisteme μP.
- Reconsiderarea noţiunii de proces,prin prisma interacţiunii dintre componentele sistemului μP şi aplicaţia căreia îi este destinat. Se specifică modul în care procesele pot sau nu constitui resurse ale sistemului,şi în ce măsură acestea primesc statutul de resursă globală. S-a propus introducerea termenilor de coordonare şi cooperare pentru a desemna tipurile specifice de interacţiuni care apar între resursele locale şi globale.
- Precizarea primitivelor caracteristice unui nucleu de operare standard care trebuie să stea în atenţia proiectantului de sisteme de operare μP.

- Elaborarea unei clasificări în patru categorii a nucleelor de sisteme de operare μP. Pentru fiecare în parte,se stabilesc caracteristicile şi deosebirile faţă de nucleul standard propus,se prezintă variantele posibile,analizîndu-se pînă la nivelul de pseudocod primitivele noi care intervin.

- Se propune introducerea a două clase de sisteme de operare,în care se împart categoriile de nuclee de operare μP elaborate. Incadrarea se face funcţie de modul în care mecanismele de coordonare şi sincronizare prezintă sau nu restricţii în gestiunea resurselor.

- Elaborarea a două modele pentru studiul gestiunii resurselor în sistemele μP; primul destinat analizei sistemelor μP cu restricţii de sincronizare,cel de al doilea destinat sistemelor μP fără restricţii de sincronizare. Cele două tipuri de modele în care se utilizează tehnica lanţurilor Markov,dau posibilitatea ca pe baza datelor furnizate de arhitectura sistemului (impusă de aplicaţie) şi a tipului de nucleu de operare ales,(considerat de proiectant optim),să se obţină o serie de informaţii privind comportarea sistemului. Rezultatele obţinute servesc la stabilirea modului optim în care se pot amplasa procesele (în memoria locală a unui MP sau în memoria comună), la tipul de nucleu folosit,la optimizarea mecanismelor de coordonare şi sincronizare locală şi globală prin specificarea lungimii optime a mesajelor.

- Elaborarea unei metodologii de proiectare pentru fiecare din nivelurile unui sistem de operare μP. In acest context se precizează structurile de date specifice fiecărui nivel împreună cu primitivele corespunzatoare. Se defineşte de asemenea modul în care se face trecerea de la un nivel inferior la unul imediat superior al sistemului de operare.

- Studiul posibilităţilor de configurare a unei arhitecturi μP realizată cu module MULTIPROM. Emiterea de recomandări privind modificările ce se cer a fi efectuate în modulele standard MULTIPROM,urmată de implementarea unui sistem de operare μP,de tip nucleu monolitic distribuit. Studiul şi sistemul realizat au validat în întregime modul nou de abordare a metodologiilor de proiectare şi realizare a sistemelor de operare μP.

In concluzie,se abordează pentru prima dată în mod unitar comportarea de ansamblu a unui sistem μP,care este privit ca o colecţie de resurse şi mecanisme care aparţin atît arhitecturii sistemului cît şi sistemului de operare,ambele categorii cooperînd la atingerea scopului impus de aplicaţia căreia îi este destinat.

Valoarea aplicativă şi direcţiile de dezvoltare viitoare.

Aspectele prezentate în lucrare au fost valorificate în cadrul unui număr de 6 lucrări publicate şi a unui contract de cercetare ştiinţifică. Este de menţionat faptul că principiul care a stat la baza conceptelor utilizate în lucrare au avut ca punct de plecare [BS86,SH84] ,un articol publicat în USA în 1986 în J. Chem. Inf. Comput. Sci. vol.26 No.2. în colaborare şi un capitol din cartea Minimum Steric Difference publicată în colaborare în 1984 în editura John Wiley & Sons Inc.Anglia.

Pe viitor se conturează noi perspective,ca urmare a domeniului de aplicabilitate din ce în ce mai larg a sistemelor mμP. Acest aspect,impune cu necesitate standardizarea într-o măsură din ce în ce mai mare a componentelor unui sistem de operare, astfel încît implementarea acestuia să reprezinte o operaţie uşor reproductibilă de la o aplicaţie la altă aplicaţie. In acelaşi timp marea diversitate a aplicaţiilor care pot apare va impune utilizarea obligatorie a unor criterii de alegere a unui anumit tip de nucleu de sistem de operare şi de dimensionare a structurilor de date aferente astfel încît performanţele sistemului pentru aplicaţia căreia îi este destinat să fie maximă.

ooOoo

9. BIBLIOGRAFIE

1. Ad79 N.Adam: "Current Issues in computer simulation",Academic Press,1979.

2. AG82 M.M.Ajmone and M.Gerla: "Markov models for multiple bus multiprocessors systems",IEEE Trans.Comput.,Vol.C-31, pag.239-248,Mar.1982.

3. AJ75 G.A.Anderson,E.D.Jensen: "Computer Interconnection Structures: Taxonomy Characteristics and Exemples",ACM Computing Surveys,Vol.7,Nr.4,Dec.1975,pag197-214.

4. Al75 A.O.Allen: "Elements of queing theory for system desing", Queuing theory for system desing,Nr.2,1975,pag.161-187.

5. An78 S.Andler: "Syncronisation Primitives and Verification of Concurrent Programs",Proceedings of the Second International Symposium on Operating Systems Theory and Practice,Rocquencourt,France,Oct.1978,pag.67-85.

6. BB8o B.A.Bowen,R.J.A.Buhr: "The Logical Design of Multiple-Microprocessor Systems",Prentice Hall New Jersey 198o

7. BD81 F.A.Briggs,M.Dubois: "Performance of cache-based multiprocessors",Journal of ACM 1981,pag181-19o.

8. Be83 W.W.Benjamin: "A Comparativ study of distributed resource sharing on multiprocessors",Journal of ACM,Nr.3,Ian. 1983,pag.3o1-3o7.

9. Bu73 J.P.Buzen: "Computational algoritms for closed queueing networks with exponential servers",Comm.ACM,Nr.16, 1973,pag.527-531.

1o. Bu78 R.J.A.Buhr: "Concurent high level language models appled to multimicroprocessor system development",Proceedings of the Second International Symposium on Operating Systems Theory and Practice,Rocquencourt,France Oct. 1978,pag.45-53.

11. BT76 J.W.Bowra,N.C.Torng: "The modeling and Design of Multiple Function Unit Processors",IEEE Transaction on Computers,Apr.1976,pag.21o-221.

12. CC81 J.Catterton,G.Casilli: "Universal development system is

aim of master-slave processors",Applying microprocessors,pag.93-98,1981.

13. CG87 I.Chlamtac,O.Ganz: "Performance of Asynchronous Multitrunk HYPERchannel Networks",IEEE Trans.Comput.Vol. C-36,Nr.2,Feb.1987,pag.138-145.

14. CH75a K.M.Chandy,U.Herzog: "Parametric Analysis of Queueing Networks",IBM J.Res.Develop.,January 1975,pag.27-42.

15. CH75b K.M.Chandy,U.Herzog: "Aproximate Analysis of General Queueing Networks",IBM J.Res.Develop.January 1975,pag. 43-49.

16. CI81 C.Constantinescu,T.Ionescu: "Bus priority resolution tehniques used in multimaster system",4th International Conference on Control Systems and Computer science, June 1981 Bucharest,Vol.2,pag.27-42.

17. CM81 K.M.Chandy and J.Misra: "Asyncronous Distributed Simulation via a Sequance of Parallel Computation",Comm.ACM, Vol.24,Nr.11,April 1981,pag.198-2o6.

18. Co78 D.Del Corso: "An experimental multimicroprocessor system with improved internal communication facilities",Euromicro Journal Nr.4,1978,pag.326-332.

19. Da82 M.Dahmke: "Microcomputer Operating System",Mc.Graw Hill 1982.

2o. DD81 P.J.Denning,D.Denis,J.Brumfield: "Low Contention Semaphores and Ready Lists",Comm.ACM,24,Nr.1o,Oct.1981,pag. 687-698.

21. Di65 E.W.Dijkstra: "Solution of problem in concurent programming control",Comm.ACM,8,Nr.9,Sept.1965,pag569-574.

22. DO86 G.Dodescu,I.Odăgescu: "Simularea sistemelor",Ed.Militară 1986.

23. EF73 J.J.Egil,J.M.Frederic: "Multiple Microprocessors with Common Main and Control Memories",IEEE Transactions on Computers,Vol.C-22,Nr.11,November 1973,pag.999-1oo7.

24. En76 P.H.Enslow: "Multiprocessors and paralel procesing",Academic Press 1976.

25. FL66 M.J.Flynn: "Very High Speed Computing Systems",Proc.IEEE, Vol.54,Nr.12,Dec.1966.

26. FT79 Kwok-Tung Fung: "On the Analysis of Memory Conflicts and Bus Contentinions in a Multiple Microprocessor System",IEEE Transaction on Computers,Vol.C-27,Nr.1,January 1979,pag.28-37.

27. FV81 G.D.Forney,J.E.Vander May: "8-bit microprocessors can control data networks",Applying microprocessors,pag. 135-141.

28. Go78 G.Gordon: "System Simulation",Prentice-Hall,Englewood cliffs,New Jersey 1981.

29. Go81 F.Gordon: "Microprocessor bus standard could care designers woes",Microprocessors and Microcomputers,Mc. Graw Hill 1981,pag.31o-316.

3o. GR78 G.Adams,T.Rolander: "Design motivations for multiple microcomputers systems",Comput.Design.,Martie 1978,pag. 81-89.

31. Ha81 B.Hartman: "16-bit 68ooo microprocessor camps an 32-bit frontier",Microprocessors and Microcomputers,1981, pag.126-134.

32. HC75 H.Hellerman,T.F.Conroy: "Computer System Performance",Mc Graw-Hill Book Company,New York,1975.

33. HH81 L.Harold,D.Harvey: "Operating Systems",Addison-Wesley Publishing Company,1981.

34. Hi81 A.Hidar: "A model of interference in a single shared bus multi-microprocessor",4-th International Conference on Control Systems and Computer Science,Iunie 1981, pag.179-184.

35.Ho82a St.Holban: "Sisteme multimicroprocesoare",Referat doctorat 1982.

36. Ho81 St.Holban: "Programarea calculatoarelor",Editura Facla, 1981,Timişoara.

37. Ho37 St.Holban: "Modele ale gestiunii resurselor în sistemele multimicroprocesor",Simpozion Fac.Automatică şi Calculatoare,Bucureşti,Oct.1987.

38. Io77 M.Iosifescu: "Lanţuri Markov finite şi aplicaţii",Ed.Tehnică Bucureşti,1977.

39. JH8o I.Jurcă,St.Holban: "Programe de comandă pentru un testor de module de memorie echipat cu uP I8o8o",Buletinul Stiinţific IPT,Vol.25 din198o,pag.183-191.

4o. JL81 P.A.Jacobson,E.D.Lazowska: "The Method of Surrogate Delays: Simultaneous Resource Possession in Analitic Models of Computer Systems" Comm.ACM 1981,pag.165-174.

41. K175 L.Kleinrock: "Queueing Systems,Vol.1 Theory",John Wiley 1975.

42. KM8o N.V.Kopchenova,I.A.Maron: "Computational Mathematics",Mir Publishers,Moscow.

43. KM81 B.J.Katz,S.P.Morse: "8o86 microcomputer bridges the gap between 8 and 16-bit designs",Microprocessors and Microcomputers,1981,pag.112-118.

44. KN78 P.Kus,N.Natarjan,K.S.Mukul: "Physical and Logical Abstractions in a Kernel",Proceedings of the Second International Symposium on Operating Systems Theory and Practice,Rocquencourt,France,Oct.1978,pag.359-368.

45. Ko78 H.Kobabayashi: "Modeling and Analysis: An Introduction to System Perormance Evaluation Methodology",Addison-Wesley,Reading,Massachusetts,1978.

46. La83 S.S.Lavenberg: "Computer Performance Modeling Handbook", Academic Press.

47. LZ86 E.D.Lazowska:J.Zahorjan: "Computer System Performance Evaluation Using Queueing Network Models",Ann.Rev.Comput.Sci.1986,pag.1o7-137.

48. MB83 J.R.Menand,M.Becker: "Modeling a Multiprocessor Architecture",IEEE Soft.Engineering,Vol.9,Nr.2,Martie 1983, pag.2o1-21o.

49. MC73 Gh.Mihoc,G.Ciucu,A.Muja: "Modele matematice ale aşteptării: Editura Academiei,Buc.1973.

5o. MH86a T.N.Mudge,J.P.Hayes,C.D.Buzzard,D.C.Winsor: "Analysis of Multiple-Bus Interconnection Networks",Journal of Parallel and distributed computing,Nr.3,1986,pag328-343.

51.MH87b T.N.Mudge,J.P.Hayes,D.C.Winsor: "Multiple Bus Architectures",Computer June 1987,pag.42-48.

52. MP78 W.Mahr,R.Patzet: "Improvements of multiprocessing capabilities of microprocessor busses",Euromicro Journal, Nr.4,pag.207-219.

53. Mü78 K.Mühlemann: "Comunication between processors without shared memory",paper presented at the Joint/IEEE on Interconnected Small Processors in Bern,21 November 1978.

54. Ng86 K.W.Ng: "Mesage-passing primitives for multimicroprocessor systems",Microprocessors and microsystems,Vol.10 Nr.3,April 1986,pag.156-160.

55. Pa83 Y.Paker: "Multi-microprocessor Systems",Academic Press, Inc.1983.

56. PT84 A.Petrescu,T.Moisa,s.a.: "Microcalculatoarele Felix M18, M18B,M118",Editura Tehnică,Bucureşti 1984,Vol.1,şi 2.

57. Pt85 D.D.Patterson: "Reduced Instruction Set Computers",Comm. ACM,Vol.28,Nr.1,Jan.1985,pag.8-21.

58. Ra75 H.A.Raphael: "Join micros into intelligent networks",Electron.Design.,Mar.1975.

59. Răd81 E.Rădăceanu: "Limbaje de simulare",Editura Militară,Bucureşti,1981.

60. Re81 M.Reiser: "Mean-value analysis and convolution method for queue-dependent servers in closed queueing networks", Performance Eval.1981,pag.7-18.

61. RG81 R.F.Rashid,R.G.George: "Accent; A communication oriented network operating system kernel",Comm.ACM,Nr.1,December 1981,pag.64-75.

62. SC81 C.H.Sauer,K.M.Chandy: "Computer Systems Performance Modeling",Prentice-Hall,Englewood,Cliffs,New Jersey,1981.

63. Sc81 H.Schwetman: "Computer System Models; An Introduction", Comm.ACM,Vol.24,Nr.11,April 1981,pag.105-112.

64. Sc83 K.Scherer: "Changing to 16 bits?",Siemens Components XVIII, Nr.3,1983,pag.86-92.

65. Sh74 A.C.Shaw: "The logical design of operating systems",Prentice-Hall,1974.

66. SH84 C.Strugaru,S.Holban: "Theoretical survey on interconnecting computers for didactic purposes",Buletinul şti-

inţific IPT,Vol.29 din 1984,pag.85-89.

67. St76 W.J.Stewart: "MARCA.A Markov Chain Analysor",raport intern IRISA,1976.

68. Sw77 R.J.Swan: "Cm*-A Modular Multi-microprocessor",AFIPS Conference Proceedings,Vol.46,1977.

69. Th72 Kenneth Thurber: "A Systematic Approach to the Design of Digital Bussing Structures",Proc.IEEE Fall Joint Computer Conference,1972,pag.719-74o.

7o. Th79 K.Thurber: "Parallel procesor architectures" - Part 1: General purpose systems,Comput.Design,Jan.1979,pag.93-97; Part 2: Special purpose Systems,Comput.Design,Febr.1979,pag.1o3-113.

71. Th86 T.Toroczkay: "Situaţia pe plan mondial a sistemelor de operare în timp real",Referat de doctorat 1986.

72. To86 D.Towsley: "Approximative Models of multiple bus multiprocessor systems",IEEE Trans.Comput.,Vol.C-35,pag.22o-228,Mar.1986.

73. Va86 Van Qost: "Multi-processor system description and simulation using structured multi-programming Languages", Faulty of Applied Science Brusseles.

74. Vă77 I.Văduva: "Modele de simulare cu calculatorul",Editura Tehnică,Bucureşti 1977.

75. Vă82 I.Văduva: "Limbajul SIMMUB",Manual de referinţă,Univ.Bucureşti,1982.

76. We74 A.J.Weissberger: "Distibuted Function Microprocessor Architecture",Comput.Design.,Nov.1974.

77. We77 A.J.Weissberger: "Analysis of Multiple-Microprocessor System Architectures",Computer Design,June 1977,pag.151.

78. Wi82 C.William Mc Donald: "A Flexible Distributed Testbed for Real-Time Application",IEEE Transactions on Computers, October 1982,pag.25-38.

79. **73 *** "Bibliotheque mathematique",Manuel d'utilisation CII 1973.

8o. **75 *** "Intel8o8o Assomby Language Programming Manual",Intel Corp,Santa Clara,Calif.,1975.

81. **76a *** "INTEL Corp.,SBC 8o/2o Hardwarw Reference Manual,1976,

82. **77a *** "Intel MULTIBUS Interfacing",Intel Corp,Santa Clara, Calif.,1977.

83. **83a *** "VMS bus (VME serial bus),Specification Manual,November,1983.

84. **84a *** "INTEL Microcomputer Systems User's Manual",Sept. 1984.

85. **86a *** "Distributed Real-Time Multimicroprocessor System", Philips Industrial Data Proccesing,Findhoven,The Netherlands.

86. **86b *** "Multibus 11 Product Data Book",Intel Corporation, 1986.

87. **87a *** "IBM PS/2 Le banc d'essai complet du Systeme personnel/2,la nouvelle norme en micro-informatique",Science & Vie Micro,Nr.39,Mai 1987,pag.69-76.

88. **87b *** "Softwarw de intercomunicaţie pe magistrala MULTIPROM",Contract IPA,D.L.6226/1 1987.

89. BS86 A.T.Balaban,S.Holban: "FORTRAN 1V Program for Computing the Numbers of General Cubic Graphs on p Vertices",Journal of Chemical Information and Computer Science,1986,vol.26,no.2,pag.72-76.

9o. SH84 Z.Simon,A.Chiriac,S.Holban,D.Ciubotaru: "Minimum Steric Difference",John Wiley § Sons Inc.Anglia, 1984.

ooOoo