INSTITUTUL POLITEHNIC TIMISOARA
FACULTATEA DE ELECTROTEHNICA

AUREL PUSCA

T E Z A D E D O C T O R A T

STUDIUL ANOMALIEI DIMENSIUNII DE PAGINA IN CADRUL UNUI SISTEM IERARHIC DE MEMORII

CONDUCATOR STIINTIFIC

Prof. univ. dr.AL ROGOJAN

Timişoara

1978

C U P R I N S

./.

CAP.1. INTRODUCERE

Programul Partidului Comunist Român de făurire a societăţii socialiste multilateral dezvoltate şi înaintare a României spre comunism, adoptat de Congresul al XI-lea al P.C.R., prevede în "Obiectivele fundamentale ale etapei următoare a istoriei României socialiste" că în anii următori "electrotehnica şi îndeosebi electronica vor cunoaşte o puternică dezvoltare. Va trebui asigurată producerea în ţară a mijloacelor de calcul, de automatizare şi conducere necesare activităţii economice şi de producţie. "

"Orientările generale cu privire la întocmirea planului cincinal de dezvoltare economico-socială a României în perioada 1981-1985", adoptate de Conferinţa Naţională a PCR din 7-9 decembrie 1977, prevăd că "în industria electronică se va urmări asimilarea şi dezvoltarea producţiei de noi tipuri de calculatoare electronice, inclusiv echipamentele periferice aferente".

In acest cadru general se înscriu şi cercetările legate de proiectarea, realizarea şi utilizarea unor sisteme de memorii performante pentru calculatoarele electronice.

1. Obiectul lucrării

Scopul acestei lucrări este de a cerceta o problemă de interes apărută în gestiunea automată a ierarhiilor de memorii şi anume anomalia dimensiunii de pagină. Cercetarea, realizată cu ajutorul unui model original al comportării de acces a unui program, stabileşte cauzele apariţiei acestei anomalii şi, pe această bază, propune o procedură de evitare a apariţiei anomaliei.

Prin ierarhie de memorii înţelegem o organizare de diferite tipuri de dispozitive de memorii cu scopul obţinerii unui sistem de memorizare cu performanţe înalte, dar cu preţ scăzut.

Un interes deosebit, teoretic şi practic, prezintă ierarhiile lineare verticale de memorii (ILVM). In cadrul unei asemenea ierarhii ILVM dispozitivele de memorii sînt dispuse în

ordine crescătoare a timpurilor de acces şi a capacităţilor, pot comunica unele cu altele dacă sînt situate pe nivele vecine şi numai primul nivel al ierarhiei poate comunica cu unitatea centrală de prelucrare (UCP).

Cercetarea problemelor legate de proiectarea şi gestiunea automată a acestor ierarhii de memorii a devenit în ultimii ani o direcţie de interes major în preocupările legate de elaborarea unor sisteme de memorii mai rapide, cu capacităţi mai mari şi mai economice. Importanţa cercetării în domeniul sistemelor ierarhice de memorii este menţionată şi subliniată de numeroşi cercetători. Astfel, a devenit un fapt acceptat că folosirea algoritmilor de gestiune automată a sistemelor de memorii, construite din cîteva nivele cu diferite timpuri de acces, a condus la o simplificare semnificativă a efortului de programare. Deşi există pînă în prezent mai multe implementări a unor sisteme de memorii cu mai multe nivele, gestiunea automată a unor asemenea sisteme este un domeniu insuficient cercetat. De exemplu, în elaborarea unor algoritmi eficienţi de gestiune a unei ierarhii de memorii, precum şi în previziunea performanţei unei asemenea ierarhii au rămas unele probleme încă neclarificate.

Una din problemele apărute în domeniul gestiunii unei ierarhii de memorii şi care pînă în prezent nu a făcut obiectul unei cercetări analitice este anomalia dimensiunii de pagină. Cercetarea acestei probleme, stabilirea cauzei apariţiei anomaliei şi elaborarea unei proceduri de evitare a apariţiei acesteia sînt realizate în lucrarea prezentă.

Din punct de vedere al organizării lucrarea este împărţită în 6 capitole. Structura ei poate fi expusă prezentînd pe scurt, în cele ce urmează, conţinutul acestor capitole.

2. Prezentarea generală a lucrării

2.1. Capitolul 1, de introducere, precizează obiectul lucrării. In continuare se face o prezentare generală a conţinutului lucrării şi a contribuţiilor originale aduse în ea de autor.

./.

2.2. Capitolul 2 este o minimonografie dedicată problemelor majore ale studiului legat de sistemele ierarhice de memorii. Scopul lui este de a introduce noţiunile de bază în analiza ierarhiilor de memorii şi de a prezenta situaţia la zi în problemele de analiză, proiectare şi realizare a unor asemenea sisteme.

Astfel, se arată obiectivele realizării unui sistem ierarhic de memorii: obţinerea unor indicatori cost/performanţă avantajoşi; simplificarea şi automatizarea programării (un sistem de gestiune automată a unei ierarhii de memorii preia multe sarcini ce reveneau programatorilor, mărind astfel productivitatea muncii acestora); crearea posibilităţilor de utilizare a noilor tipuri de dispozitive de memorii şi extinderea domeniului aplicaţiilor; studiul comportării de acces a programelor şi al comportării sistemelor de calcul.

Se cercetează apoi parametrii principali ai unui sistem ierarhic de memorii şi anume: tehnologia de bază, configuraţia, comportarea de acces a programului şi algoritmii utilizaţi în transferul de informaţii între nivelele ierarhiei.

UCP, sub controlul programului, produce o serie secvenţială de referiri la sistemul de memorii. Aceste referiri ale procesorului se fac la adrese logice (nume ce servesc pentru a identifica în mod unic fiecare unitate individuală de informaţii memorate, indiferent de localizarea ei în cadrul ILVL). Secvenţa în timp a referirilor la adrese logice este denumită şirul de referiri.

Unitatea standard de memorare a informaţiilor şi de transfer între nivelele vecine ale ierarhiei de memorii este pagina. Secvenţa în timp a referirilor procesorului la pagini logice este denumită şirul de referiri de pagină.

Programul unui utilizator împreună cu datele aferente sînt încărcate în nivelul inferior al unei ILVL; pe măsura executării programului pagini individuale de instrucţii şi/sau de date sînt transferate spre partea superioară a ierarhiei, ajungînd în primul nivel, singurul care comunică cu unitatea centrală de prelucrare. Secvenţa combinată de adrese de instrucţii şi de adrese de date, secvenţă generată de către program, defineşte comportarea de acces la memorii a programului.

./.

Gestiunea unei ierarhii de memorii are rolul de a decide cum trebuie organizată informaţia în cadrul ierarhiei între nivelele acesteia şi în cadrul fiecărui nivel şi a decide când să se transfere informaţia între nivelele ierarhiei. Algoritmii utilizaţi în transferul de pagini între nivelele ierarhiei operează în sfera de aducere-plasare a paginilor (localizează în cadrul ierarhiei o pagină adresată de UCP şi a aduce în primul nivel) şi în sfera de înlocuire-replasare a paginilor (înlocuieşte o pagină din primul nivel al ierarhiei). Problema majoră nu este de a decide ce pagină trebuie să fie adusă, ci de a decide ce pagină trebuie să fie înlocuită. Regula utilizată pentru a decide ce pagină trebuie înlocuită este numită algoritm de înlocuire. Situaţia în care într-un nivel oarecare al ierarhiei este adresată o pagină ce nu se găseşte în acel nivel este denumită defect de pagină şi are ca consecinţă iniţierea de către sistemul de gestiune automată a ierarhiei a acţiunii de aducere a paginii cerute din nivelul inferior al MIM în care aceasta este dispusă.

În continuare se introduc indicatorii de performanţă ai unui sistem ierarhic de memorii: proporţia succeselor (raportul dintre numărul de referiri de pagină satisfăcute şi numărul total al referirilor de pagină în cadrul unui anumit nivel), proporţia eşecurilor (raportul dintre numărul de defecte de pagină şi numărul total al referirilor de pagină în cadrul unui anumit nivel), timpul de acces efectiv şi costul efectiv.

Evaluarea şi optimizarea unor sisteme ierarhice de memorii se pot realiza cu tehnici diferite de simulare şi modelare matematică. Simularea este utilizată în situaţiile în care nu sînt disponibile tehnici analitice; ea asigură rezultate a căror exactitate depinde de gradul de detaliere cu care este realizată; de obicei este costisitoare. Modelarea matematică în studiul ierarhiilor de memorii se utilizează în situaţiile în care se urmăreşte în special o înţelegere calitativă a modului în care sistemul este afectat de variaţiile diferiţilor parametri; ea conduce la rezultate de

./.

obicei mai puţin precise ca în cazul simulării.

Dintre tehnicile de simulare sînt discutate cele de eşantionare şi de tratare pe principiul stivei, iar tehnicile de modelare matematică sînt exemplificate prin utilizarea programării neliniare şi a modelelor cu şiruri de aşteptare.

In continuare sînt comentate unele realizări şi cercetări în domeniul ierarhiilor de memorii.

Astfel sînt analizate unele sisteme cu memorie virtuală paginată şi segmentată. In cadrul sistemelor cu memorie virtuală paginată se cercetează factorii ce afectează performanţa unui asemenea sistem: viteza şi numărul dispozitivelor de paginare, viteza unităţii centrale de prelucrare, capacitatea memoriei reale, capacitatea memoriei virtuale. In continuare sînt arătate căile de îmbunătăţire a acestei performanţe, care sînt: selectarea judicioasă a proporţiei între capacitatea memoriei reale şi capacitatea memoriei virtuale, ţinînd seama de viteza de calcul a UCP; selectarea unei dimensiuni de pagină corespunzătoare capacităţii sistemului de operare; utilizarea unui număr sporit şi mai rapid de dispozitive de paginare; utilizarea unui cod ce nu se modifică pe el însuşi; utilizarea unor memorii tampon de I/E mai mari; structurarea programelor de aplicaţii; alocarea unei anumite capacităţi de memorie reală în cadrul aplicaţiilor în prelucrarea pe loturi; utilizarea unui algoritm de planificare a priorităţilor; utilizarea unor proceduri de paginare fracţionată; utilizarea în cadrul sistemului a mai multor dimensiuni distincte de pagină. In încheierea capitolului sînt cercetate sisteme realizate pe baza unor ILVM cu două şi trei nivele.

2.3. In capitolul 3 este prezentată anomalia dimensiunii de pagină.

Dimensiunea paginii este un parametru important al unei ierarhii de memorii, afectînd în mod direct performanţa acesteia. Din acest motiv s-a acordat o atenţie deosebită determinării dimensiunii optime de pagină. S-a constatat că factorii principali ce influenţează alegerea dimensiunii de pagină sînt fragmentarea memoriei şi eficienţa operaţiei de transport a paginii. Prin fragmentarea memoriei se înţelege împărţirea memoriei disponibile într-un număr mare de domenii libere, separate prin do-

./.

menii ocupate. Acest fenomen este de natură statistică şi apare în situaţia în care domenii de memorie de dimensiuni variabile sînt ocupate şi apoi eliberate de diverse rutine, după intervale variabile de timp. Fragmentarea memoriei conduce la scăderi în utilizarea acesteia, căci apar situaţii în care fiecare din domeniile libere disponibile este prea mic pentru a satisface o cerere de spaţiu de memorie, deşi suma capacităţilor domeniilor libere este suficientă. Utilizarea memoriei este măsurată prin procentul din capacitatea totală a memoriei a capacităţii memoriei ocupate de către cererile satisfăcute de spaţiu de memorie. O problemă de interes deosebit cercetată a fost efectul asupra utilizării memoriei a rotunjirii cererilor de memorie la un multiplu întreg al unităţii de alocare (pagini). S-a constatat astfel că prin mărirea paginii utilizarea memoriei scade. Acest efect se datorează faptului că pe măsura creşterii dimensiunii paginii şi datorită rotunjirii cererii de memorie se măreşte de asemenea partea din spaţiul de memorie alocat şi nefolosit (deci irosit).

Timpul necesar pentru a realiza transportul unei pagini între două nivele vecine ale unei ILVM este format din timpul mediu de acces şi din timpul de transfer.

Timpul mediu de acces al unui dispozitiv de memorii este valoarea medie a timpului consumat de acest dispozitiv de memorii între momentul primirii unei adrese fizice de înregistrare a unei unităţi informaţionale şi momentul în care această unitate informaţională devine disponibilă. Timpul de transfer al unei pagini între două nivele vecine ale unei ILVM este determinat de dimensiunea paginii şi de rata de transfer cu valoare mai mică a dispozitivului de memorii situat pe nivelul mai inferior în cadrul ierarhiei.

La dispozitivele de memorii ce intră în componenţa unei ILVM timpul de acces variază considerabil mai mult decît variază rata de transfer. Această situaţie pune problema avantajelor posibile de obţinut (respectiv un timp mediu mai mic de transport al paginilor între nivelele ierarhiei) prin transferul în cadrul unui singur acces, a unui volum sporit de informaţii sau, cu alte cuvinte, în cadrul utilizării unei

./.

dimensiuni mărite de pagină. S-a cercetat creşterea timpului de transport în cazul transferului unei pagini de dimensiune dublă 2p faţă de timpul de transport al unei pagini de dimensiune p constatîndu-se că această creştere este substanţială în cadrul transferurilor între nivelele superioare ale ierarhiei, caracterizate prin timpi rapizi de acces şi este mică pentru nivelele inferioare ale ierarhiei, caracterizate prin timpi lenţi de acces. Astfel, datorită creşterii mari a timpului de transport, pentru dispozitivele de memorii cu timpi rapizi de acces utilizarea unei pagini de dimensiune mărită nu creează avantaje considerabile; în cazul însă al dispozitivelor de memorii cu timpi lenţi de acces, datorită creşterii mici a timpului de transport, utilizarea unei pagini de dimensiune mărită este avantajoasă. Introducerea recentă în cadrul ierarhiilor de memorii a unor noi tehnologii de dispozitive de memorii (memorii CCD - bazate pe dispozitive cu transfer de sarcină, memorii bubble - cu bule magnetice şi altele) cu timpi rapizi de acces reduce drastic avantajele posibile ale utilizării unor dimensiuni mari de pagină.

Pe baza analizei factorilor implicaţi în alegerea dimensiunii de pagină se trage concluzia că utilizarea unor pagini de dimensiuni mici în cadrul sistemelor de calcul moderne bazate pe ILVM conduce la îmbunătăţiri în gestiunea sistemului de memorii. Aceste îmbunătăţiri se referă la utilizarea mai eficientă a spaţiului de memorie şi la valoarea mică a consumului neproductiv de timp utilizat pentru accesul hardware şi gestiunea software a paginilor.

In continuare este definită situaţia de anomalie a dimensiunii de pagină. In acest scop vom considera execuţia unui program în cadrul unei ILVM în două situaţii distincte, caracterizate prin utilizarea unor dimensiuni diferite de pagină p1 şi p2 astfel ca

$$p_2 = \frac{p_1}{k}$$

unde, de obicei, $k = 2^i$, $i = 1,2,...$

Situaţia de anomalie a dimensiunii de pagină apare în cazul în care numărul defectelor de pagină generate la execuţia programului şi utilizarea unei pagini de dimensiune mai mică p_2

./.

va fi de peste k ori mai mare decît numărul defectelor de pagină generate la execuţia aceluiaşi program şi utilizarea unei pagini de dimensiune p_1. Numărul mare de defecte de pagină, sau, cu alte cuvinte, activitatea exagerată de paginare observată în cazul utilizării unei pagini de dimensiuni reduse p_2, are ca efect creşterea consumului neproductiv de timp utilizat pentru accesul hardware şi gestiunea software a paginilor. Deci, deşi în general utilizarea unor dimensiuni mici de pagină aduce îmbunătăţiri în utilizarea sistemului de memorii, în cazul apariţiei situaţiei de anomalie a dimensiunii de pagină se constată o scădere a performanţei (consum mare de timp neproductiv). Anomalia dimensiunii de pagină a fost constatată şi descrisă, dar nici o lucrare nu a fost dedicată studiului acestei probleme şi clarificării factorilor implicaţi în ea. In consecinţă lipseşte o înţelegere precisă a mecanismului de apariţie a acestei anomalii. In aceste condiţii rezultă necesitatea unor studii suplimentare în această direcţie. Importanţa unor asemenea studii rezultă din faptul că pe baza cunoaşterii mecanismului de apariţie a acestei anomalii se vor putea elabora metode sau proceduri de prevenire a anomaliei. Aceste metode şi proceduri vor avea un interes practic imediat. Ele vor permite obţinerea unor îmbunătăţiri suplimentare în utilizarea unui sistem ierarhic de memorii prin folosirea unor dimensiuni reduse de pagină, fără riscul apariţiei unei activităţi exagerate de paginare.

2.4. In capitolul 4, cu scopul de a crea un instrument care să poată fi utilizat în cercetarea anomaliei dimensiunii de pagină, este elaborat un model analitic al comportării de acces a unui program într-un mediu cu memorie virtuală. Elaborarea acestui model a fost necesară deoarece nici unul din modelele analitice ale comportării de acces a unui program descrise în literatură nu poate fi folosit în scopul urmărit de noi, întrucît aceste modele nu iau în consideraţie variaţia dimensiunii de pagină.

Capitolul debutează prin analiza critică a modelelor analitice de bază ale comportării de acces a unui program studiate pînă în prezent în literatură.

./.

Astfel în cazul funcţiei de timp de existenţă, care stabileşte o relaţie între lungimea medie a intervalelor de execuţie (între defecte succesive de pagină) şi dimensiunea memoriei alocate programului pentru desfăşurare, se constată că modelul este valid pentru programe reale numai pe o porţiune a domeniului de variaţie a alocării memoriei şi că parametrii acestui model nu sînt în mod clar definiţi pe baza proprietăţilor programului care este modelat.

In privinţa modelului de grup de lucru, care este bazat pe recunoaşterea comportării localizate a programelor (prin comportare localizată înţelegînd faptul că în decursul oricărui interval de execuţie programul favorizează, în mostra lui de referiri la memorie, unele din paginile lui într-o măsură mai mare decît pe altele) se constată dificultatea de a măsura dinamic parametrul de grup de lucru, cu scopul de a estima dimensiunea grupului de lucru; de asemenea modul cum este definită dimensiunea grupului de lucru nu permite includerea paginilor care sînt adresate o singură dată. In continuare sînt analizate critic cîteva modele stochastice, ce se referă la reprezentarea analitică a şirului de referiri de pagină a unui program: modelul cu probabilităţi constante şi modelul markovian staţionar. Astfel se constată că modelul ce presupune probabilităţi constante de referire a paginilor sau probabilităţi constante de tranziţie între pagini impune un grad de simplificare prea mare al comportării de acces a unui program, iar modelul ce utilizează proprietăţile unui proces semi-Markov conduce la necesitatea estimării unui număr mare de parametri, care nu pot fi puşi în legătură directă cu proprietăţile programului modelat. In această situaţie pentru analiza anomaliei dimensiunii de pagină apare necesitatea elaborării unui instrument de cercetare corespunzător, în cazul de faţă un model al comportării de acces a unui program, care să ia în consideraţie şi efectul variaţiei dimensiunii de pagină, iar parametrii lui să fie în mod clar definiţi pe baza proprietăţilor programului care este modelat.

In cele ce urmează, ţinînd seama de cele de mai sus, este elaborat un model analitic al comportării de acces a unui program într-un mediu cu memorie virtuală, model ce presupune următoarea comportare de acces a programului:

./.

(1) selectarea aleatoare a unei locaţii "a" în spaţiul de adrese virtuale al programului;

(2) selectarea aleatoare a unui ciclu cu lungimea α;

(3) accesul la α locaţii în ciclul (2) de mai sus secvenţial de δ ori;

(4) repetarea paşilor de mai sus.

Deci, în conformitate cu cele de mai sus, comportarea de acces a programului constă în apelarea unei serii de cicluri, poziţia şi dimensiunea cărora se schimbă în mod aleator.

Se urmăreşte apoi a se determina numărul defectelor de pagină generate de un program cu o asemenea comportare de acces. În acest scop se cercetează efectul asupra numărului defectelor de pagină a următorilor factori:

- lungimea ciclului de program;
- pregătirea anterioară apelării ciclului curent;
- repartiţia lungimii ciclurilor; pentru aproximarea repartiţiei lungimii ciclurilor este utilizată repartiţia de tipul Γ care permite o deosebită flexibilitate în controlul asupra formei curbei ce reprezintă funcţia de densitate de repartiţie a lungimii ciclurilor.

În acest mod în lucrare se obţine o expresie analitică pentru funcţia ce determină valoarea medie a numărului de defecte de pagină, care apar în decursul execuţiei unui program într-un mediu cu memorie virtuală.

Modelul elaborat este verificat prin faptul că, pe baza lui, se regăsesc pe cale analitică:

(1) caracteristicile calitative ale funcţiei ce determină numărul defectelor de pagină generate la execuţia unui program, caracteristici observate în cadrul unor studii publicate de simulare;

(2) rezultatele numerice experimentale.

Studii publicate de simulare au stabilit, pe baza analizei unor şiruri de referiri de pagină generate în cadrul execuţiei unor programe reale, următoarele 3 caracteristici ale funcţiei ce determină numărul defectelor de pagină gene-

./.

rate la execuţia unui program:

(a) această funcţie este o funcţie concavă în dependenţa de numărul de instrucţii executate;

(b) această funcţie are o asimptotă pentru valori mari ale numărului de instrucţii executate;

(c) numărul de instrucţii executate de program între defecte succesive de pagină creşte cu numărul de pagini acumulate de program.

In lucrare aceste 3 caracteristici sînt stabilite analitic, pe baza expresiei obţinute pentru funcţia ce determină valoarea medie a numărului de defecte de pagină ce apar la execuţia unui program.

In privinţa verificării rezultatelor numerice obţinute cu ajutorul modelului s-au folosit unele rezultate experimentale publicate (există raportate în literatură numeroase rezultate experimentale privind comportarea de acces a unor programe într-un mediu cu memorie virtuală; rezultatele folosite de noi sînt acceptate unanim ca semnificative). Pentru programele pentru care avem deja la dispoziţie rezultate numerice experimentale s-au calculat cu ajutorul modelului, folosind expresia analitică ce determină valoarea medie a numărului defectelor de pagină şi utilizînd un program pe care l-am denumit GAMINC, valorile pentru numărul defectelor de pagină generate în cadrul execuţiei acestor programe. Pe baza comparării rezultatelor numerice experimentale publicate în mai multe lucrări cu cele determinate analitic cu ajutorul modelului propus se constată o bună concordanţă. In acest mod se dovedeşte că modelul propus dă rezultate corecte.

2.5. In capitolul 5 se cercetează anomalia dimensiunii de pagină cu ajutorul modelului elaborat în capitolul precedent. Anume, pe baza expresiei analitice obţinute se calculează valoarea medie DPA a numărului de defecte de pagină care apar în decursul execuţiei unui program şi utilizării succesive a două dimensiuni de pagină p şi, respectiv, p/k.

Fie aceste valori DPA (p) şi DPA (p/k).

Situaţia de anomalie a dimensiunii de pagină apare în cazul în care numărul defectelor de pagină generate la execuţia programului şi utilizarea unei pagini de dimensiune p/k va fi

./.

de peste k ori mai mare decît numărul defectelor de pagină generate la execuţia aceluiaşi program şi utilizarea unei pagini de dimensiune p, adică

$$DIF = DPA\left(\frac{p}{k}\right) - k \cdot DPA(p) > 0$$

Pornind de la expresia pentru DPA(p) obţinută în capitolul 4, se obţine pentru DIF o expresie analitică relativ simplă, ce permite identificarea posibilităţilor de apariţie a situaţiilor de anomalie a dimensiunii de pagină.

Pentru a interpreta sensul rezultatului analitic obţinut se defineşte comportarea localizată a unui program într-un mediu cu memorie virtuală. Prin această comportare localizată se înţelege faptul că programul în decursul unui anumit interval de execuţie, favorizează (în cadrul şirului lui de referiri de pagină), unele din paginile lui într-o măsură mai mare decît pe altele. Factorii care conduc la această comportare sînt: fluxuri de instrucţii secvenţiale, modularitatea funcţională, organizarea datelor înrudite după conţinut, ciclurile, modul de programare. Se cercetează apoi tipurile de comportare localizată a unui program şi anume localizarea temporală şi cea spaţială. Pe baza acestor tipuri de comportare localizată a unui program se introduce conceptul de localizare, cu caracter general, spaţio-temporală. Se cercetează apoi modul cum diferiţi algoritmi de înlocuire iau în consideraţie localizarea temporală şi cea spaţială. Se constată că algoritmii uzuali de înlocuire se preocupă mult de aspectele temporale ale mostrei de referiri a programului la spaţiul adreselor. Aspectele spaţiale sînt tratate ca un produs secundar, prin faptul că algoritmii de înlocuire aduc în memoria principală, la un moment dat, o pagină întreagă, adică o regiune spaţială.

Ţinînd seama de acest mod în care sînt luate în consideraţie localizarea temporală şi spaţială a programelor de către algoritmii de înlocuire uzuali, se constată că micşorarea dimensiunii de pagină conduce la creşterea performanţei (mărirea valorii proporţiei succeselor) algoritmilor de înlo-

./.

cuire în cazul unui program caracterizat printr-o comportare localizată pronunţat temporală şi la scăderea performanţei (micşorarea valorii proporţiei succeselor) în cazul unui program caracterizat printr-o comportare localizată pronunţat spaţială. Acest lucru este o urmare a faptului că, prin micşorarea dimensiunii de pagină (în cadrul unei alocări date în M_1), în nivelul M_1 al ILVM se va realiza o colecţie mai variată de porţiuni de program, deşi fiecare din ele mai mică ca dimensiune. Acest tip de colecţie poate favoriza un program cu referiri dispersate şi grupate la spaţiul adreselor, ca urmare a unui caracter temporal al comportării sale de acces. Pe de altă parte însă, micşorarea dimensiunii de pagină defavorizează un program ce are o comportare de localizare spaţială căci,prin micşorarea dimensiunii paginii,se micşorează şi numărul de adrese ce pot fi apelate succesiv în cadrul aceleiaşi pagini. Se subliniază faptul că algoritmii de înlocuire trebuie să ia în consideraţie ambele tipuri de comportare localizată a programelor: atît temporală, cît şi spaţială. In final, pe baza conceptului de localizare spaţio-temporală,se dă o interpretare cu caracter general rezultatului analitic obţinut în cercetarea anomaliei dimensiunii de pagină şi anume că situaţia de anomalie poate apare în cazul acelor programe sau secţiuni de program caracterizate printr-o slabă localizare temporală şi o pronunţată localizare spaţială.

2.6. Capitolul 6 are ca obiect studiul prevenirii apariţiei anomaliei dimensiunii de pagină. Elaborarea unor măsuri care să prevină apariţia acestei anomalii este realizată prin cercetarea mai în detaliu a mecanismului prin care comportarea de acces a programului conduce la apariţia anomaliei, respectiv prin cercetarea şirului de referiri de pagină. Această cercetare ne-a condus la identificarea situaţiilor de apariţie a anomaliei dimensiunii de pagină. Pe baza identificării acestor situaţii, în lucrare se propune o procedură de modificare a algoritmilor de înlocuire, care realizează prevenirea apariţiei anomaliei.

Această procedură constă în următoarele:

(1) paginile programului în cauză se organizează în grupe, mărimea cărora corespunde unei dimensiuni mărite de pagină; o asemenea grupă este numită pagină-martor;

(2) algoritmii de înlocuire uzuali ce aplică la ansamblul paginilor unei grupe.

./.

Se consideră situaţiile în care algoritmul de înlocuire operează (a) asupra mulţimii paginilor programului şi-în paralel- (b) asupra mulţimii paginilor-martor. Procedura propusă evită faptul ca, la un moment dat în desfăşurarea unui program, o pagină oarecare A a acestuia să fie înlocuită dintr-un nivel al ierarhiei (în cadrul situaţiei a de mai sus), dacă, în acelaşi moment al desfăşurării programului, pagina-martor din care pagina A face parte nu a fost şi ea înlocuită din acelaşi nivel (în cadrul situaţiei b). Condiţia ca o pagină a unei grupe să nu poată fi înlocuită dintr-un nivel al ierarhiei, dacă pagina-martor corespunzătoare nu a fost şi ea înlocuită din acelaşi nivel, poate fi realizată în diferite moduri, care sînt discutate în lucrare.

Modul de aplicare al acestei proceduri este exemplificat în cazul unor algoritmi de înlocuire cunoscuţi, dar modificaţi conform procedurii descrise: anume, în lucrare se dau exemple de aplicare a algoritmilor LRU şi FIFO modificaţi. Se constată, în cazul şirurilor de referiri de pagină considerate:

(1) aplicarea algoritmilor LRU şi FIFO nemodificaţi conduce la apariţia anomaliei dimensiunii de pagină;

(2) aplicarea algoritmilor LRU şi FIFO modificaţi previne apariţia anomaliei.

In acest context se discută unele aspecte legate de aplicarea algoritmilor modificaţi asupra ordonării paginilor.

In continuare se analizează eficienţa aplicării algoritmilor modificaţi. Se stabileşte că procedura de modificare a algoritmilor de înlocuire fixează o limită superioară pentru numărul defectelor de pagină în regiunea de referire densă a programului la memorie, fără a micşora valoarea proporţiei succeselor în regiunile cu referiri rară şi moderată. Astfel, utilizarea acestei proceduri creează posibilitatea ca să se obţină îmbunătăţiri suplimentare în gestiunea unui sistem ierarhic de memorii prin utilizarea unor dimensiuni reduse de pagină, fără riscul unui consum mare de timp neproductiv datorat activităţii exagerate de paginare, procedura propusă evitînd apariţia anomaliei dimensiunii de pagină.

./.

In încheierea capitolului sînt expuse unele probleme, legate de cele tratate în lucrare, cercetarea cărora ar prezenta interes în perspectivă:

- studiul şi elaborarea unor algoritmi de înlocuire, care să ia în consideraţie comportarea de localizare spaţială a programelor;
- studiul relaţiei algoritmilor modificaţi prin procedura propusă cu alţi algoritmi de înlocuire;
- cercetarea eficienţei algoritmilor modificaţi prin studii suplimentare (prin simulare şi măsurători directe efectuate în cadrul unei ierarhii de memorii).

2.7. In capitolul 7, de concluzii, după sublinierea interesului teoretic şi practic al rezolvării problemei cercetate în lucrare, a anomaliei dimensiunii de pagină, în cadrul general al problemelor legate de gestiunea automată a ierarhiilor de memorii, se prezintă contribuţiile originale aduse în cadrul acestei lucrări de autor. Astfel, contribuţiile originale mai importante se referă la:

- elaborarea unui model analitic al comportării unui program operînd într-un mediu cu memorie virtuală, prin obţinerea unei expresii analitice pentru funcţia ce determină valoarea medie a numărului de defecte de pagină, ce apar în decursul execuţiei unui program într-un mediu cu memorie virtuală;
- stabilirea proprietăţilor funcţiei ce determină valoarea medie a numărului defectelor de pagină, ce apar în cadrul execuţiei unui program într-un mediu cu memorie virtuală;
- studiul analitic, cu ajutorul modelului elaborat al comportării de acces a programului, a situaţiilor de apariţie a anomaliei dimensiunii de pagină;
- introducerea conceptului de comportare localizată spaţio-temporală şi interpretarea cu caracter general, pe baza acestui concept, a apariţiei anomaliei dimensiunii de pagină;
- identificarea mecanismului de apariţie a anomaliei dimensiunii de pagină pe baza cercetării şirului de referiri de pagină a programului;
- elaborarea unei proceduri de prevenire a apariţiei anomaliei dimensiunii de pagină şi descrierea modului ei de aplicare - modificarea corespunzătoare a algoritmilor de înlocuire uzuali şi modul lor de operare;

./.

- studiul eficienţei algoritmilor modificaţi prin procedura propusă şi precizarea unor perspective de interes în domeniu.

*

* *

Realizarea prezentei lucrări a fost posibilă sub îndrumarea permanentă a prof.dr.ing. Al.Rogojan, conducătorul ştiinţific al lucrării, căruia autorul îi exprimă pe această cale profunde mulţumiri pentru întreaga activitate de îndrumare ştiinţifică şi metodologică oferită.

Autorul exprimă de asemenea mulţumirile sale prof. D. Ferrari, dr.M. Kobayashi şi dr.E. Lau (EECS - Universitatea Berkeley - SUA) şi ing.C. Maşek (ITC - Bucureşti) pentru discuţiile foarte utile avute asupra unor probleme din lucrare.

CAP.2. SISTEME IERARHICE DE MEMORII

1. Generalităţi.

Evoluţia sistemelor de calcul a fost marcată de o cerere în continuă creştere pentru dispozitive de memorii mai rapide, cu capacităţi mai mari şi mai economice. Accentul principal în această evoluţie a fost pus pînă în prezent pe tehnologia hardware, însă, pe măsură ce sistemele de calcul au avansat, s-a manifestat un interes în creştere în problema de proiectare a sistemelor, ce realizează utilizarea cea mai performantă şi cea mai eficientă a unor tehnologii hardware date.

O metodă importantă de obţinere a unei utilizări eficiente a unor tehnologii hardware date este utilizarea ierarhiilor de memorii. Astăzi este un lucru recunoscut faptul că cerinţele în conflict a unor memorii cu performanţe înalte, dar de cost scăzut, pot fi satisfăcute printr-o mixtură de tehnologii, combinînd dispozitive de memorii - DM costisitoare cu performanţe ridicate cu DM ieftine, cu performanţe scăzute. Acestei strategii i s-au atribuit diferite nume ca "sistem ierarhic de DM", "gestiunea automată a DM cu mai multe nivele", "memorie virtuală" şi "sistem de memorie virtuală pentru gestiunea automată a unor ierarhii de DM cu mai multe nivele". O ierarhie de memorii a fost pentru prima oară implementată în sistemul de calcul elaborat la Universitatea din Manchester în Anglia în 1949.

Cercetările în domeniul tehnicilor automate pentru ierarhii de memorii pot fi datate în urmă cu cca. 15 ani, însă şi în prezent sînt rămase nerezolvate sau insuficient înţelese multe probleme. Această situaţie poate fi parţial explicată prin faptul că aceste sisteme de ierarhii de memorii tind să fie:

- foarte complexe;
- nepotrivite pentru majoritatea tehnicilor analitice convenţionale;
- puternic influenţate de tehnologia în rapidă evoluţie a calculatoarelor.

./.

2. Obiective ale unui sistem ierarhic de memorii.

Obiectivele propuse prin realizarea unui sistem ierarhic de DM se referă la:

(1) Obţinerea unor indicatori cost/performanţă avantajoşi.

Pe măsură ce tehnicile de arhitectură şi tehnologia calculatoarelor au avansat, a devenit posibilă realizarea unor sisteme de calcul cu viteze foarte mari. Aceste sisteme sînt evaluate în valori de MIPS (milioane de instrucţii pe secundă). Există sisteme experimentale de peste 100 MIPS (ILLIAC IV şi CDC - STAR). Unele sisteme convenţionale de capacitate mare au atins limite de 5 sau 10 MIPS (CDC - 7600 sau IBM 370-195). S-a observat însă că strîns legate de viteza procesorului sînt cerinţele de intrare/ieşire ale unui sistem convenţional. Pe baza unor măsurători experimentale a fost postulat că în cadrul unui sistem de calcul se atinge în medie valoarea de 1 bit intrare/ieşire pentru fiecare instrucţie executată. (Această valoare este des referită ca constanta lui Amdahl(2)). Astfel, multe din sistemele de calcul de înaltă performanţă au fost confruntate cu probleme dificile de "gîtuire" în domeniul intrare/ieşire, în special în cazul cînd aceste cereri tind să apară grupate în pachete. Un sistem ierarhic de memorii este util în această situaţie, reducînd considerabil dimensiunile acestei probleme.

La celălalt capăt al domeniului de capacităţi se pot constata progresele substanţiale realizate în ultimii ani de minicalculatoare (procesoare de cost redus). Termenul "mini" poate fi înşelător. Aceste procesoare sînt, în mod obişnuit, de sute de ori mai rapide decît calculatoarele comerciale timpurii, la o fracţiune de cost. Astfel, minicalculatoarele contemporane operează aproximativ 1 milion adunări de numere a 5 cifre pe secundă, în timp ce UNIVAC I (cca.1951) putea executa cca. 2000 adunări de numere a 12 cifre pe secundă. Datorită progreselor tehnologice şi economiilor de proporţii rezultate din producţia de serie mare, unele minicalculatoare sînt oferite la preţuri mai mici de 2000 dolari, iar microcalculatoare realizate într-o singură pastilă sînt disponibile la preţuri în jur de doar 5 dolari (170). Impactul apariţiei acestor procesoare asupra sistemelor

./.

de calcul este legat însă şi de posibilitatea producerii unor dispozitive de memorii de mare capacitate şi ieftine. Un procesor de 5 dolari nu este interesant dacă costul memoriilor este în domeniul zecilor de mii de dolari.

Prin dezvoltarea unui sistem eficace de ierarhii de memorii se poate avansa pe calea reducerii costurilor DM la nivelul acestor procesoare ieftine. Ca rezultat, un număr mare de soluţii tehnice cunoscute, relative la problemele de tratare a informaţiilor, vor deveni, în sfîrşit, soluţii economic convenabile.

(2) Simplificarea şi automatizarea programării.

Organizarea unui sistem ierarhic de memorii a unui calculator are un impact considerabil asupra arhitecturii programelor şi eficienţei programatorilor. Intr-o mare măsură creşterea posibilă a productivităţii este obţinută prin eliminarea sau reducerea limitărilor impuse în mod obişnuit de sistemele de memorii. Aceste restricţii deseori abat programatorul, în măsura în care el alocă o cantitate substanţială de timp pentru a învinge limitările sistemului, în loc de a rezolva problema propriu-zisă. Astfel, în literatura de specialitate se remarcă faptul că conţinutul de erori al unor programe este deseori dependent de disponibilul de capacitate de memorii. Dacă disponibilul este foarte limitat, autorii software-ului trebuie să recurgă la superpoziţii şi alte artificii de codificare pentru a comprima funcţiile dorite în spaţiul de memorie alocat. Se presupune că aceste artificii introduc o mare complexitate şi sînt locul multor erori. Acest efect este citat de proiectanţii de calculatoare destinate navelor spaţiale, unde alocarea unui alt bloc de memorie de 4 K este o decizie majoră de proiectare.

Mai precis, programatorul trebuie să se îngrijească de:

a) eficienţa codului limbajului de programare;

Dacă un compilator al unui limbaj evoluat tinde să producă programe care sînt considerabil mai mari decît cele produse de către un translator de nivel mai jos, poate deveni necesar să se folosească limbajul de nivel mai jos, pentru a economisi memorie. Această limitare este contrară faptului general acceptat că un limbaj de nivel înalt măreşte productivitatea programării.

./.

b) dimensiunea programului;

Pentru o dimensiune specifică de memorie există programe care nu pot fi scrise uşor pentru a se încadra în această limitare de capacitate. Totuşi programatorii încearcă deseori, dar cu eforturi considerabile.

c) structuri de date;

Programatorul este deseori confruntat cu necesitatea de a alege între o reprezentare a structurii de date care este convenabilă pentru folosire şi o altă reprezentare care economiseşte memorie. Această economisire poate pretinde folosirea unei reprezentări a structurii de date incomode sau complicate.

d) caracteristici de echipamente specifice;

Dacă programatorul trebuie să obţină maximum-ul din sistemul de memorii avut la dispoziţie, în ceea ce priveşte capacitatea şi performanţa, el poate recurge la tehnici care sînt particulare la dispozitivele de memorii respective. Dacă aceste dispozitive de memorii sînt schimbate, poate exista o influenţă considerabilă asupra software-ului.

Ierarhiile de memorii îşi propun să elimine, să automatizeze sau cel puţin să reducă considerabil problemele de programare descrise mai sus.

(3) Crearea posibilităţilor de utilizare a noilor tipuri de dispozitive de memorii şi extinderea domeniului aplicaţiilor.

Pînă nu demult, în utilizarea dispozitivelor de memorii a existat o tendinţă de a raporta aplicaţiile la tehnologiile specifice disponibile. Aceasta a determinat faptul ca anumite domenii de aplicare să fie abandonate ca neconvenabile şi multe strategii de gestiune a memoriilor să fie discreditate ca inaplicabile sau ineficiente. Cîteodată se au în vedere doar aplicaţiile şi tehnicile în uz şi se evită sau se ignoră alternativele posibile şi motivele care au dus la ocolirea acestor alternative.

In prezent sîntem martorii unei creşteri mari a interesului pentru aplicaţii şi tehnologii. Necesităţile noilor aplicaţii se referă la dispozitive de memorii mai mari şi mai rapide, la preţuri mai mici.

Noi direcţii în domeniul tehnologiei pentru dispozitive de memorii, ca şi unele consideraţii de ansamblu asupra acestui

./.

domeniu şi perspectivele lui, fac obiectul a numeroase lucrări şi studii. Astfel Bobeck ş.a.(24 , Chang(34) şi Ypma(169) descriu sisteme cu memorii cu bule magnetice (bubble domain memory) şi posibilităţile acestei tehnologii; Carnes ş.a.(30), Amelio (3) şi Rege(139) descriu memorii utilizînd dispozitive cu transfer de sarcină (charge coupled devices) şi performanţele acestui tip de memorii. Alţi cercetători descriu alte tipuri de memorii: Anacker (4)- memorii bazate pe efectul de superconductibilitate; Gillis ş.a.(85) - memorii holografice, Dell(50) şi Scarrot(140 - memorii optice. De asemenea, sisteme de memorii cu utilizări speciale sînt descrise de Brewer ş.a. (27)- memorii BORAM, Spain ş.a.(154)- memorii DOT, Hughes ş.a.(95)- memorii BEAMOS.

Lucrări de sinteză şi de definire a tendinţelor de dezvoltare în acest domeniu sînt numeroase. Selectăm ca principale următoarele: Bogdan ş.a. (25), Cojanu ş.a.(42) Hoagland(90)şi(91), Hodges(93)şi(94) , Laliotis(107), Mazda(122), Opran(132), Thomson ş.a.(157), Waas(161) şi Williams(166). In bibliografia acestor lucrări pot fi găsite numeroase alte lucrări de interes.

Bazaţi pe caracteristicile tehnice superioare demonstrate în laborator pentru noi tipuri de memorii şi presupunînd condiţii de asimilare eficientă pentru producţia de serie, este posibil să ne aşteptăm la schimbări importante în următorii ani. In tabelul 1 sînt specificate caracteristicile de performanţă, precum şi costul unor tehnologii de memorii fie uzuale în prezent, fie de curînd descrise. Am dorit a da o imagine comparativă a performanţelor, ca şi a costurilor. Astfel se poate observa că tipurile de dispozitive de memorii descrise acoperă aproximativ 8 ordine de mărime pe scala timpului de acces şi aproximativ 4 ordine de mărime pe scala costului. Acest lucru este foarte semnificativ, dacă ţinem seama de importanţa care însoţeşte în mod normal o creştere de 10-20% în performanţă sau o scădere de 10-20% în preţ la sistemele curente contemporane.

./.

NR.	TIP	VOLATIL	MECANIC	ACCES DATE	DIMENS.BITI	COST/BIT	TIM ACC
1	TTL RAM	DA	NU	CUV.ALEATOR	10^3-5.10^5	5 ¢	60
2	MOS RAM	DA	NU	CUV.ALEATOR	4.10^3-10^6	1,25	300
3	RAM - MIEZURI	NU	NU	CUV.ALEATOR	10^5-5.10^7	0,7	500
4	MEMORII CCD	DA	NU	BLOC ALEATOR	10^5-10^8	0,1	config.m 10 config.l 2,5
5	MEMORII BUBBLE	NU	NU	BLOC ALEATOR	5.10^5-2.10^8	0,03-0,05	config.m 0,5 - config.l 1 s
6	DISC FLEXIBIL - TAMBUR	NU	DA	BLOC SERIAL	5.10^5-5.10^6	0,05	100 ms/
7	CASETA MAGN.	NU	DA	BLOC SERIAL	10^6 - 10^7	0,04	2
8	DISC CAPETE FIXE	NU	DA	BLOC SERIAL	10^7 -2.10^8	0,08	8
9	DISC AMOVIBIL	NU	DA	BLOC SERIAL	5.10^7-5.10^9	0,0025	50
10	BANDA MAGN.	NU	DA	BLOC SERIAL	10^8 - 10^{10}	0,0001	10

Tabelul 1. Ierarhie de dispozitive de memorii.

(4) Studiul comportării de acces a programelor şi al comportării sistemelor de calcul.

Comportarea detaliată în exploatare a sistemelor de calcul este deseori foarte complexă. Astfel, unele decizii privind proiectarea hardware, software şi de sistem trebuie deseori făcute pe baza unei cunoaşteri insuficiente. O mai bună înţelegere a comportării de acces a programelor şi a comportării sistemelor de calcul este esenţială pentru dezvoltarea inteligentă şi eficientă a sistemelor viitoare.

3. Parametrii principali ai unui sistem ierarhic de memorii.

Termenul "ierarhie de memorii" a fost utilizat pentru a descrie o varietate de configuraţii de memorii.

Corespunzător modului în care componentele sînt organizate, ierarhiile de memorii pot fi clasificate în două mari categorii:

./.

a) ierarhie centralizată. In această configuraţie o memorie centrală, principală, asigură unitatea centrală de prelucrare (UCP) cu informaţiile pe care aceasta le necesită. Diferitele tipuri de dispozitive de memorii sînt conectate la această memorie principală prin canale şi pot comunica unele cu altele numai prin memoria centrală.

b) ierarhie verticală. In această categorie intră configuraţiile de memorii care sînt aranjate în ordine descrescătoare a vitezelor şi crescătoare a capacităţilor (figura 1).

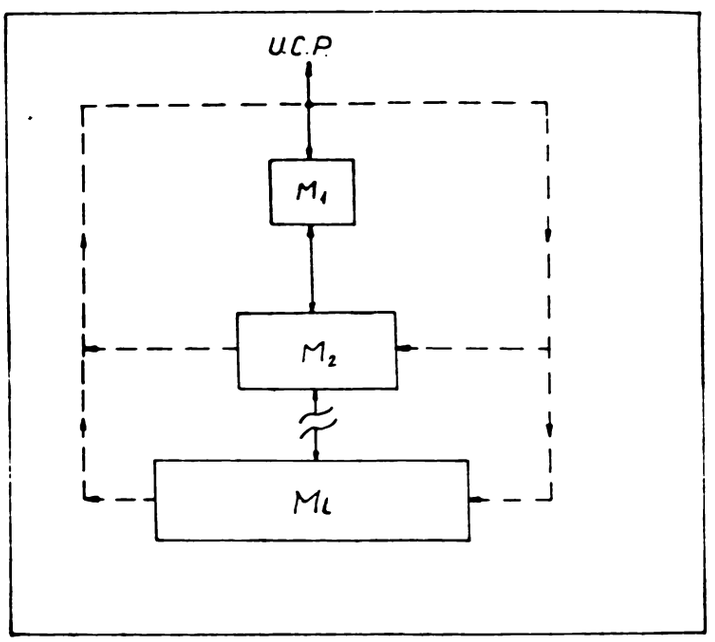


Fig.1. - Ierarhie verticală generalizată de memorii.

In această configuraţie memoriile pot comunica unele cu altele (chiar dacă nu sînt situate pe nivele învecinate) sau cu UCP, după cum este indicat în fig.1 prin linii întrerupte. In general însă, aceste canale transferă puţine informaţii şi poate fi presupus că memoriile pot comunica unele cu altele numai dacă sînt situate pe nivele alăturate. In acelaşi timp numai primul nivel al ierarhiei poate comunica cu UCP. Odată impuse aceste restricţii ierarhia verticală de memorii va fi numită ierarhie lineară verticală de memorii (ILVM).

Iniţial, toate informaţiile rezidă în M_1 şi vor fi deplasate spre partea superioară a ierarhiei, pe măsura solicitărilor UCP. Deoarece $M_1 < M_2 < \ldots < M_l$, M_1 va fi eventual plin;

./.

cu scopul de a plasa o nouă "tranşă" de informaţii în acest nivel, este deci necesar a deplasa un anumit volum de informaţii din M_1 în M_2, lucru ce ar putea conduce la o altă deplasare din M_2 în M_3 etc. Prin urmare informaţiile circulă în timpul desfăşurării calculelor vertical prin ierarhie, în ambele sensuri; informaţiile care sînt utilizate la un anumit moment dat se găsesc în nivelele superioare ale ierarhiei.

Parametrii principali ai unei ILVM pot fi grupaţi în patru categorii: (1) tehnologia de bază, (2) configuraţia, (3) comportarea de acces a programului şi (4) algoritmul.

3.1. Tehnologia de bază.

Parametrii tehnologiei de bază sînt:

b_i costul/bit şi

t_i timp mediu de acces.

Fiecare nivel al ierarhiei este caracterizat de un cuplu de valori (b_i, t_i). Aceşti parametri sînt, evident, dependenţi de nivelul de dezvoltare al tehnologiilor de producere a dispozitivelor de memorii. Acest nivel permite realizarea, la un moment dat, a unui număr limitat de (b_i, t_i) alternative pe care le pune la dispoziţia proiectantului de sistem (a se vedea tabelul 1).

Costul este o funcţie descrescătoare de timpul de acces; curba reprezentată în fig.2 a fost construită pe baza datelor din tabelul 1.

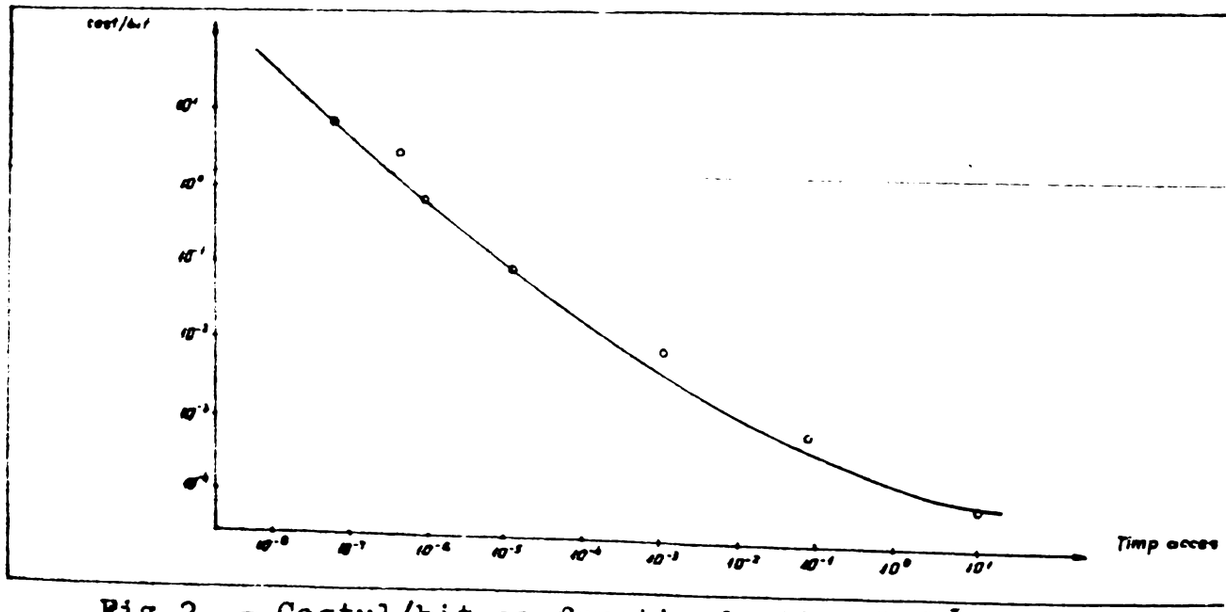


Fig.2. - Costul/bit ca funcţie de timpul de acces.

./.

Timpul de acces este timpul consumat de un DM între momentul primirii unei adrese fizice de înregistrare a unei unităţi informaţionale şi momentul în care această unitate informaţională devine disponibilă.

3.2. Configuraţia.

Parametrii ce intră în această categorie sînt următorii:

l - numărul de nivele,

c_i - capacitatea de memorizare,

r_i - rata de transfer,

p_i - dimensiunea paginii (numărul de octeţi în pagină).

Proiectantul sistemului are o mare flexibilitate în organizarea dispozitivelor de memorii. Prin structurarea în serie şi/sau paralel a componentelor unui anumit nivel al ILVM, este posibil să se realizeze o gamă largă de valori pentru capacitatea memoriei şi rata maximă de transfer. Pe o bază mai largă, proiectantul trebuie să determine numărul nivelelor l în sistemul de memorii şi p dimensiunea unei pagini (unitatea de informaţii transferată între două nivele vecine).

In privinţa alegerii dimensiunii unei pagini există trei strategii de bază:

(1) alegerea unei singure valori pentru dimensiunea paginii p, care este folosită peste tot în cadrul ierarhiei. Această abordare este folosită în cadrul majorităţii sistemelor cu gestiune automată cu mai multe nivele contemporane;

(2) posibilitatea de a alege un număr arbitrar de valori ale lui p. Această abordare este specifică sistemelor de memorii cu gestiune manuală;

(3) determinarea a l valori pentru p; în cadrul transferului de informaţii între două nivele vecine ale ILVM fiind utilizată o dimensiune proprie de pagină.

3.3. Comportarea de acces a programului.

UCP, sub controlul programului, produce o serie secvenţială de referiri la sistemul de memorii. Aceste referiri ale procesorului sînt sub forma unor referiri la adrese logice, ce servesc pentru a identifica în mod unic fiecare unitate individuală de informaţii memorată, indiferent de localizarea ei ($X_1, X_2, \ldots X_n$). Secvenţa în timp a referirilor de adrese logice este denumită şirul de referiri. Un asemenea şir este notat astfel:

$$Z = z_1, z_2, \ldots, z_\lambda$$

unde:

z_t - numele adresei referită la momentul t şi λ este lungimea şirului

In general, fiecare program, împreună cu datele lui de intrare, va genera un şir distinct de referiri. De remarcat că în această situaţie nu sîntem interesaţi de scopul programului sau limbajul utilizat la scrierea lui, ci doar de şirul lui de referiri.

Comportarea de acces a programului, observată din punctul de vedere al ierarhiei de memorii, este definită de secvenţa combinată de adrese de instrucţii şi de date emise de către program.

Deoarece deplasarea informaţiilor în cadrul ILVM este realizată prin transferul de pagini, ca problemă de interes apare analiza acestui transfer de pagini. Acest lucru se poate realiza considerînd secvenţa în timp a referirilor la pagini logice; această secvenţă este numită şirul de referiri de pagină. Un şir de referiri de pagină, pentru valori date ale dimensiunilor p_i ale paginilor, este:

$$X(i) = x_1(i), x_2(i), \ldots\ldots, x_{\lambda_i}(i)$$

unde:

$x_t(i)$ este numele paginii care conţine z_t.

In cadrul ILVM, referirile sînt trecute spre un nivel mai inferior, numai dacă ele nu au putut fi satisfăcute de nivelele mai înalte. De aici rezultă că fiecare nivel va avea un şir diferit de referiri de pagină; toate aceste şiruri sînt derivabile algoritmic din acelaşi şir al referirilor.

Două concepte utilizate frecvent în studiul comportării unui program într-o ILVM sînt conceptele de spaţiu al numelor şi de proces.

3.3.1. Spaţiul numelor.

Spaţiul de nume (sau spaţiul adreselor logice) reprezintă mulţimea numelor (a adreselor logice) disponibile unui procesor pentru a fi folosite ca identificatori de date.

Adresele logice sînt adresele folosite de programatori la scrierea unui program.

./.

Un spaţiu de adresare este caracterizat prin capacitate de adresare şi structură. Capacitatea de adresare este dată de mulţimea adreselor disponibile, cu alte cuvinte de spaţiul de adresare fizică.

Structura spaţiului de adresare influenţează modul de scriere al programelor.

Tipul cel mai frecvent de structură este cea liniară: într-un spaţiu linear de nume numele permise sînt întregi 0,1,..N. Capacitatea acestui tip de spaţiu este determinată de numărul de biţi utilizaţi pentru reprezentarea adreselor absolute.

Un alt tip comun de spaţiu de nume este cel segmentat.

Un spaţiu segmentat de nume este compus dintr-o mulţime de spaţii lineare distincte de nume. Fiecare spaţiu linear de nume este utilizat pentru a adresa o mulţime ordonată de elemente de informaţie, ce au fost declarate ca formînd un segment. Un spaţiu segmentat de nume este deseori descris ca un spaţiu bidimensional de nume, deoarece pentru a adresa un element de informaţie este necesar să se specifice două nume (numele segmentului şi numele elementului în segment). Diferite segmente pot avea dimensiuni diferite.

Segmentarea realizează adresarea relativă în spaţiul de nume, prin diviziunea spaţiului de nume în locaţii contigue (segmente) şi considerarea adreselor din interiorul unui segment ca fiind adrese relative la începutul acestuia.

Avantajele aduse de segmentare decurg din faptul că segmentul reprezintă o notaţie comodă de nivel înalt pentru crearea unei structurări semnificante a informaţiilor utilizate de un program (138). Astfel, între avantajele utilizării acestei structuri se numără:

(1) segmentele formează o unitate foarte comodă pentru realizarea diviziunii informaţiilor între programe.

Proceduri sau rutine reutilizabile (ce nu pot fi modificate) pot fi adresate simultan de cîţiva utilizatori. Din cauza segmentării lucrărilor independente este posibil să se creeze şi să se folosească informaţii dintr-o bază comună; prin limitarea privilegiilor de acces un segment poate fi protejat de utilizatori neautorizaţi.

./.

(2) se creează posibilitatea modularizării. Secţiuni ale unor lucrări mari pot fi identificate, compilate şi executate independent. Uneori o lucrare mare poate fi constituită din rutine scrise independent. Astfel de module de programe sînt legate în momentul execuţiei.

(3) utilizarea unui spaţiu segmentat de nume creează facilităţi în manipularea masivelor de date. Astfel, pentru fiecare mulţime de elemente de informaţie, a cărei dimensiune variază dinamic, se poate folosi un segment distinct.

Spaţiul segmentat de nume poate fi clasificat corespunzător caracteristicilor spaţiului prevăzut pentru nume de segmente, adică pentru primul termen al perechii ordonate (nume de segment, numele elementului în cadrul segmentului). Astfel, pentru nume de segmente se pot folosi spaţii de nume linear sau simbolic. In consecinţă se vor deosebi spaţiul segmentat linear de nume şi spaţiul segmentat simbolic de nume.

Diferenţa principală între aceste două tipuri de spaţii segmentate de nume este că în cadrul spaţiului segmentat simbolic segmentele nu sînt ordonate în nici o privinţă, dat fiind că utilizatorii nu sînt înzestraţi cu nici un mijloc de manipulare a unui nume de segment pentru a produce alt nume. Lipsa ordonării înseamnă că nu există o contiguitate a numelor (adică nu există nici o relaţie între ultima adresă a unui segment N şi prima adresă a segmentului N + 1), ce să provoace genul de probleme care sînt prezente în cadrul procedurilor de alocare a memoriei şi relocare a adreselor (138). In consecinţă, un spaţiu segmentat simbolic de nume implică proceduri mai simple de administrare decît spaţiul segmentat linear.

Un avantaj posibil al utilizării unui spaţiu segmentat linear de nume este faptul că el permite indexarea pentru numele de segmente. Aceasta deoarece în cadrul segmentării lineare, la trecerea dintr-un segment în altul, adresele sînt în continuare. Ultima adresă a segmentului N este urmată de prima adresă a segmentului (N + 1); deci este posibilă modificarea prin indexare a unei adrese din segmentul N pentru obţinerea unei adrese din segmentul (N + 1).

./.

Exemple de sisteme ce utilizează spaţii segmentate simbolic de nume sînt: Multics - GE 645 (48; 133), iar sisteme ce utilizează spaţii segmentate linear de nume sînt: IBM - 360/67 (83), IBM - 370 (98), RCA-SPECTRA 70/46 (131).

Intre numele din spaţiul numelor şi locaţiile din memoria fizică nu există nici o realţie apriorică; această corespondenţă se stabileşte în cadrul mecanismului de translatare a adreselor (7) (fig.3.).

In cadrul unor sisteme de calcul ce utilizează un spaţiu segmentat de nume, segmentele reprezintă şi unitatea de alocare fizică a memoriei. Aşa este cazul sistemului Burroughs (117).

Mai frecvent însă, pentru a simplifica transpunerea segmentelor de dimensiune arbitrară într-o memorie de dimensiune fixă, cît şi pentru utilizarea mai eficientă a memoriei interne, segmentele şi memoria internă sînt împărţite în blocuri egale ca dimensiune, numite pagini. Pagina, transparentă pentru programator, este unitatea standard de memorare şi transmisie a informaţiilor. Spaţiul memoriei interne este împărţit în cadre de pagină. Segmentele de lucrare, aşa cum sînt primite de la utilizatori, sînt încărcate în nivelul 1 al ierarhiei de memorii M_1; în timpul executării programului, pagini individuale ale unui segment sînt transferate spre partea superioară a ILVM, ajungînd în cadre de pagină ale memoriei principale M_1. Această metodă este numită paginare şi se referă deci la organizarea şi alocarea spaţiului fizic de memorare.

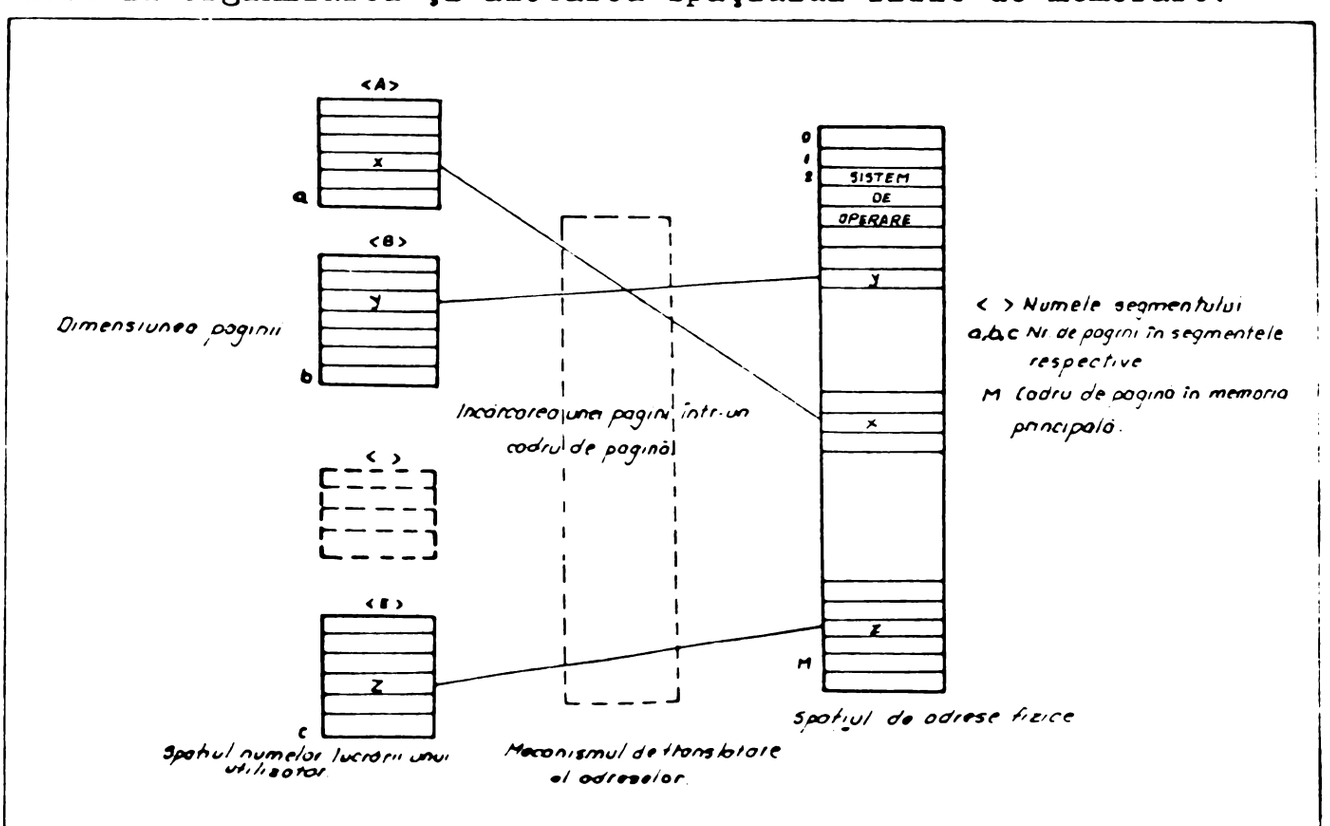


Fig.3 Reprezentarea spaţiului numelor şi al adreselor fizice

Metoda paginării se poate referi şi numai la memoria internă; în acest caz se realizează o administrare mai simplă a memoriei interne, iar alocarea zonelor de memorie pentru programe se face prin alte metode (25). In cazul sistemelor cu memorii virtuale, paginarea se referă la ansamblul de dispozitive de memorii disponibil, operînd în cadrul ierarhiei.

Traducerea adreselor din spaţiul numelor în adrese fizice este denumită mecanismul de translatare a adreselor.

In continuare, vom exemplifica acest mecanism de translatare a adreselor în cadrul schemei de segmentare cu paginare utilizate în sistemul Multics (48). Pentru adresarea memoriei trebuie cunoscut numărul segmentului şi adresa (relativă) din cadrul segmentului: această formă a adresei se numeşte adresă generalizată. Pentru fiecare program de utilizator din sistem există un tabel descriptor de segmente (TDS); acest tabel conţine cîte un cuvînt denumit cuvînt descriptor de segment (CDS) pentru fiecare segment cunoscut programului respectiv. Pentru un program în execuţie, adresa de început a tabelului descriptor de segmente (TDS) este conţinută în registrul de bază pentru un segment descriptor (RBD). Adresa din cuvîntul descriptor de segment (CDS) reprezintă adresa de început a unui tabel de pagini, ce conţine cuvinte descriptor de pagini (CDP), ce compun acel segment. Descriptorul unei pagini conţine adresa fizică de început a paginii respective şi informaţii cu privire la pagină (existenţa sau absenţa în memoria internă, iar în cazul existenţei, dacă a fost sau nu modificată de la aducerea în memoria internă etc.).

O asemenea schemă este arătată în fig.4.

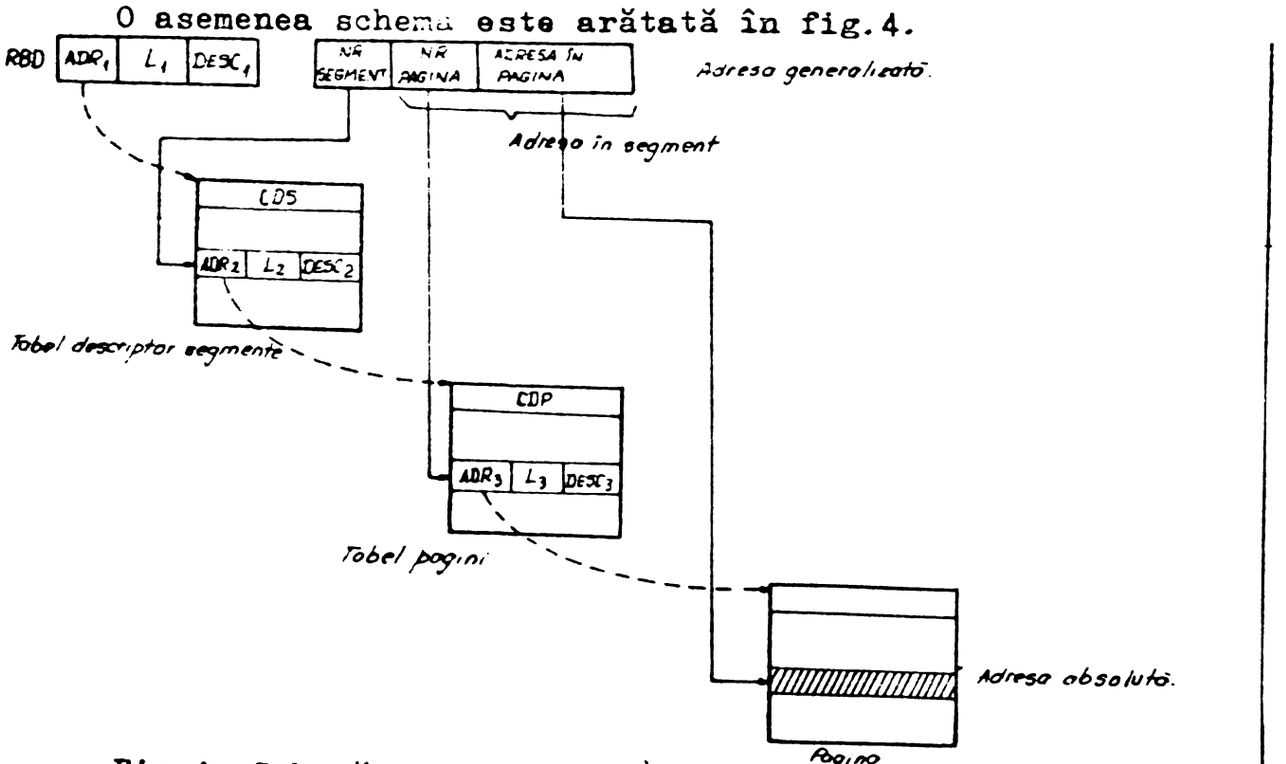


Fig.4. Schemă de translatare a adreselor cu două nivele

Aceeaşi schemă de bază de translatare a adreselor indicată în fig.4 se utilizează şi în cadrul sistemelor 360/67 şi 370 (138,96).

Un mecanism de paginare mai complicat este întîlnit în cadrul configuraţiilor DEC System 20 (106), permiţînd sistemului de operare să realizeze o administrare mai eficientă a programelor pe baza paginării la cerere (vezi secţiunea 3.4 a acestui capitol) şi o înaltă diviziune a datelor şi programelor. Acest mecanism este realizat printr-o combinaţie de hardware şi microcod, numită facilitate de paginare. Această facilitate utilizează urmărirea adreselor de pagină pe mai multe nivele.

Un spaţiu de adrese de 256 Kcuv. (1 cuv.are 36 biţi) formează o secţiune, care este identificată printr-un indicator de secţiune. Un indicator de secţiune indică un tabel de pagini, care la rîndul lui conţine indicatori reprezentînd fiecare pagină a acelei secţiuni. Indicatorul de secţiune poate fi de format imediat, cu diviziune sau indirect; în fiecare caz el indică adresa din memoria principală a unei pagini, care conţine un tabel de pagini. Un tabel de pagini este o pagină ce conţine indicatori de pagină pentru o secţiune. Intr-o secţiune există 512 pagini, iar fiecare indicator este un cuvînt. Avînd în vedere că dimensiunea unei pagini este de 512 cuvinte, rezultă că un tabel de pagini este o pagină completă.

Un indicator imediat conţine adresa fizică a paginii corespunzătoare. Acest tip de indicator este denumit de asemenea propriu, deoarece pagina este proprie doar tabelului de pagini ce conţine acest indicator.

Un indicator cu diviziune conţine un index ce indică locul tabelului de pagini speciale/ cu diviziune (SPT). Intrarea corespunzătoare din SPT conţine adresa fizică pentru pagină. Intrarea SPT este găsită la adresa de memorie fizică obţinută prin sumarea conţinutului registrului de bază SPT şi a indexului SPT din indicatorul cu diviziune. Acest indicator este numit cu diviziune, deoarece mai multe tabele de pagini pot conţine asemenea indicatori, care să se refere la aceeaşi pagină fizică. Indiferent de numărul de tabele de pagini ce conţin un asemenea indicator cu diviziune, adresa fizică este înregistrată o singură dată în SPT.

./.

De aceea monitorul poate deplasa pagina, avînd de actualizat numai o singură adresă. Un indicator indirect identifică un alt tabel de pagini şi un indicator în cadrul acelui tabel de pagini. El determină extragerea şi interpretarea noului indicator.

Indicatorul indirect este folosit pentru a face o pagină a unui spaţiu de adrese echivalentă unei pagini a altui spaţiu de adrese. Indicatorul indirect identifică tabele obiect de pagini, folosind un index SPT. Adresa fizică a tabelului de pagini este găsită la intrarea corespunzătoare din SPT la fel ca în cazul indicatorului cu diviziune. Acesta este realizat astfel încît adresa fizică a unui tabel de pagini să poată fi conţinută într-un singur loc şi să nu necesite a fi duplicată în indicatorii de pagină. Indată ce tabelul obiect de pagină a fost găsit, cîmpul cu numărul paginii din cadrul indicatorului indirect este utilizat ca index pentru a selecta un nou cuvînt indicator din tabelul de pagini. Acest nou indicator poate fi de oricare tip din cele trei tipuri, facilitatea de paginare putînd interpreta lanţuri de indicatori indirecţi cu o lungime oarecare.

Un tabel cu pagini speciale/cu diviziune SPT conţine adrese fizice pentru pagini ce sînt folosite de mai multe tabele de pagini sau care sînt folosite într-un mod special, de exemplu tabele de pagini. Un registru conţine adresa de bază a SPT. Indexul SPT găsit în indicatori este adunat la adresa de bază a SPT pentru a forma adresa de memorie fizică a intrării corespunzătoare. Intrarea SPT conţine o adresă fizică.

Fluxul de interpretare al indicatorilor este descris în (106).

Prezenţa sau absenţa unei anumite pagini în memoria principală este precizată de starea cîmpului (poziţionat 1 sau 0) de prezenţă a paginii în memoria principală. Un asemenea cîmp este asociat fiecărei intrări a tabelului de pagini.

In cazul în care procesorul încearcă să adreseze o pagină al cărui cîmp de prezenţă în memorie este poziţionat 0 (indicînd deci lipsa ei), se generează prin hardware o între-

rupere automată a execuţiei programului în curs de desfăşurare, iar procesorul iniţiază acţiunea de aducere a paginii lipsă în memoria principală dintr-un nivel inferior al ILVM. O asemenea situaţie este numită defect de pagină (page-fault). După ce pagina cerută a fost plasată în memoria principală fiind utilizabilă, intrarea corespunzătoare în tabelul de pagini este actualizată; cîmpul de adresă va conţine adresa memoriei principale unde începe cadrul de pagină alocat, iar cîmpul de prezenţă în memorie este poziţionat 1. In acest moment este reluată execuţia întreruptă a programului care a necesitat aducerea acestei pagini. Mai tîrziu, cînd pagina va fi scoasă din memoria principală, cîmpul de prezenţă al acestei pagini va fi din nou poziţionat 0.

Translatarea adreselor logice în adrese fizice este o operaţie ce implică deci mai multe consultări de tabele din memorie, ceea ce necesită un anumit timp. Pentru a scurta acest timp de consultare se pot utiliza registre sau memorii imediate rapide, pentru memorarea tabelelor de pagini. Această soluţie este convenabilă numai pentru calculatoarele de capacităţi mici, respectiv cu un număr mic de pagini fizice. O soluţie eficientă în cazul calculatoarelor de capacităţi medii şi mari este utilizarea unor memorii (registre) asociative (adresabile prin conţinut), în care se memorează atît adresa virtuală, cît şi cea reală a unor pagini de program selectate după anumite reguli (vezi secţiunea 3.4 - algoritm - a acestui capitol). In alocarea registrelor asociative se pot adopta diferite soluţii. Cele alese pentru sistemele Multics şi IBM 360-67 sînt descrise în (134) şi respectiv (83).

In cazul utilizării unor memorii asociative, la translatarea unei adrese virtuale se cercetează mai întîi aceste registre asociative. In cazul în care adresa virtuală căutată este memorată în unul din aceste registre, din acelaşi registru se poate obţine şi adresa fizică a începutului cadrului de pagină corespunzător. In cazul că în registrele asociative nu se găseşte adresa virtuală căutată, va fi necesară consultarea tabelelor din memorie. Scurtarea timpilor de căutare se realizează pe seama timpului de acces mai mic al registrelor asociative, com-

./.

parativ cu cel al memoriei principale.

Utilizarea registrelor asociative poate fi realizată pentru translatarea adreselor unui spaţiu de nume, atît unidimensional, cît şi bidimensional. In primul caz se utilizează memorii asociative cu o singură trecere (cazul sistemului IBM 360-67); în cel de-al doilea caz este necesară utilizarea unor memorii asociative cu două treceri: (sistemul Multics): la prima trecere se urmăreşte identificarea numelui segmentului, la cea de-a doua numărul paginii în cadrul segmentului (133).

O tratare mai largă a conceptului de segmentare şi a procedurii de paginare poate fi găsită în (138) şi (25).

Trebuie remarcat că, din punct de vedere teoretic, orice sistem ar putea implementa o ILVM în software. Ceea ce ar trebui făcut este aplicarea unor "capcane" pentru adrese şi crearea unor tabele de referinţă şi a unui program de evidenţă a corespondenţelor adreselor. Intr-un sens mai riguros, se precizează că un sistem conţine o ILVM cînd aspecte importante funcţionale ale unui asemenea sistem au fost implementate prin hardware. Astfel, adresarea virtuală modifică structurile de adresare reală prin:

- permiterea canalelor I/E de a folosi adresarea indirectă;
- acceptarea de către UCP a adreselor de date ce depăşesc capacitatea lui de memorie reală;
- asigurînd o schemă de translatare a adreselor din virtuală în reală şi invers.

In cazul IBM - System 370 aceste lucruri se realizează prin facilităţile:

- channel indirect data addressing;
- extended core mode;
- dynamic address translation.

In cazul acestui sistem, memoria paginată externă (ajungînd la o capacitate de 16 Mo) utilizează segmente de 16 Ko sau 1 Mo şi pagini de 2 Ko sau 4 Ko (1 Ko = 1024 octeţi). Diviziunile folosite sînt funcţie de alegerea sistemului de operare. Partea de translatare dinamică a adreselor este implementată hardware, iar evidenţa corespondenţelor paginilor este sub su-

./.

pravegherea sistemului de operare. Procesul de translatare a adreselor şi de evidenţă a corespondenţelor paginilor sînt transparente pentru utilizator, ce operează simplu cu o structură logică obişnuită de adrese.

O realizare deosebită în domeniu este facilitatea de extensie a spaţiului de adrese virtuale în sistemul DEC 10 (106). Această facilitate de adresare extinsă măreşte spaţiul de adrese virtuale de utilizator la 1 bilion de cuvinte, permiţînd programelor de utilizator să adreseze pînă la 4096 secţiuni, fiecare a 256 Kcuvinte. Fiecare instrucţie specifică o adresă de 30 biţi, implicit, sau explicit. Programatorul poate utiliza secţiunile ca entităţi logice separate sau ca un spaţiu unic de adresare contiguu. Masivele şi şirurile de date pot fi arbitrar de lungi şi pot traversa graniţele secţiunilor. Insă şirul de instrucţii trebuie să transfere controlul în mod explicit între secţiuni şi nu poate trece graniţele de secţiuni. Cu scopul de a specifica adrese de 30 biţi, indexarea şi adresarea indirectă a calculatorului au fost modificate (106) .

3.3.2. Proces.

Procesul este noţiunea prin care se defineşte execuţia unui program de către un procesor într-un mediu cu memorie virtuală.

O definire echivalentă a procesului este aceea conform căreia un proces este o secvenţă ordonată de referiri la informaţiile înmagazinate în spaţiul numelor, sub controlul unui flux de instrucţii.

Un proces are patru stări de existenţă în timp real:

(1) în desfăşurare, înţelegînd prin acest lucru că el obţine posibilitatea de utilizare a procesorului;

(2) gata de desfăşurare; procesul cere, dar nu obţine utilizarea procesorului; cu alte cuvinte procesul este întrerupt din cauză că procesorul este utilizat de un alt proces;

(3) în aşteptarea unei pagini; procesul este temporar întrerupt din cauză că lipseşte o pagină din M_1. Execuţia este reluată imediat ce pagina lipsă a fost plasată în M_1 şi procesorul este disponibil;

./.

(4) blocat; procesul nu cere utilizarea procesorului, din cauză că aşteaptă producerea unor evenimente externe (aşteptate), astfel ca un mesaj sau un semnal de la alt proces, de la un aparat sau de la un utilizator aflat la o consolă.

Figura 5 ilustrează tranziţiile posibile între aceste stări.

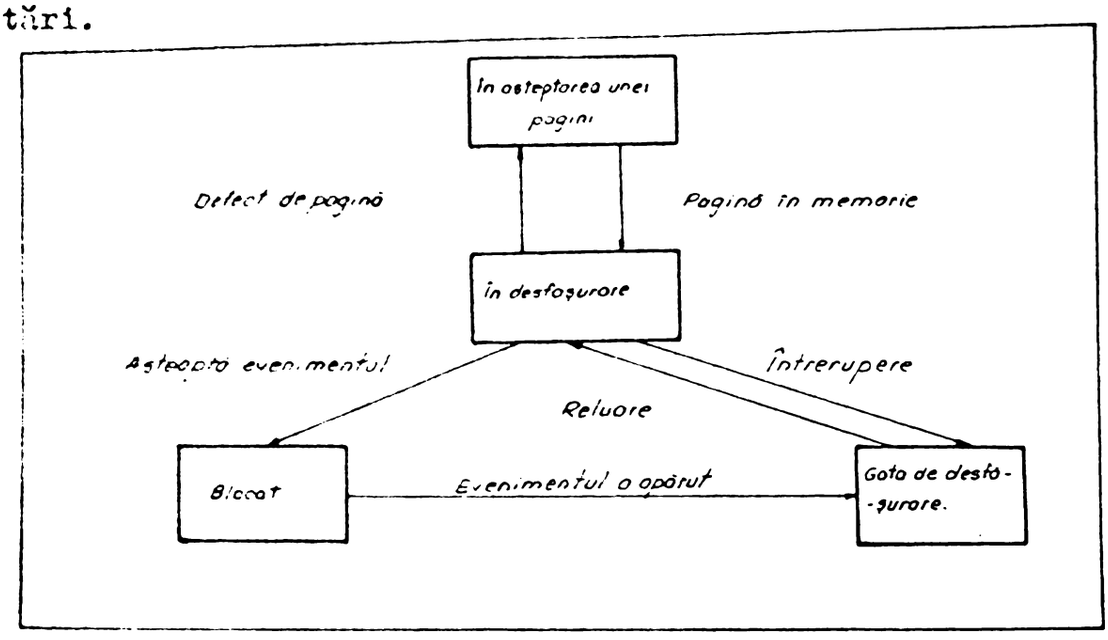


Fig.5. Stări ale unui proces.

Tranziţia de la starea "în desfăşurare" la cea "gata de desfăşurare" apare ca urmare a unei întreruperi decise de sistemul de operare cu scopul de a asigna procesorul unei alte utilizări, de exemplu execuţiei unui alt proces.

Tranziţia de la starea "gata de desfăşurare" la cea de "în desfăşurare" - specificată ca reluare - apare ca urmare a deciziei sistemului de operare de a aloca din nou procesorul acestui proces.

Tranziţia arătată în fig.5 de la starea "în aşteptare a unei pagini" la cea de "în desfăşurare" are corespondent în realitate situaţiei existenţei unui procesor asignat permanent acestui proces, ce ar fi imediat disponibil continuării execuţiei, în momentul cînd procesul a obţinut pagina găsită lipsă. Dacă însă timpul de aşteptare pentru a obţine pagina lipsă este mai mare decît timpul necesar comutării procesorului altui proces, nu este economic a asigna procesorul unui singur proces şi în acest caz procesul se întoarce în starea de "în

./.

desfăşurare" prin starea de "gata de desfăşurare" .

Vorbind despre procese, facem o distincţie între timpul virtual şi timpul real. Timpul virtual este timpul observat de către un proces, dacă nu ar fi niciodată întrerupt, deci timpul total acumulat în starea de desfăşurare. Timpul virtual, numit de asemenea timp de execuţie sau timp de proces, este măsurat în unităţi de timp virtual, de obicei cicluri ale memoriei M_1.

O unitate de timp virtual este intervalul între două referiri succesive la informaţiile ce constituie un proces. Vom considera timpul virtual ca fiind continuu, deşi în realitate este discontinuu, dar intervalul între două valori consecutive este foarte mic.

Timpul real este timpul virtual, la care se inserteazâ corespunzător întîrzierile ce se referă la stările de "în aşteptarea paginii", "blocat" şi "gata de desfăşurare" .

Deoarece un proces este o secvenţă ordonată de referiri la informaţii el este descori numit un proces secvenţial. Intr-un sistem de calcul multiprocesor, pot fi executate simultan mai multe procese; avem de-a face cu procese secvenţial-paralele.

In interesul generalităţii programării şi de asemenea cu scopul de a asigura o protecţie totală a fiecărui utilizator, sistemele moderne cu memorii virtuale oferă facilitatea unui calculator virtual (99, 100). Un calculator virtual este un echivalent simulat al unui calculator real şi constă din UCP virtuală, memorie virtuală şi dispozitive I/E virtuale. Sistemul de operare are sarcina de a simula UCP, memoria şi dispozitivele I/E.

In sistemul Multics, mecanismul de control al traficului (133) tratează asignarea procesorilor reali la procesori virtuali şi comunicarea între procesorii virtuali.

In cadrul IBM - System 370 Virtual Machine Facility - VM/370 (99) permite utilizatorilor să dezvolte calculatoare virtuale. VM/370 alocă resursele reale ale sistemului utilizatorilor prin modul de diviziune al calculatorului. Utilizatorii primesc frecvent intervale de timp de UCP şi îşi împart memoria reală a sistemului prin utilizarea paginării la cerere (vezi secţiunea 3.4. a acestui capitol). Componenta principală a VM/370 - Control Program - administrează resursele System/370, inclusiv timpul UCP,

./.

pentru a crea şi controla multiple calculatoare virtuale simultane, ce pot rula sub diferite sisteme de operare.

Deşi programul de utilizator este în mod tradiţional gîndit ca un ansamblu de elemente componente - module de instrucţii de program, masive de date, proceduri şi subrutine - programul nu poate rula izolat într-un calculator. Programul necesită resurse software, precum tratarea întreruperilor, a cererilor de serviri I/E etc. In trecut acest software nu era considerat ca fiind o parte a programului, în mod logic însă aceste două părţi interacţionează şi formează un ansamblu unitar complet. Un asemenea ansamblu poate fi executat într-un calculator virtual, care asigură utilizatorului toate resursele sistemului de care acesta are nevoie (100). Fiecărui utilizator îi apare să aibă acces la disponibilităţile funcţionale complete ale unui asemenea sistem.

3.4. Algoritmul.

Gestiunea unei ierarhii de memorii are rolul de a decide cum trebuie organizată informaţia în cadrul ierarhiei între nivelele acesteia şi în cadrul fiecărui nivel şi a decide cînd să se transfere informaţia între nivelele ierarhiei. Algoritmii utilizaţi în transferul de pagini între nivelele ierarhiei operează în două sfere.

(1) aducere (plasare): localizează pagina cerută într-un nivel al ierarhiei şi o încarcă în memoria principală M_1, poziţionează 1 bitul de prezenţă a respectivei pagini în intrarea corespunzătoare a tabelului de pagini;

(2) înlocuire (replasare):înlocuieşte o pagină din M_1, poziţionează 0 bitul de prezenţă a intrării respective a tabelului de pagini.

Regula utilizată pentru a decide ce pagină trebuie înlocuită este numită algoritm de înlocuire.

Algoritmii de gestiune pot încărca pagini în M_1 înainte de a fi necesitate (pre-paginare), la momentul în care sînt necesitate (paginare la cerere) sau chiar mai tîrziu.

Multe strategii utilizează paginarea la cerere, adică nu este întreprinsă nici o acţiune de aducere a unei pagini în memoria principală pînă ce un proces nu face o referire la

./.

ea. Paginarea la cerere este de obicei preferată pre-paginării, căci costul implementării este mai mic şi deoarece pre-paginarea nu îmbunătăţeşte considerabil performanţa.

După cum s-a arătat mai sus, în mod obişnuit nu există informaţii anterioare relative la problemele de alocare a resurselor. Singurul argument important favorizînd pre-paginarea este posibilitatea deplasării unor blocuri mari continue de pagini din nivelele inferioare ale ierarhiei, astfel că timpul de acces cumulat este mai mic într-o rulare lungă.

Problema majoră în cadrul gestiunii unei ILVM nu este a decide care pagină să fie încărcată, ci a decide ce pagină să fie înlocuită. Un algoritm de gestiune al ILVM ar trebui să încerce să menţină în M_1 paginile care sînt cele mai probabile de a fi utilizate. Astfel, cea mai bună alegere pentru înlocuire o oferă pagina cu cea mai mică probabilitate de a fi utilizată din nou, imediat.

In continuare vom prezenta comparativ strategiile curente de înlocuire.

Dacă o pagină a fost modificată de la aducerea ei în M_1 înlocuirea ei implică transferarea ei într-un nivel M_i; o pagină nemodificată este pur şi simplu ştearsă, căci presupunem că există o copie a ei într-un nivel M_i.

Alocarea de pagini în nivelul M_1 al unei ILVM în cadrul unui sistem ce permite multiprogramarea (cum sînt majoritatea sistemelor moderne contemporane) poate fi tratată pe baza unei diviziuni fixe sau variabile:

(1) diviziunea fixă; înainte de a fi rulat, unui program îi este garantată o diviziune fixă a memoriei M_1 pentru uzul privat;

(2) diviziunea variabilă; programelor le este permisă libera concurenţă pentru spaţiul de memorie. In aceste condiţii, necesităţile unui program, apărute în cursul desfăşurării lui, pentru un spaţiu de memorie mai mare sau mai mic, conduc la alocarea acestui program a unei cote părţi (diviziune) din memorie, mai mare sau, corespunzător, mai mică.

Intr-un context de elemente identice, se poate arăta că

./.

strategiile cu diviziunea variabilă a memoriei conduc la probabilităţi de lipsă a unor pagini (vezi secţiunea 4.1.) mai mici decît cele în cazul strategiilor cu diviziune fixă.

Algoritmii de înlocuire (care pot fi utilizaţi cu strategii de bază atît cu diviziunea fixă, cît şi cu diviziunea variabilă a memoriei) se împart în următoarele trei clase, ordonate în termenii creşterii costurilor, aferente logicii cerute pentru implementare.

(1) algoritmi statici - înlocuirea este făcută conform unei reguli predeterminate; ei nu folosesc nici o informaţie privind utilizarea paginilor; aceste reguli sînt simplu de implementat;

(2) algoritmi de utilizare ce folosesc informaţii privind utilizarea paginilor, în general măsurînd intervalele de timp de la ultima referire la fiecare pagină;

(3) algoritmi de cerere ce încearcă să predetermine, pe baza mostrelor de referiri curente, grupul de pagini cel mai probabil de a fi folosit imediat.

Unui program îi este acordat mai mult sau mai puţin spaţiu corespunzător cererii lui pentru spaţiu.

Se poate arăta (130) că în situaţii identice, algoritmii statici conduc la probabilităţile de lipsă a unei pagini cele mai ridicate, şi că algoritmii de cerere la probabilităţile cele mai mici.

Există doi algoritmi statici de interes:

(1) aleator (RAND). Ori de cîte ori este necesitată o pagină nouă în memorie, o pagină este selectată aleator pentru a fi înlocuită.

Implementarea este simplă, necesitînd numai un generator de numere aleatoare.

(2) primul intrat, primul ieşit (FIFO). Ori de cîte ori este necesitată o pagină nouă în memorie, este solicitată pentru a fi înlocuită pagina care a fost adusă în memorie cel mai puţin recent (adică de intervalul cel mai lung de timp). In timp ce algoritmul RAND implică un generator de numere aleatoare, FIFO cere un contor şi implementarea este chiar mai simplă, după cum urmează.

./.

Paginile nivelului M_1 sînt considerate un grup ciclic; presupunem că cele N pagini ale lui M_1 sînt numerotate 0,1,...,N-1 şi un punctator k arată că pagina k a fost cea mai puţin recent adusă în M_1. Cînd este cerută o nouă pagină, (k+1) mod N → k şi pagina k este înlocuită.

Principalul argument pentru eventuala utilizare a acestor doi algoritmi statici este simplitatea lor de implementare, însă rezultate experimentale (18,19,159) arată că algoritmii de utilizare, în ciuda unor costuri mai ridicate, depăşesc considerabil în performanţă algoritmii statici.

Există următorii algoritmi de utilizare de interes:

(1) cel mai puţin recent utilizat (least recently used-LRU). Ori de cîte ori este necesară o nouă pagină în memorie, este înlocuită pagina nereferită de cel mai lung interval de timp.

Implementarea este posibilă în felul următor:

Fiecare intrare în tabelul de adrese al paginilor conţine un bit de utilizare, care este poziţionat 1 la fiecare nouă referire a paginii respective. La intervalele periodice, toate intrările tabelului de pagini sînt cercetate, biţii de utilizare poziţionaţi 0 şi înregistrările de utilizare actualizate. La apariţia unui defect de pagină, este solicitată pentru înlocuire pagina a cărei înregistrare de utilizare specifică intervalul cel mai lung de timp de la ultima referire;

(2) cel mai puţin frecvent utilizat (least frequently used-LFU).

Intr-o manieră asemănătoare celei descrise în cadrul algoritmului LRU, paginile sînt organizate conform frecvenţei referirii şi algoritmul selectează pentru înlocuire pagina care a fost referită de numărul cel mai mic de ori;

(3) primul adus, nefolosit, primul înlocuit (first in not used, first out - FINUFO) (133).

Implementarea este aproximativ identică ca în cazul algoritmului FIFO, cu deosebire că în cadrul acestui algoritm se folosesc şi biţii de utilizare din tabelul de pagini.

Fie k punctatorul ce ciclează pe cele N pagini ale memoriei M_1. Ori de cîte ori apare un defect de pagină, k este incrementat pînă ce este găsită o pagină al cărei bit de utilizare

./.

este poziţionat 0; această pagină este selectată pentru înlocuire. Cînd k traversează o pagină al cărei bit de utilizare este poziţionat 1, acest bit este comutat.

Acest algoritm ce combină simplitatea lui FIFO cu caracterul mai sofisticat al LRU a fost propus pentru sistemul Multics.

(4) detectare de cicluri ATLAS

Calculatorul Ferranti-Atlas (105) are o strategie de paginare ce încearcă să detecteze comportarea ciclică în mostrele de referire a paginilor.

Acest lucru se realizează prin supravegherea lungimii de inactivitate (pericade de timp fără referiri) pentru fiecare pagină; pa baza acestor valori sînt presupuse valorile următoarei perioade de inactivitate. Inlocuirea se îndreaptă spre acele pagini care au o valoare mare presupusă de inactivitate; obiectivul principal al acestui algoritm a fost să detecteze cicluri în programe cu scopul de a încerca să memoreze blocurile pe care acesta le conţine.

Performanţa a fost satisfăcătoare pentru programe ce manifestă o comportare de acces ciclică şi nesatisfăcătoare pentru programe ce manifestă un caracter aperiodic în mostrele de referiri la memorie din cauză că algoritmul încerca să detecteze cicluri acolo unde nu existau.

Implementarea este costisitoare;

(5) OPT (121). Acest algoritm de înlocuire se poate dovedi optim în aceea că produce numărul minim de referiri la pagini, care nu se găsesc în memoria M_1. Cu scopul de a aplica acest algoritm, trebuie cunoscute mostrele exacte de referire ale programelor ce alcătuiesc sarcina sistemului.

Intr-o trecere a şirului de referiri de pagină, algoritmul înregistrează dinamic acele pagini care vor fi referite din nou mai tîrziu. In acest mod se poate obţine o ordine determinată aprioric de înlocuire a paginilor.

Algoritmul nu este practic, căci presupune cunoaşterea în avans a mostrelor de referire, însă este foarte util ca termen de comparaţie în analiza altor algoritmi de înlocuire.

./.

Algoritmii de cerere sînt bazaţi pe necesităţile dinamice de memorie ale programelor, ce alcătuiesc sarcina sistemului.

Două categorii de algoritmi justifică cercetarea:

(1) strategia de grup de lucru (52,57).

Grupul de lucru al unui program (vezi cap.4 secţiunea 1.2 pentru o descriere mai completă) este definit ca numărul distinct de pagini referite de către program în decursul unui anumit interval de timp; acest interval de timp este cunoscut ca reprezentînd parametrul grupului de lucru.

Intr-un mediu multiprogramat numai acele programe sînt eligibile de a putea fi rulate, a căror grup de lucru nu depăşeşte spaţiul propriu disponibil în memoria M_1; acest lucru garantează că suma grupurilor de lucru ale tuturor programelor active nu depăşeşte dimensiunea M_1. Odată ce programul devine activ, el se poate dezvolta după necesităţi, în concordanţă cu fluctuaţiile în dimensiunea grupului lui de lucru. In caz de necesitate, el poate înlocui pagini din alte programe, conform unui alt algoritm.

Această strategie este atît un algoritm de planificare, cît şi unul de înlocuire.

Fiecare pagină aparţinînd grupului de lucru la un moment dat, a unui program oarecare în rulare, trebuie să fie menţinută în memoria M_1. Paginile ce nu fac parte din grupul de lucru sînt obiect de înlocuire la apariţia necesităţii de spaţiu.

Prin variaţia parametrului grupului de lucru astfel ca să conţină mereu un număr fix de pagini, strategia de grup de lucru poate fi utilizată pentru a simula algoritmul LRU (56).

Ca o deficienţă intrinsecă asociată acestui algoritm, remarcăm dificultatea de a măsura dinamic parametrul de grup de lucru, cu scopul de a estima dimensiunea grupului de lucru;

(2) algoritm de tip Round-Robin modificat. In cadrul acestui algoritm fiecare program este favorizat în cadrul unui anumit interval de timp. In timpul intervalului lui de favorizare, un program se dezvoltă înlocuind pagini ale altor programe. Pagina aleasă pentru înlocuire este selectată pe baza altui algoritm.

./.

O deficienţă a acestui algoritm constă în dificultatea de a selecta un interval de timp corespunzător.

Aprecieri generale.

Una din lucrările clasice privind analiza algoritmilor de înlocuire este cea a lui Belady (18). Iată unele din concluziile lucrării:

- FIFO nu este semnificativ mai bun decît RAND; în cîteva cazuri, chiar mai slab;

- LRU şi FINUFO sînt considerabil mai buni decît algoritmii statici. Precizia cu care este înregistrată istoria trecută a procedurii de paginare nu pare să fie deosebit de importantă.

In ciuda complexităţii, algoritmul de detectare a ciclurilor ATLAS a fost apreciat ca producînd rezultate slabe; în unele cazuri performanţa obţinută cu acest algoritm a fost chiar mai rea decît utilizînd FIFO sau RAND (104). Motivul pentru această comportare neaşteptată, pare să fie în variaţia largă a dimensiunii ciclurilor programelor.

Denning (56) compară FIFO, LRU, FINUFO, RANDOM şi strategia de grup de lucru propusă de el, rezultatele lui fiind în concordanţă cu Belady (18).

Apare desigur întrebarea: "Cît de importantă este funcţia algoritmului de înlocuire în performanţa globală a unei ILVM?" Multe cercetări independente (84,89,123) indică că efectul algoritmului de înlocuire este de o importanţă secundară.

Din rezultatele acestor lucrări menţionate poate fi desprinsă următoarea concluzie. Dacă nu sînt admise costuri mari pentru implementarea algoritmului de înlocuire, atunci pot fi utilizaţi algoritmii FIFO sau RAND; dacă însă pot fi admise aceste costuri, atunci nu merită să se utilizeze un algoritm mai complex decît LRU sau FINUFO.

4. Indicatori de performanţă.

Intr-un cadru general, următorii indicatori de performanţă ai sistemului pot fi consideraţi semnificativi:

- utilizarea UCP: procentajul timpului în codul căruia UCP este ocupată cu executarea unui lucru "util";

./.

- capacitatea de tratare a sistemului (throughput) măsoară numărul mediu de lucrări executate în cadrul unei unităţi de timp;

- timpul de răspuns al sistemului: măsoară timpul de calcul mediu pentru terminarea unei lucrări.

Din păcate, însă, este foarte dificil a face o legătură directă între acesti indicatori de performanţă generali ai sistemului şi performanţa sistemului de memorii.

In această situaţie vom considera numai indicatorii ce au legătură directă cu performanţa efectivă a ierarhiei de memorii.

4.1. Proporţia succeselor. Proporţia eşecurilor.

Datorită structurii strict ierarhice a sistemului de memorii şi a modului de lucru al algoritmilor de înlocuire a paginilor, putem analiza performanţa sistemului prin considerarea separată a nivelurilor ierarhice, pornind de la M_1. Deoarece un anumit nivel al ILVM primeşte o cerere de aducere a unei pagini numai în situaţia în care informaţia dorită nu a fost găsită într-un nivel mai înalt al ierarhiei, fiecare nivel observă un şir diferit de referiri de pagină.

Fie $X(i) = x_1(i), x_2(i), \ldots, x\lambda_i(i)$
şirul de referiri de pagină a unui program observat de M_i.

Notăm:

λ_i = lungimea şirului de referiri de pagină;

σ_i = grupul de pagini distincte în şir;

η_i = numărul de pagini în σ_i

De exemplu, pentru următorul şir de referiri de pagină:

X(1): A, B, C, B, C, A, C

unde, prin litere mari de tipar s-au reprezentat adrese logice de pagină, se obţin următoarele valori:

$\lambda_1 = 7$

σ_1 = A, B, C

$\eta_1 = 3$

Pentru un anumit şir de referiri de pagină X(i), un nivel M_i al sistemului de memorii şi un algoritm de înlocuire, definim şirul de referiri de aducere a paginilor $X_a(i)$ secvenţa în timp a referirilor de pagină din X(i) care nu au fost găsite în M_i.

Fie λ_{ai} lungimea şirului $X_a(i)$.

Referirile de pagină, care au fost găsite în M_i le vom

./.

numi succese. Numărul succeselor este numărul de referiri satisfăcute de M_i şi este:

$$\text{nr. succese} = \lambda_i - \lambda_{ai}$$

Numărul de referiri de pagină ce nu au fost găsite în M_i este numit număr de eşecuri:

$$\text{nr. eşecuri} = \lambda_{ai}$$

In general, într-o ierarhie de memorii se urmăreşte maximizarea numărului de succese, sau, echivalent, minimizarea numărului de eşecuri. Este mai comod însă a considera raportul între numărul de succese (sau eşecuri) şi numărul total al referirilor - se obţin astfel proporţia succeselor şi proporţia eşecurilor:

$$S_i = \frac{\lambda_i - \lambda_{ai}}{\lambda_i} = 1 - e_i$$

$$e_i = \frac{\lambda_{ai}}{\lambda_i}$$

Fie X(1) un şir de referiri de pagină generat de UCP şi aplicat nivelului M_1 ca intrare (ierarhia este întotdeauna accesată la nivelul superior) şi fie Λ lungimea acestui şir. ($\Lambda \equiv \lambda_1$).

Proporţia eşecurilor de sistem a unui anumit nivel este definită ca:

$$e_{si} = \frac{\lambda_{ai}}{\Lambda}$$

iar proporţia succeselor de sistem, relativă la nivelul i, este:

$$s_{si} = 1 - e_{si}$$

Aplicînd determinările de mai sus la şirul de referiri de pagină X(1) generat de procesor şi primit de M_1, constatăm că şirul de referiri de aducere a paginilor $X_a(1)$, generat de M_1, este de fapt şirul de referiri de pagină X(2) primit de M_2.

Prin repetarea succesivă a acestui proces, putem crea şiruri de referiri de aducere a paginilor şi proporţiile respective de succese şi eşecuri pentru fiecare nivel al ierarhiei.

Presupunînd că toate informaţiile referite de program există în ierarhia de memorii, rezultă că suma proporţiilor suc-

./.

ceselor de sistem este egală cu unitatea:

$$\sum_{i=1}^{\ell} s_{si} = 1 \quad .$$

Există multe utilizări şi interpretări posibile ale proporţiei succeselor s_i. O trăsătură comună a tuturor acestor aplicaţii este faptul că s_i asigură o legătură comodă între un model abstract al ILVM şi sarcina de lucru reală a unui sistem de calcul (77).

Pentru un şir de referiri de pagină X(i) dat şi un algoritm dat de înlocuire a paginilor, proporţia succeselor s_i depinde numai de capacitatea de memorizare a nivelului M_i. Această funcţie este numită uneori funcţia de succese $s(c_i)$. Ea reprezintă probabilitatea găsirii informaţiei cerute pentru fiecare referire de memorie, la o memorie de capacitatea c.

Funcţia de eşecuri este definită corespunzător:

$$e(c) = 1 - s(c)$$

şi reprezintă probabilitatea de a nu găsi informaţia cerută în cadrul unei referiri de memorie într-o memorie de capacitatea c.

O mostră de funcţie e(c) este dată în figura 6.

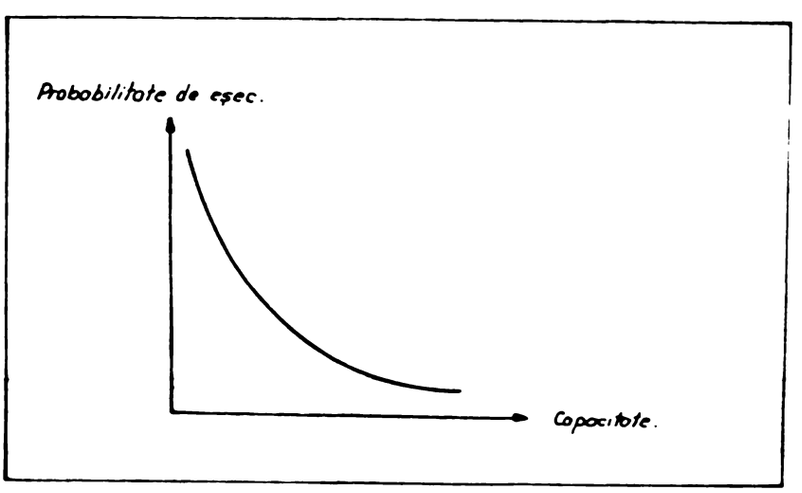


Fig.6. O reprezentare calitativă a unei dependenţe e(c).

Cunoaşterea funcţiilor s(c) şi e(c) permite tratarea unor probleme de optimizare ce apar în cadrul proiectării unui sistem de ierarhii de memorii, în legătură cu combinarea diferitelor tehnologii, cc au caracteristici diferite de cost-performanţă (a se vedea secţiunea 5.2.1.).

4.2. Timp de acces efectiv. Cost efectiv.

Indicatorii generali ai performanţei unui ILVM sînt timpul

./.

de acces efectiv t_{ef} şi costul efectiv b_{ef}, definite astfel:

$$t_{ef} = t_1 s_{s1} + t_2 s_{s2} + \ldots + t_l s_{sl} = \sum_{i=1}^{l} t_i s_{si}$$

$$b_{ef} = \frac{b_1 c_1 + b_2 c_2 + \ldots + b_l c_l}{c_1 + c_2 + \ldots + c_l} = \frac{\sum_{i=1}^{l} b_i c_i}{\sum_{i=1}^{l} c_i}$$

t_{ef} şi b_{ef} pot fi considerate caracterizînd întreaga ierarhie de memorii, echivalînd-o cu un sistem cu un singur nivel.

Din punctul de vedere al indicatorilor de cost-performanţă pentru un proiectant trebuie să fie indiferentă alegerea între un dispozitiv de memorie cu o singură tehnologie şi un singur nivel, cu timpul de acces mediu t_{ef} şi costul/bit b_{ef} şi un sistem ierarhic de memorii cu parametrii de performanţă (t_{ef}, b_{ef}).

In cazul că proiectantul sistemului necesită o memorie cu indicatorii (t,b) şi o astfel de tehnologie de bază nu există, el va încerca să construiască o ierarhie de memorii, astfel încît:

$$(t_{ef}, b_{ef}) = (t, b)$$

5. Evaluarea şi optimizarea ILVM.

In timpul proiectării unei anumite configuraţii de sistem, proiectantul de calculatoare are două alternative principale: una de a simula sistemul şi alta de a dezvolta un model matematic ce încorporează parametrii principali de interes.

In continuare vom arăta modalităţile de aplicare a acestor tehnici în problemele de evaluare şi optimizare a ILVM, comparînd utilitatea şi avantajele lor.

5.1. Simulare.

In mod empiric, cu cît mai puţin este cunoscută comportarea sistemului, cu atît mai probabilă este necesitatea simulării. De asemenea, cu cît mai complicat este un sistem, cu atît mai microscopică, mai detaliată trebuie să fie simularea.

O abordare tipică în analiza ILVM a fost simularea unei varietăţi de scheme, la diferite nivele de detaliere, cu scopul

./.

de a limita cîmpul proiectelor -soluţii posibile. Aceste simulări sînt rulate cu unele şiruri de referiri reale, ce sînt considerate a fi reprezentative pentru sarcina pe care sistemul o va trata, în eventualitatea că va fi implementat.

Sînt multe motive ce fac studiile de simulare a ILVM foarte lungi şi detaliate:

a) este foarte dificil a caracteriza proprietăţile statistice a programelor ce acoperă un mare domeniu de aplicaţii.

Prin urmare este necesar ca determinarea performanţelor unei anumite configuraţii de memorii să se facă pe baza şirurilor de referiri reale, obţinute fie sintetic, fie din programe reale.

Trebuie subliniat că obţinerea şirurilor de referiri din programele reale nu este deloc un proces ieftin. Nu numai că trebuie să fie disponibile echipamente hardware şi software corespunzătoare pentru a înregistra succesiunea referirilor, dar de asemenea volumul de informaţii de memorat este foarte mare; un program FORTRAN tipic poate genera 40.000 referiri. După selectarea unei anumite dimensiuni de pagină, acest şir de referiri trebuie adus la forma unui şir de referiri de pagină;

b) procedura de aducere şi înlocuire a paginilor nu se pretează a fi tratată în formă analitică.

Procesul de transfer al paginilor în cadrul ierarhiei de memorii este dependent de proprietăţile statistice ale încărcării sistemului şi nu este posibil a reprezenta acest proces în formă analitică. Chiar dacă am asocia şirului de referiri de pagină un anumit tip de comportament probabilistic, activitatea de transpunere în termeni matematici a acestui comportament nu este o sarcină uşoară;

c) algoritmul de înlocuire este probabil principalul factor ce face modelul analitic foarte dificil de formulat.

Chiar dacă sînt făcute presupuneri foarte stricte asupra activităţii de paginare, este greu a formula un model ce ţine seama într-un mod cantitativ de algoritmul de înlocuire.

A simula o ILVM cere, în mod obişnuit, tratarea fiecărei adrese singulare, cu actualizarea corespunzătoare a tabelului

./.

adresei paginilor, a informaţiilor ce stau la baza activităţii algoritmului de înlocuire, a şirurilor pentru tratare UCP şi I/E etc.

5.1.1. Tehnici de eşantionare.

Uneori pot fi propuse modele analitice, dar nu este posibil a obţine soluţii generale pentru model într-o formă compactă. Cazul apare frecvent cînd distribuţiile probabilistice ale unuia sau mai multor parametri sînt complexe sau nu pot fi exprimate în termeni matematici. In mod obişnuit într-o asemenea situaţie se utilizează metoda convenţională Monte Carlo sau tehnici de eşantionare.

Fie X o variabilă aleatoare cu o funcţie de densitate a probabilităţii arbitrară f(x). Presupunem că se doreşte a genera valori ale lui x, astfel ca aceste valori să urmeze distribuţia prescrisă.

Fie F(x) distribuţia de probabilitate a variabilei aleatoare x:

$$F(x) = \int_{-\infty}^{x} f(x)\, dx$$

Atunci, o valoare de eşantion a lui x, x_o, poate fi obţinută astfel:

$$x_o = F^{-1}(R) \qquad (a)$$

unde R este o variabilă aleatoare uniform distribuită în $[0,1]$.

Dacă funcţia inversă pentru distribuţia probabilităţii variabilei aleatoare arbitrare nu există, este întotdeauna posibil să se genereze eşantioane a acelei variabile aleatoare, prin înlocuirea relaţiei (a) cu o expresie aproximativă.

De asemenea, am presupus că există disponibil un generator aleator cu distribuţie uniformă.

Alte metode, în afara acestei metode directe, de generare a numerelor aleatoare urmînd o lege de probabilitate specificată, sînt:

- metoda respingerii, datorată lui von Neumann, 1951;
- metoda compunerii, propusă de Butler, 1956;
- metoda compunerii-respingerii, propusă de Butcher, 1961.

Descrierea acestor metode se poate găsi în tratate de

./.

statistică.

Fie un şir de valori ale variabilei aleatoare x, generate conform uneia din metodele de mai sus.

$(x_1, x_2, \ldots\ldots, x_n)$

Legea numerelor mari arată că:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} x_i = E(x)$$

Prin urmare, cu cît se doreşte o exactitate mai mare, cu atît mai multe valori de eşantion ar trebui generate. Acesta este un motiv pentru care, chiar dacă modelul probabilistic pentru un anumit sistem poate fi propus, simularea este în general o metodă costisitoare.

Au fost propuse cîteva tehnici pentru a grăbi convergenţa mediei unui grup de valori de eşantion la valoarea aşteptată: unele din ele sînt explicate în detaliu de Shedler şi Yang (150) în lucrarea lor ce se ocupă de unele aspecte ale simulării unui sistem de paginare.

5.1.2. Tehnici de tratare pe principiul stivei.

O contribuţie importantă în domeniul simulării unei ILVM a fost adusă de Mattson ş.a. (121). Metoda, denumită tratare pe principiul stivei, poate fi utilizată pentru determinarea timpului mediu de acces a unei ILVM generale, presupunînd că organizarea ierarhiei este complet asociativă sau grup asociativă (vezi secţiunea 6.2. a acestui capitol) şi algoritmul de înlocuire utilizat este din categoria algoritmilor de stivă. Un algoritm de înlocuire este denumit de stivă dacă el produce o ordonare totală a tuturor paginilor referite anterior şi utilizează această ordonare pentru luarea deciziilor de înlocuire. Ordonarea poate fi reprezentată ca o listă de priorităţi ce urmăreşte o relaţie de ordonare; pagina cu prioritatea cea mai mică este selectată pentru înlocuire. Exemple tipice de algoritmi de stivă sînt LRU şi LFU în timp ce FIFO nu intră în această categorie.

Lucrarea consideră o organizare complet asociativă cu înlocuire LRU.

./.

Fie $X = \{x_1, x_2, \ldots x_t\}$ un şir de referiri de pagină ce serveşte ca intrare pentru ierarhie.

Γ_t reprezintă grupul paginilor distincte în X, iar

η_t reprezintă numărul paginilor în Γ_t.

Fie C o limită superioară a capacităţii memoriei M_1 (în pagini); $B_t(C)$ reprezintă grupul de pagini în M_1 imediat după ce pagina x_t a apărut în şirul de referiri.

Faptul că LRU este un algoritm de stivă, implică:

$$B_t(1) \subset B_t(2) \subset \qquad \subset B_t(\eta_t)$$

şi

$$B_t(C) = \begin{cases} C & \text{pentru } 1 \leq C < \eta_t \\ \eta_t & \text{pentru } C \geq \eta_t \end{cases}$$

Pe baza proprietăţii de incluziune precedente, grupul de pagini Γ_t poate fi ordonat într-o listă:

$$S_t = \{s_t(1) \quad s_t(2) \quad s_t(\eta_t)\} \qquad \text{unde:}$$

$$s_t(i) = B_t(i) - B_t(i-1)$$

pentru $i = 1,2,\ldots,\eta_t$

Grupul de pagini S_t este denumit stivă LRU şi reprezintă conţinutul lui M_1 în orice moment t.

Fie C_t capacitatea critică, astfel ca:

$$x_t \in B_{t-1}(C)$$

dacă şi numai dacă $C \geq C_t$.

Capacitatea critică pentru o pagină care nu a mai fost referită anterior este considerată arbitrar ca infinită.

Datorită modului în care este construit S_t, C_t este pur şi simplu poziţia paginii x_t în stiva LRU; C_t este de asemenea numit distanţă în stivă deoarece reprezintă poziţia oricărei pagini referite anterior în stiva LRU.

Fie $n(\Delta)$ numărul de ori, de care o anumită distanţă în stivă Δ apare la tratarea şirului de referiri X; atunci numărul de ori de care o pagină este găsită în M_1 se poate determina simplu:

$$N(C) = \sum_{\Delta=1}^{C} n(\Delta)$$

./.

De aici, proporţia eşecurilor poate fi exprimată prin:

$$e(C) = 1 - N(C) / \Lambda$$

unde Λ este numărul total al paginilor în şirul de referiri.

Procedura de actualizare pentru stivă este simplă:

$s_t(1) = x_t$

$s_t(i) = s_{t-1}(i-1)$ pentru $2 \leq i \leq C_t$ şi $C_t \neq \infty$

$s_t(i) = s_{t-1}(i-1)$ pentru $2 \leq i \leq \eta_t$ şi $C_t = \infty$

Contribuţia principală a acestei metode constă în simplitatea ei şi posibilitatea obţinerii proporţiei eşecurilor sau complementului ei într-o singură trecere a şirului de adrese. De semnalat este faptul că această valoare se poate obţine simultan nu numai pentru capacitatea maximă C a lui M_1, dar şi pentru orice capacitate mai mică decît ea.

Această metodă, cu mici modificări, se aplică la o ILVM cu orice număr de nivele, orice algoritm de stivă şi de asemenea organizări grup asociative.

Deşi par foarte atractive, tehnicile de stivă implică un volum foarte mare de calcule, deoarece stiva trebuie să fie actualizată la fiecare nouă referire în şirul de referiri de pagină. Dacă capacitatea limită superioară C este mare, după cum ne putem aştepta în mod obişnuit, această tehnică devine foarte scumpă.

In concluzie, simularea este necesitată ori de cîte ori nu sînt disponibile tehnici analitice. Ea asigură rezultate a căror exactitate depinde de gradul de detaliere cu care este realizată; de obicei este costisitoare, atît în efortul de programare, cît şi în timpul de calcul.

5.2. Modelare matematică.

5.2.1. Probleme de optimizare de programare neliniară.

Se consideră un sistem ILVM cu l nivele, fiecare nivel i fiind caracterizat de timpul de acces t_i, costul/bit b_i şi capacitatea c_i.

Criteriul de optimizare ales în (35) este de a minimiza timpul de acces efectiv al ierarhiei t_{ef} şi a satisface restricţiile privind costul total al sistemului şi capacităţile, sau, echivalent, de a minimiza costul total al sistemului şi a satis-

./.

face restricţiile privind timpul de acces efectiv al ierarhiei t_{ef} şi capacităţile.

Problema este următoarea:

Sînt date:

- capacitatea de memorizare c_l;
- funcţia de cost de tehnologie b(t), considerată ca funcţie de timpul de acces;
- costul total al sistemului b_s

$$b_s = \sum_{i=1}^{l} b(t_i)\ c_i$$

- funcţia de succese s(c) sau funcţia de eşecuri e(c), considerate dependente de capacitate (a se vedea secţiunea 4.1);
- numărul de nivele l.

Variabilele:

$$t_1, t_2, \ldots\ldots, t_l$$

$$c_1, c_2, \ldots\ldots, c_{l-1}$$

Să se minimizeze funcţia:

$$t_{ef} = \sum_{i=1}^{l} e(c_{i-1})\ t_i$$

şi să se satisfacă restricţiile:

$$b = \sum_{i=1}^{l} b(t_i).\ c_i \leqslant b_s$$

$t_i > 0$ pentru $i = 1,2,\ldots,l$ şi

$c_i > 0$ pentru $i = 1,2,\ldots,l-1$

Deoarece funcţiile s(c)(corespunzător e(c)) şi b(t) în cazul general nu sînt lineare, minimizarea se prezintă sub forma unei probleme de programare neliniară.

In (32) se presupune că funcţia de eşecuri e(c) şi cea de cost de tehnologie b(t) sînt funcţii-puteri, şi anume:

$e(c) = c^{-\alpha}$ (m)

$b(t) = t^{-\beta}$ şi (n)

./.

Aceste presupuneri par restrictive, însă la nivelul actual al analizei şi datorită insuficienţei unor date, ele ne permit o înţelegere mai bună a comportării ILVM şi posibilitatea de a determina influenţa anumitor parametri asupra acestei comportări. De altfel, aproximaţia (n) este utilizată şi în lucrarea (114) cu parametrul $0,2 \leq \beta \leq 96$, iar funcţia putere (m) nu este mult diferită de proporţia experimentală a eşecurilor, dată în (120).

5.2.2. Utilizarea modelelor cu şiruri de aşteptare.

In cadrul unui sistem de calcul cu multiprogramare, mai multe lucrări concurează pentru a utiliza resursele comune ale sistemului. Presupunînd că numărul de cereri este mai mare decît numărul unităţilor de resurse comune, în mod necesar se formează în sistem şiruri. Asemenea şiruri de aşteptare se formează şi pentru procedura de utilizare a ierarhiei de memorii. Analizînd comportarea acestor şiruri, pot fi obţinute multe rezultate privind comportarea sistemului. In mod raţional, teoria şirurilor de aşteptare a devenit unul din mijloacele importante ale analizei ILVM.

Am arătat mai înainte că caracterizarea statistică a unor parametri este o problemă foarte dificilă; de multe ori neavînd nici o alternativă, se recurge la simulare. Foarte des însă, precizia cu care sînt obţinute anumite rezultate nu reprezintă obiectivul principal, ci mai curînd se urmăreşte o înţelegere calitativă a modului cum sistemul este afectat de variaţiile diferiţilor parametri. Pentru aceste tipuri de studii, modelele ce utilizează şiruri de aşteptare sînt foarte utile.

Utilizarea modelelor cu şiruri de aşteptare ca mijloc de analiză a avut şi insuficienţe, precum:

(a) au fost analizate numai configuraţii restrînse şi mai curînd specializate. Pentru proiectant a fost necesar să încadreze sistemul lui real în aceste modele şi să facă în acest scop restricţii foarte severe. Fiind dată pentru analiză o anumită configuraţie de sistem, procedura a fost de a examina diferite structuri de aşteptare pentru care au fost obţinute soluţii matematice, sperînd ca una din aceste structuri să fie cumva corespunzătoare;

(b) soluţia noilor structuri de aşteptare a fost foarte nesistematică; în principal, bazaţi pe rezultate anterioare, se

./.

testa o soluţie ce apărea ca probabilă, pentru a determina dacă ea satisface ecuaţiile de bază ale sistemului de aşteptare. Dacă o asemenea soluţie s-ar dovedi nepotrivită, printr-o selectare atentă repetată şi analiza erorilor, s-ar urmări obţinerea unei soluţii corecte.

Pînă la obţinerea unei soluţii automate şi bine înţelese, nu s-ar fi pus problema unei utilizări eficace a sistemelor cu aşteptare de către analistul de sistem.

O asemenea metodă a fost propusă şi anume Irani şi Wallace (101) au descris o metodă, prin care pot fi determinate automat soluţii numerice pentru reţele arbitrare de aşteptare cu timp de servire exponenţial. (Notăm că într-un sistem de calcul toate distribuţiile timpului de servire sînt presupuse exponenţiale). Acesta a fost un rezultat important, deoarece a făcut posibilă analiza într-un mod interactiv a diferitelor configuraţii de sistem. Limitarea acestei metode este aceea că pentru un număr mare de utilizatori solicitînd servirea sau o reţea mare de aşteptare, trebuie să fie inversate matrici foarte mari, implicînd un volum important de calcule.

Un pas important în automatizarea soluţiei pentru reţelele cu aşteptare sînt lucrările lui Chandy (32,33). El dă o metodă prin care pot fi determinate într-o manieră surprinzător de simplă, soluţii algebrice a unor reţele de aşteptare arbitrare cu timpi de servire exponenţial.

O metodă pentru studiul ILVM utilizînd acest aparat al sistemelor de aşteptare este dat în (136). Metoda este rapidă şi flexibilă la schimbarea parametrilor sistemului. Astfel, se cercetează efectul schimbărilor în parametrii sistemului (nivelul de multiprogramare, numărul nivelelor de memorii şi capacitatea lor, numărul canalelor, balanţa activităţilor UCP şi de I/E) asupra caracteristicilor sistemului (utilizarea diferitelor staţii de servire, lungimea şirurilor de aşteptare).

6. Realizări şi cercetări în domeniul ILVM.

Scopul cercetărilor în domeniul ILVM este de a dezvolta ierarhii de memorii cu caracteristici cost-performanţă

./.

atractive. O cale simplă de a scădea considerabil parametrul de cost/bit este de a dimensiona nivelele ierarhiei într-un mod crescător pe măsură ce trecem de la nivele cu performanţe ridicate şi cost ridicat la nivelele cu performanţe scăzute şi cost scăzut (adică $b_1 > b_2 > b_3$, ..., şi $c_1 < c_2 < c_3$ ...,).

Presupunînd că referirile de adrese generate de UCP sînt distribuite uniform în timp şi în domeniul de adrese, fiecare adresă ar avea o probabilitate egală de a fi referită în orice moment. Această probabilitate ar fi:

$$Pr\ (ref) = \frac{1}{c_1 + c_2 + \ldots + c_1}$$

Astfel, proporţia aşteptată de succese a sistemului s^o_{si} pentru fiecare nivel i, este proporţională cu dimensiunea nivelului.

De exemplu:

$$s^o_{s1} = c_1/(c_1 + c_2 + \ldots + c_1)$$

Şi, deoarece am presupus că $c_1 < c_2 < \ldots < c_1$, rezultă că:

$$s^o_{s1} < s^o_{s2} < s^o_{s3} < \ldots < s^o_{s1}$$

Astfel, funcţia aşteptată de succese pentru nivelul 1 domină (este aproximativ egală cu 1), deoarece am presupus că acesta este nivelul cu capacitatea cea mai mare. Referindu-ne la definiţia timpului de acces efectiv al ierarhiei t_{ef}, putem stabili că el va fi aproximativ egal cu timpul de acces al nivelului cu performanţa cea mai coborîtă (1) deoarece toţi ceilalţi termeni în relaţia t_{ef} (secţiunea 4.2) ar fi neglijabili. Dacă această analiză ar fi adevărată, ierarhia noastră de memorii ar avea, în final, o performanţă cu puţin mai bună decît nivelul de performanţă cel mai coborît la o creştere modestă în preţ - un rezultat nu deosebit de îmbucurător.

In realitate însă, ierarhiile de memorii nu se comportă în acest mod. A fost observat în mod empiric că programele reale îşi distribuie referirile lor la memorie astfel încît în decursul unui interval de timp este folosit numai un subset din informaţiile disponibile. In cap.5 vom prezenta o analiză detaliată a acestui comportament.

./.

După cum s-a arătat mai înainte, fiecare nivel al ierarhiei "observă" o altă trasă de adrese a programului. Nivelele înalte ale ierarhiei urmăresc mostrele de referire "microscopică" instrucţie cu instrucţie, pe cînd nivelele medii urmăresc mostre de referire mai mari, subrutină cu subrutină. Nivelele joase ale ierarhiei urmăresc referirile UCP, atunci cînd aceasta trece de la un subsistem la alt subsistem.

Determinarea valorilor s_i a proporţiei succeselor pentru diferitele nivele M_i, în cadrul însă mai restrîns al unei clase de ILVM, numită ierarhie cu faze, este realizată în (77). Metoda folosită se bazează pe o generalizare a tratării pe principiul stivei (descrisă la 5.1.2.) numită tratare pe principiul stivei adaptată.

Vom prezenta acum, pe scurt, realizări şi cercetări în domeniul ILVM, marea majoritate referindu-se la ierarhii cu 2 şi 3 nivele, deşi, de exemplu, într-o carte a lui Lorin (116) este prezentată o ierarhie cu 7 nivele.

6.1. Sisteme cu memorie virtuală.

Sistemele timpurii automate de memorii au fost bazate pe ierarhii cu două nivele de memorie cu miezuri - tambur (dispozitive similare celor din poziţiile 3 şi 6 ale tabelului 1). Această tehnică a fost introdusă în sistemul Atlas (73,105) în anii 1960. De atunci a fost şi este utilizată în multe sisteme contemporane. Astfel, memorii virtuale sînt oferite de sistemele IBM-370, UNIVAC-seria 70, DEC-10, Burroughs - seriile 500 şi 700, unele sisteme CDC, Xerox-seria Sigma, Fujitsu-seria Facom, Amdahl 470 V/6 ş.a.

Performanţa sistemelor cu memorie virtuală a fost studiată de numere şi cercetători: Belady (18), Coffman şi Varian (40,159), Hatfield (88), Sayre (145). Intre rezultatele lui Coffman, de exemplu, a fost notat că deşi raportul $c_1/(c_1+c_2) = 0,25$, proporţia succeselor s_1 depăşeşte 95%. Hatfield a studiat performanţa programelor de sistem, care au fost proiectate cu atenţie şi a determinat că pentru proporţii ale $c_1/(c_1+c_2)$ de 0,25 a fost posibil ca proporţia succeselor să depăşească 99%.

Există două abordări de bază pentru a obţine efectul de memorie virtuală, segmentare şi paginare.

./.

IBM, CDC, Xerox şi Univac au ales paginarea, Burroughs a implementat segmentarea.

Vom prezenta sumar aceste abordări.

6.1.1. Memorie virtuală paginată.

Paginarea împarte memoria virtuală în blocuri (pagini) multiple de dimensiune fixă şi egală, care sînt apoi aduse în memoria principală pe măsura necesităţilor.

Memoria paginată are un mare avantaj: programatorii nu mai trebuie să fie preocupaţi de dimensiunea programelor lor.

Partiţiile în memoria virtuală pot fi suficient de mari pentru toate scopurile practice.

De exemplu, sistemul de operare utilizat în cadrul IBM System/370 - Operating System - Virtual Storage 2 (OS/VS2) are la dispoziţie o capacitate de memorie virtuală de 16 milioane octeţi. Intr-o asemenea situaţie, întotdeauna vor exista partiţii disponibile în memoria virtuală, indiferent de capacitatea memoriei reale. Sistemul de operare va plasa în memoria reală, la momentul respectiv, numai paginile necesitate de desfăşurarea programului.

In aceste condiţii, programatorii de aplicaţii îşi pot concentra eforturile asupra aplicaţiei respective; programatorii ce se ocupă cu întreţinerea programelor nu trebuie să se mai îngrijească de posibilitatea încadrării unui program modificat într-o partiţie dată (ei măresc doar partiţia respectivă dacă este necesar), iar programatorii de sistem nu mai au de înfruntat probleme legate de limitări de capacitate a memoriei şi pot să se concentreze asupra administrării priorităţii lucrărilor.

Un alt avantaj important adus de acest tip de memorie virtuală constă în eliminarea fragmentării externe a memoriei (137). Problema fragmentării memoriei în cadrul unei ILVM este discutată în cap.3, secţiunea 1.1. Aici ne rezumăm a arăta că, în situaţia împărţirii memoriei în cadre de pagină, se creează posibilitatea folosirii integrale a spaţiului de memorie; nu vor mai exista domenii libere de memorie reală (ce apar şi se amplifică între partiţiile unui sistem de operare convenţional) ce nu pot fi folosite pentru a dispune în ele programe, deoarece sînt prea mici pentru acest scop.

./.

Vom analiza în continuare factorii ce afectează performanţa unui sistem cu memorie virtuală paginată şi apoi căile de îmbunătăţire a acestei performanţe.

6.1.1.1. Factori ce afectează performanţa unui sistem cu memorie virtuală paginată.

Factorii ce afectează performanţa unui sistem într-un mediu cu memorie virtuală paginată (pe lîngă cei generali - ce se referă la un mediu cu memorie non-virtuală) sînt:

(1) viteza şi numărul dispozitivelor de paginare.

Operaţia de transport a paginilor între memoria principală şi cea auxiliară implică consumul unui anumit timp; acest timp este o funcţie de caracteristicile tipului de dispozitive de memorii utilizate. Pentru un anumit tip dat de asemenea dispozitive poate fi calculată rata maximă de paginare, exprimată în numărul de defecte de pagină ce pot fi tratate într-un interval dat de timp.

Capacitatea maximă de paginare a unui sistem dat poate fi mărită prin utilizarea unor dispozitive de memorii auxiliare mai rapide sau prin utilizarea mai multor asemenea dispozitive.

(2) viteza UCP. O relaţie de neconcordanţă între viteza UCP şi viteza dispozitivelor de paginare poate conduce la situaţia în care UCP este obligată în mod frecvent să aştepte executarea unor operaţii I/E de paginare. Cînd cererile pentru operaţii de paginare sînt permise să apară mai frecvent decît rata de paginare pe care sistemul o poate asigura, astfel încît sistemul poate executa puţin sau chiar deloc lucrul util în afara celui legat de activitatea de paginare, sistemul este într-o situaţie numită thrashing.

Sistemele de operare lucrînd cu memorii virtuale supraveghează activitatea sistemului, pentru a determina momentele în care activitatea de paginare devine excesivă. Intr-un asemenea moment, este executată deactivarea unei lucrări. Ca urmare, cadrele de pagină asociate cu lucrarea deactivată devin disponibile şi pot fi alocate altor lucrări pentru a reduce activitatea de paginare. Mai tîrziu, la reducerea suficientă a activităţii de paginare, lucrarea deactivată este reactivată.

./.

Din cele de mai sus rezultă că UCP mai rapide impun utilizarea unor dispozitive de memorii auxiliare de paginare mai rapide pentru a putea asigura tratarea ratei mai mari a defectelor de pagină:

(3) capacitatea memoriei reale. Capacitatea memoriei reale prezentă în sistem influenţează numărul de defecte de pagină ce apar, în cursul tratării unui şir de lucrări. Dacă capacitatea memoriei reale a sistemului este egală cu capacitatea totală a memoriei virtuale utilizată de către lucrările în execuţie simultană, pentru aceste lucrări nu apare nici un defect de pagină. Cînd capacitatea memoriei reale este mai mică decît cea a memoriei virtuale utilizate, apar defecte de pagină.

Numărul total al defectelor de pagină care apar pentru un anumit şir de lucrări este influenţat de raportul dintre memoria virtuală utilizată şi memoria reală disponibilă. Presupunînd că capacitatea memoriei virtuale utilizată în cadrul sistemului rămîne constantă, raportul memoriilor virtuală/reală poate varia, ca urmare a modificărilor în capacitatea memoriei reale efectiv disponibile pentru paginare. Aceste modificări sînt datorate variaţiilor ce apar în timpul tratării şirului de lucrări, a numărului de pagini fixe din memoria reală (care nu sînt supuse - pentru un termen lung sau scurt, definit de sistemul de operare - activităţii curente de paginare). Capacitatea memoriei reale disponibile pentru paginare într-un moment oarecare este diferenţa între capacitatea memoriei reale a sistemului şi capacitatea paginilor definite ca fixe.

Pe măsură ce raportul memoriilor virtuală/reală creşte, de obicei creşte şi rata defectelor de pagină. Această creştere a ratei defectelor de pagină este lentă pe un interval, ca la depăşirea unei anumite valori a proporţiei memoriilor, creşterea să devină rapidă. Figura 7 ilustrează o dependenţă generală între numărul defectelor de pagină şi proporţia memoriilor virtuală/reală. La reducerea capacităţii memoriei reale disponibile vor apărea mai multe cereri către dispozitivele de paginare. Dacă capacitatea memoriei reale devine prea mică, creşterea ratei defectelor de pagină poate depăşi limita permisă. De aceea, pentru a obţine un anumit nivel de performanţă, în cadrul unei configuraţii

./.

(ce administrează o anumită capacitate de memorii virtuale şi tratează un şir dat de lucrări), devine necesară utilizarea unei capacităţi mai mari de memorie reală în cazul utilizării unor dispozitive de paginare relativ lente, comparativ cu situaţia în cadrul utilizării unor dispozitive de paginare rapide;

(4) structura programului. Modul şi frecvenţa de referire a paginilor în cadrul unui program au un efect considerabil asupra numărului defectelor de pagină ce apar la tratarea acestui program. Astfel, în cadrul unui program ce face în mod constant referiri la un mare număr de pagini diferite de instrucţii şi date pentru durate foarte scurte de timp, într-o manieră aleatoare, poate apărea o rată excesivă de paginare.

Majoritatea tipurilor de programe manifestă o localizare naturală în caracteristica referirilor, astfel încît ele pot fi structurate pentru a opera ca o serie de faze logice. In cazul cel mai simplu, de exemplu, un program poate consta din punct de vedere logic dintr-o fază de iniţializare, o fază principală, una sau mai multe faze de tratare a excepţiilor şi o fază de terminare.

Comportarea de acces a programelor va fi discutată în detaliu în capitolul 4, unde vom construi un model corespunzător al acestei comportări.

O pagină este definită ca activă, dacă în timpul executării unei faze logice, are o probabilitate mare de a fi referită de mai multe ori; o pagină pasivă este definită ca avînd o probabilitate mică de a fi referită mai mult decît o dată în cursul executării fazei.

O fază logică este însoţită de o activitate minimă de paginare la execuţia ei cînd paginile ei active rămîn în memoria reală în timpul execuţiei lor, iar paginile pasive sînt aduse în memoria reală, la cerere. Un program utilizează memoria reală în modul cel mai eficient, cînd instrucţiile şi datele active în cadrul fiecărei faze logice sînt conţinute în numărul cel mai mic posibil de pagini;

(5) capacitatea memoriei virtuale.

Odată cu creşterea capacităţii memoriei virtuale uti-

./.

lizată în sistem, creşte de asemenea şi numărul paginilor active şi pasive pe care sistemul trebuie să le trateze. La o creştere dată a memoriei virtuale, proporţia paginilor active şi pasive depinde de modul în care este utilizată memoria virtuală suplimentară. Atîta timp cît este disponibilă suficientă memorie reală pentru a conţine toate sau majoritatea numărului crescut de pagini active, creşterea în activitatea de paginare va fi necesitată în principal pentru paginile pasive şi este relativ mică. Imediat ce utilizarea unei memorii virtuale mai mari determină ca numărul paginilor active să depăşească în mod constant capacitatea memoriei reale, activitatea de paginare devine mai intensă. Crescînd continuu numărul paginilor de tratat, activitatea de paginare poate depăşi capacitatea de paginare maximă a sistemului, dacă nu are loc deactivarea unor lucrări (fig.7).

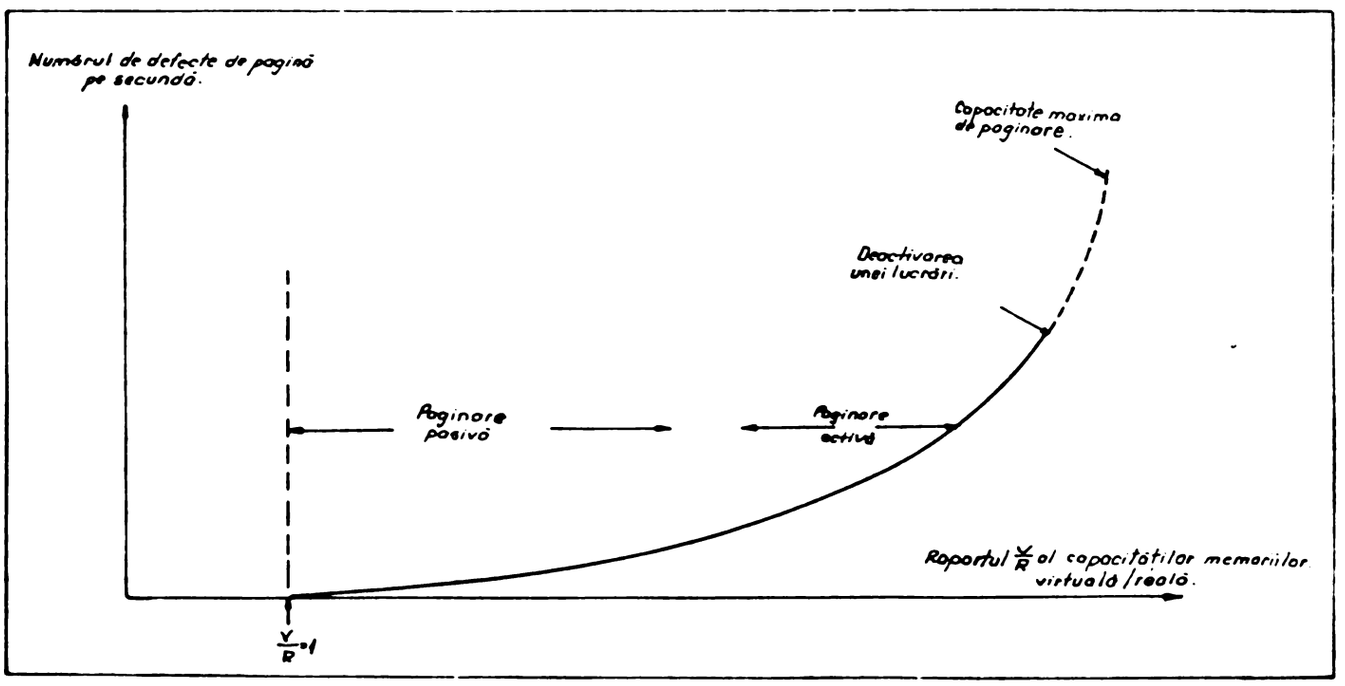


Fig.7. Efectul creşterii raportului capacităţilor memoriilor virtuală/reală asupra numărului defectelor de pagină.

In figura 7 prin activitate de paginare pasivă/activă s-a reprezentat activitatea de paginare, în cadrul căreia are loc în majoritatea timpului transportul paginilor pasive/active.

Pe măsura creşterii raportului capacităţilor memoriilor virtuală/reală creşte şi numărul defectelor de pagină, sistemul

./.

trecînd de la o activitate de paginare pasivă la o paginare activă şi se apropie de capacitatea maximă de paginare a sistemului.

Figura 8 ilustrează modul de variaţie al performanţei sistemului în domeniile de paginare activă şi pasivă, ţinînd seama de toţi factorii ce influenţează performanţa.

Utilizarea mărită a memoriei virtuale este avantajoasă pentru performanţa sistemului pînă la un anumit punct. După aceasta, pot fi utilizate capacităţi de memorie virtuală adiţionale pentru tratarea unor funcţii suplimentare la un cost variabil în performanţa sistemului.

Performanţa sistemului se menţine la un nivel ridicat în întreg domeniul de paginare pasivă. Aceasta deoarece în acest domeniu există o activitate de paginare relativ redusă şi de asemenea o probabilitate mare ca întregul sau majoritatea timpului consumat pentru tratarea paginării (timp UCP şi I/E) poate fi suprapoziţionat cu alte prelucrări.

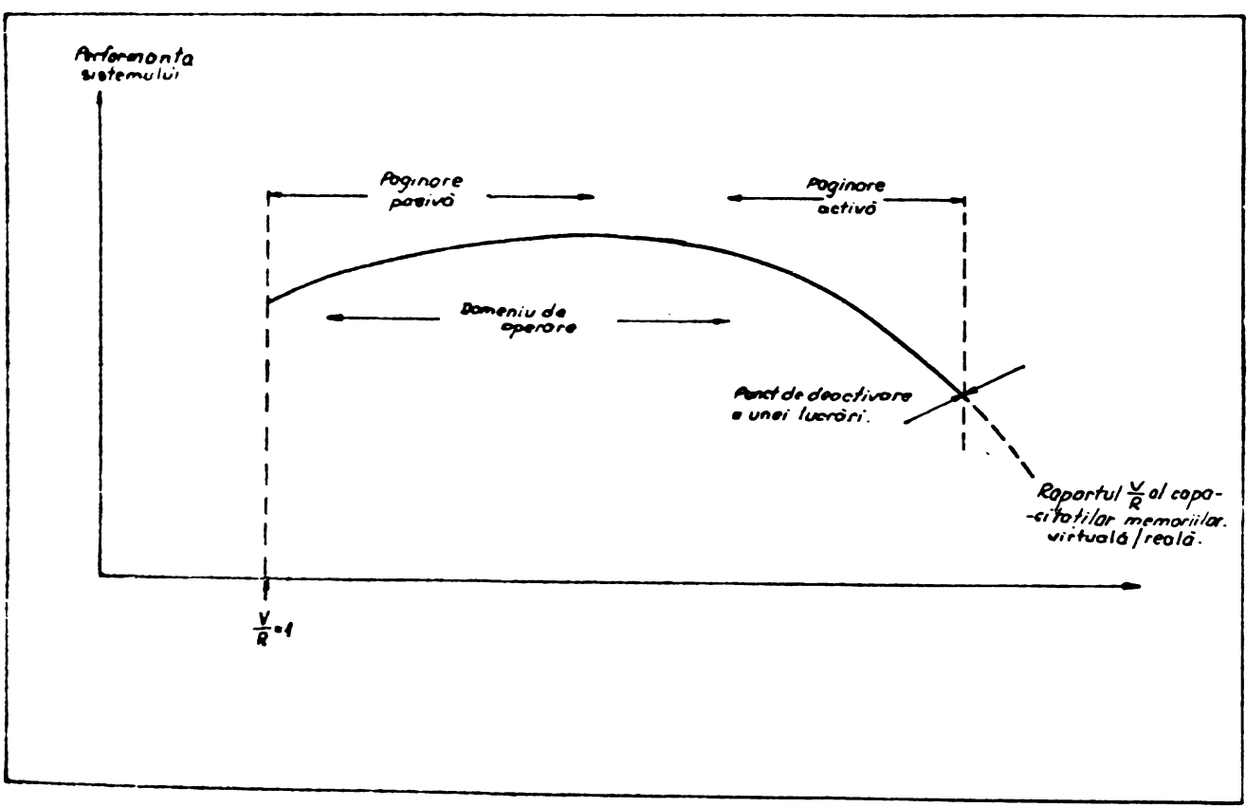


Fig.8. O curbă generală de performanţă a unui siste pentru un mediu cu memorie virtuală.

Pe măsură ce activitatea de paginare se intensifică, există o probabilitate mai mare ca tratarea UCP să fie oprită pentru a aştepta terminarea unei operaţii de paginare. In măsura în care UCP intră în starea de aşteptare mai frecvent pentru a aştepta paginarea şi o proporţie mai mică din această activitate de paginare este superpoziţionată cu alte proceduri de tratare, performanţa sistemului scade.

Intr-un sistem real, proporţia capacităţilor memoriilor virtuală/reală şi rata defectelor de pagină variază în timpul tratării de sistem, deoarece capacitatea memoriei virtuale utilizată (din capacitatea totală disponibilă) şi capacitatea memoriei reale disponibile pentru paginare variază. Cea mai bună performanţă globală a sistemului este atinsă cînd activitatea de paginare se situează în majoritatea timpului în domeniul identificat în figura 8 ca domeniu de operare. O reducere mai semnificativă a performanţei este observată cînd se atinge domeniul de activitate de paginare activă. O activitate de paginare activă ocazională poate fi considerată, în cazuri de excepţie, ca acceptabilă. O paginare activă mai frecventă poate fi executată pentru a realiza o funcţie dorită, ce nu justifică schimbarea configuraţiei sistemului. Insă, cînd activitatea de paginare în sistem este situată în mod constant la punctul în care are loc deactivarea unei lucrări, trebuie făcute schimbări în configuraţia sistemului pentru a îmbunătăţi performanţa. Astfel de schimbări pot fi mărirea capacităţii memoriei reale, utilizarea unui număr mai mare de dispozitive de memorie auxiliară de paginare sau a unor dispozitive mai rapide, sau instalarea unei UCP mai rapide.

6.1.1.2. Căi de îmbunătăţire a performanţei sistemului cu memorie virtuală paginată.

Tinînd seama de cerinţele suplimentare de resurse, ce apar în cadrul unui sistem operînd cu memorii virtuale, ca şi de factorii noi ce afectează performanţa unui asemenea sistem, studiile întreprinse în această direcţie pun în evidenţă anumite măsuri care pot fi luate cu scopul de a îmbunătăţi performanţa sistemului sau de a atinge un optimum al acestei performanţe.

./.

Asemenea măsuri sînt:

(1) utilizatorul să selecteze judicios proporţia între capacitatea memoriei reale şi capacitatea memoriei virtuale, ţinînd seama de viteza de calcul a UCP;

(2) selectarea unei dimensiuni de pagină, care să fie corespunzătoare capacităţilor sistemului de operare (problema dimensiunii de pagină este discutată în detaliu în capitolul 3, secţiunea 1);

(3) creşterea capacităţii sistemului de tratare a defectelor de pagină, în cazul în care activitatea de paginare conduce în mod constant la deactivarea unor lucrări. Această creştere poate fi realizată prin:

a) utilizarea unor dispozitive de memorie de paginare mai rapide;

b) utilizarea unui număr mai mare de dispozitive de memorie de paginare;

(4) utilizarea unui cod ce nu se modifică pe el însuşi (precum cel produs de limbajul Assembler şi compilatorii limbajului PL/1). Utilizarea unui asemenea tip de cod poate reduce volumul activităţii de transport a paginilor din memoria principală în cea auxiliară;

(5) utilizarea unor memorii tampon de I/E mai mari.

Suplimentar cu reducerea timpului total de I/E necesar pentru tratarea unui set de date (după cum se întîmplă şi într-un mediu cu memorie non-virtuală), această măsură reduce numărul de cereri de I/E necesare pentru a trata setul de date şi prin aceasta reduce timpul consumat de CPU pentru translatarea şi execuţia programului de canal în discuţie.

Luarea acestei măsuri trebuie să ţină seama de asemenea, de capacitatea memoriei reale existente în sistem;

(6) structurarea noilor programe de aplicaţii şi restructurarea programelor de aplicaţii existente pentru a realiza operarea lor eficientă într-un mediu cu paginare (89,145). In realizarea acestei măsuri trebuie urmărită considerarea următoarelor patru aspecte:

a) programele trebuie modularizate;

b) modulele utilizate împreună trebuie dispuse, dacă este posibil, în aceeaşi pagină;

./.

c) modulele utilizate rar trebuie grupate la sfîrşitul programelor;

d) adresele (marcajele) de salt trebuie menţinute în apropierea adresei instrucţiei de salt, de preferat în aceeaşi pagină.

Se urmăreşte prin aceasta ca rutinele utilizate cel mai frecvent să fie conţinute într-un domeniu al programului cît mai mic posibil şi ca un procentaj mai ridicat al codului utilizat cel mai frecvent să fie rezident în memoria reală. Cu alte cuvinte, modificările grupează împreună codul cel mai frecvent utilizat. Determinarea codului cel mai frecvent utilizat se poate face "manual", prin considerarea numărului de ori de care un program execută un modul, sau automat, prin utilizarea unor pachete disponibile de optimizare a codului, precum Cotune, Fortune, Storage III. Analiza comparativă a eficienţei acestor procedee este realizată în (145). Prin aplicarea acestei măsuri se poate realiza o reducere a operaţiilor de paginare cuprinsă între 5o şi 9o%, conform unor aprecieri IBM;

(7) alocarea, în cadrul aplicaţiilor în prelucrarea pe loturi a unei capacităţi de memorie reală, cuprinsă între o jumătate şi o treime din necesităţile totale de memorie (9). Prin această măsură, aplicaţiile critice vor fi tratate în condiţii relativ bune, fără a degrada performanţa altor lucrări în prelucrarea pe loturi.

Aplicaţiilor de teleprelucrare li se alocă memorii pe baza nivelului lor de activitate. După ce au fost asigurate partiţii şi alocată memorie aplicaţiilor critice, pot fi alocate restul programelor şi partiţiilor. Fiecare partiţie ar trebui să primească suficientă memorie reală pentru a permite programului cel mai lung din cadrul grupului să se încadreze, în ceea ce priveşte raportul capacităţilor memoriilor virtuală şi reală, într-un domeniu învecinat valorii de 4:1 (9). Fiecărei partiţii îi trebuie asigurat un domeniu virtual suficient pentru a înmagazina programul cel mai lung. Dacă capacitatea memoriei reale este insuficientă pentru a asigura valoarea menţionată a proporţiei capacităţilor memoriilor, numărul de partiţii trebuie redus

./.

corespunzător;

(8) utilizarea unui algoritm de planificare a priorităţilor ce permite paginarea numai pentru lucrările cu prioritatea cea mai înaltă, în cazul apariţiei situaţiei de thrashing;

(9) utilizarea a două cîmpuri suplimentare (a cîte 1 bit) asociate fiecărei intrări în tabelul de pagini, obţinîndu-se o reducere suplimentară a interferenţei datorate activităţii de paginare. Un asemenea cîmp este cel numit "pagină modificată", ce este poziţionat ø la încărcarea paginii în memoria internă şi este poziţionat 1 la prima instrucţie de scriere, ce face referire la această pagină. Dacă pagina ce trebuie înlocuită nu a fost modificată în timpul utilizării ei (cîmpul de modificare fiind poziţionat 0) ea nu mai este transferată înapoi spre memoria auxiliară, ci este pur şi simplu ştearsă.

Al doilea cîmp denumit "pagină utilizată" este utilizat la descrierea statistică a frecvenţei de utilizare relativă a paginilor. Acest cîmp este poziţionat 1 la orice referire a paginii respective; la anumite intervale o rutină a sistemului de operare examinează acest cîmp, notează starea lui şi îl poziţionează 0. Pe baza valorilor acestui cîmp se pot lua decizii privind înlocuirea unei pagini (secţiunea 3.4);

(10) utilizarea unei proceduri de paginare fracţionată.

Posibilitatea de a transfera din memoria virtuală în cea reală fracţiuni de pagini ar conserva memorie reală şi ar reduce sarcina de cicluri furate*), impusă prin activitatea de paginare, care este deosebit de risipitoare, în cazul în care părţi mari ale paginilor nu sînt folosite. S-ar crea astfel şi posibilitatea ca programatorii să poată scrie programe mai eficiente, decît acelea a căror dimensiune este un multiplu întreg al dimensiunii paginii utilizate de sistemul de operare;

---

*) furt de cicluri (cycle-stealing) - tip comun de obţinere a ciclurilor de memorie pentru transferurile I/E, în care UCP este blocată unul sau mai multe cicluri, dacă este necesar, de către un transfer I/E pentru a permite ca datele să fie introduse sau extrase din memorie, "furînd" astfel cicluri ale UCP. Acest tip de operare încetineşte funcţionarea UCP, căci aceasta are mai puţine cicluri utilizabile într-o unitate de timp.

./.

(11) utilizarea, în cadrul sistemului, a mai multor dimensiuni distincte de pagini. In acest mod se realizează utilizarea mai eficientă a capacităţii memoriei reale şi, ca şi mai sus, se creează posibilitatea producerii unor programe mai eficiente;

(12) analiza ciclurilor programului.

Compilatorul împarte programul în blocuri de instrucţii şi cercetează interacţiunea lor. Intenţia este de a recunoaşte cicluri şi de asemenea cicluri incluse în alte cicluri, cu scopul de a permite simplificarea ciclului cel mai interior. Compilatorul rearanjează instrucţiile astfel încît să permită ciclului încadrarea într-o singură pagină (sau mai multe pentru un ciclu lung), ce ar fi situate, în timpul executării ciclului, în memoria reală.

O analiză a cauzelor ce conduc la apariţia situaţiei de thrashing în activitatea de paginare şi recomandarea unor măsuri de prevenire a ei pot fi găsite în (55).

6.1.2. Memorie virtuală segmentată.

In schema de segmentare Burroughs sistemul de operare împarte codul obiect produs în segmente logice de program, corespunzător specificaţiilor făcute de programator.

Fiecare program constă dintr-un segment de bază nesuperpoziţionabil şi segmente asociate superpoziţionabile (overlayable).

Segmentul de bază conţine toate domeniile de date, secţiunile cu fişiere I/E şi toate celelalte domenii ale programului ce sînt, în orice mod, subiect de modificări; de aceea pentru un program activ segmentul de bază trebuie să fie rezident în memoria principală. El trebuie să prevadă legarea cu celelalte segmente dispuse într-o memorie auxiliară cu acces direct. Deoarece codul de program situat în această memorie auxiliară nu este niciodată modificat, segmentele nebazice active anterior nu trebuie să fie transferate înapoi în memoria auxiliară, în situaţia cînd un nou segment este adus în memoria principală. Această metodă de segmentare îşi găseşte un echivalent în tehnicile de programare standard rădăcină/superpoziţie în folosinţă de mulţi ani.

Memoria virtuală paginată utilizează, după cum am văzut, blocuri mici - pagini - de informaţii, ce trebuie să fie manipulate între memoria virtuală şi cea reală.

Cu scopul de a evita cheltuielile mari, implicate de tra-

./.

tarea software a numeroaselor cereri de I/E ale schimbului continuu de pagini, producătorii de sisteme de memorie virtuală paginată au găsit necesar să implementeze porţiuni semnificative ale acestui mecanism în hardware.

Burroughs a implementat memoria virtuală segmentată în software, utilizînd segmentul rădăcină ca fundament.

Deoarece segmentele non-rădăcină nu sînt niciodată transferate înapoi în memoria virtuală, activitatea de thrashing este un fapt aproape imposibil în cadrul acestui sistem.

Avînd în vedere însă că numai codul nemodificabil poate fi memorat în memoria virtuală, există în mod potenţial o mare risipă de memorie, în special deoarece domeniile tampon trebuie să fie menţinute chiar atunci cînd fişierele nu sînt referite.

Pe de altă parte, software-ul Burroughs necesită alocarea continuă a spaţiului în memoria reală, implicînd alocarea unui spaţiu egal cu dimensiunea segmentului celui mai mare al programului. Dacă acest spaţiu nu poate fi rezervat în memoria reală, execuţia programului nu poate fi iniţiată. Datorită, însă, Master Control Program - MCP, şansele ca programul să fie în situaţia de a nu putea fi rulat, ca urmare a lipsei de memorie continuă, sînt reduse semnificativ, datorită facilităţii numită sistem de comprimare a memoriei.

Comprimarea memoriei este iniţiată automat de MCP. Cînd o lucrare se termină, MCP examinează şirul lucrărilor în aşteptare pentru a determina dacă una dintre ele s-ar încadra în domeniul eliberat. Dacă da, acea lucrare este iniţiată; dacă nu, lucrările în tratare sînt deplasate (comprimate) în memorie, pentru a ocupa domeniul eliberat. Spaţiul făcut disponibil de către lucrările comprimate este continuu şi este alocat unei noi lucrări. Astfel, programatorii, deşi nepreocupaţi de cerinţele generale de memorie ale unui program, trebuie să ia în consideraţie cerinţa de nemodificare a segmentelor, dimensiunea absolută a segmentu-

./.

lui celui mai mare şi dimensiunea relativă a restului de segmente. Segmentul cel mai mare trebuie să se încadreze în spaţiul de memorie reală disponibilă pentru program, pentru a obţine posibilitatea de a apela la sistem. Segmentele trebuie să fie de lungimi aproximativ egale, pentru a evita risipa ce ar putea fi cauzată prin plasarea segmentelor mici în spaţiul de memorie reală rezervat pentru segmente mult mai mari.

6.2. Sisteme cache.

Sistemele cache sînt bazate pe o ierarhie cu două nivele: cache - principală (poziţiile 1 şi 2 sau 3 din tabelul 1).

Primele sisteme descrise de acest tip folosesc ca unitate de transfer între cele două nivele ale memoriei cuvîntul.

Vom prezenta sumar două asemenea tipuri de sisteme:

(1) sistem cu memorie "slave".

Una din primele lucrări privitoare la ILVII (154) introduce conceptul de memorie slave.

Succesul cu care anumite memorii tampon operînd la viteze electronice a mărit viteza de execuţie a anumitor funcţii asociate cu controlul unor calculatoare convenţionale, a condus la această idee. Sistemul propus combină o memorie cu miezuri cu ciclul de 1 μs (slave) cu o memorie cu miezuri mai lentă cu ciclul de 8 μs (memorie principală) cu scopul de a obţine pentru combinaţia realizată a unui ciclu mediu ce se apropie de acela al memoriei slave. Termenul "slave" derivă din faptul că toate referirile la memorie sînt obţinute de UCP din memoria slave; dacă o referire nu este găsită acolo, este adusă în memoria slave din memoria principală. Sînt propuse două scheme de organizare.

Una se aplică la cazul unei memorii slave de capacitate mică (32 cuvinte): cuvintele din memoria principală sînt transpuse în cuvinte ale memoriei slave, astfel încît nu este necesară nici o organizare asociativă. Această organizare, rapidă şi simplă, are unele dezavantaje, ce vor deveni clare mai tîrziu.

A doua schemă de organizare, cunoscută ca partiţie - slave, se referă la o memorie - slave mare, de 32 K. In această organizare, cîteva blocuri de program îşi divid memoria-slave;

./.

(2) sistem cu memorie look-aside.

A devenit un fapt cunoscut că programele nu-şi distribuie aleator referirile lor în spaţiul adreselor, ci au tendinţe să opereze în anumite domenii de informaţii. Memoria look - aside a fost creată pentru a face uz de această proprietate cu scopul de a creşte rata de instrucţii a calculatoarelor convenţionale.

Acest tip de memorie este format din registre asociative. Fiecare registru asociativ conţine o cheie formată din două părţi. Partea de adresă conţine adresa cuvîntului pe care registrul îl memorează şi partea de uz conţine unele informaţii în legătură cu algoritmul de înlocuire utilizat de către sistem (LRU). Dacă partea de adresă a cheii în registrul asociativ corespunde adresei unei anumite referiri, o astfel de referire este obţinută din memoria look-aside; dacă nu se poate stabili nici o corespondenţă, atunci acea referire este obţinută din memoria principală, ce constituie al doilea nivel al ierarhiei.

Selectarea cuvîntului ce trebuie înlocuit cu scopul de a dispune noua referire, este făcută conform unui algoritm de înlocuire, cablat în sistem.

Toate registrele asociative, sînt examinate simultan, aşa că întîrzierea provocată este mică.

Cu această structură au fost obţinute unele rezultate încurajatoare, principalul fiind următorul. Dacă un calculator convenţional dispune de o memorie principală cu miezuri cu ciclul de 1 μs, 128 registre asociative sînt suficiente pentru a obţine un ciclu efectiv de memorie pentru combinaţie de 350-400 ns.

Cercetările ulterioare au pus problema utilizării ca unitate de transfer între cele două nivele de memorii a blocului, format din mai multe cuvinte. La prima vedere poate să nu apară să fie prea avantajos a transfera din M_2 în M_1 mai mult decît un cuvînt odată; dacă timpul pentru a transfera un anumit număr de cuvinte creşte linear cu numărul de cuvinte, (ca în cazul unei memorii look-aside) transferarea cîtorva cuvinte odată ar fi o risipă, numai dacă toate cuvintele transferate nu vor fi

./.

referite în viitorul apropiat. Dacă însă un grup de cuvinte (un bloc) poate fi transferat din memoria principală cu cheltuieli de timp mici, situaţia este diferită. Motivul pentru care se urmăreşte utilizarea unei unităţi de transfer formate din mai multe cuvinte rezidă în natura programelor de calculator şi anume în faptul că o referire dată este probabil de a fi urmată de o alta, care este localizată fie secvenţial, fie în imediata ei vecinătate în memoria principală.

Pentru a realiza transferul unui bloc de cuvinte din M_2 în M_1 este necesar de a crea o lăţime de bandă suficientă. Aceasta se poate obţine prin creşterea dimensiunii căii de date şi prin intercalare. Intercalare înseamnă că M_2 este împărţită în cîteva module de memorie şi că mai multe referiri consecutive sînt memorate succesiv în fiecare modul.

In continuare vom da o descriere şi comparare calitativă a diferitelor organizări posibile ale memoriei M_1.

Există patru scheme:

(1) organizarea în sector. Ambele memorii M_1 şi M_2 sînt împărţite în sectoare, a căror dimensiune este de cîteva ori mai mare decît dimensiunea selectată a blocului. Fiecare sector este împărţit în blocuri; fiecărui sector din M_1 îi este asociat un identificator cu scopul de a determina sectorul din M_2 al cărui conţinut îl păstrează.

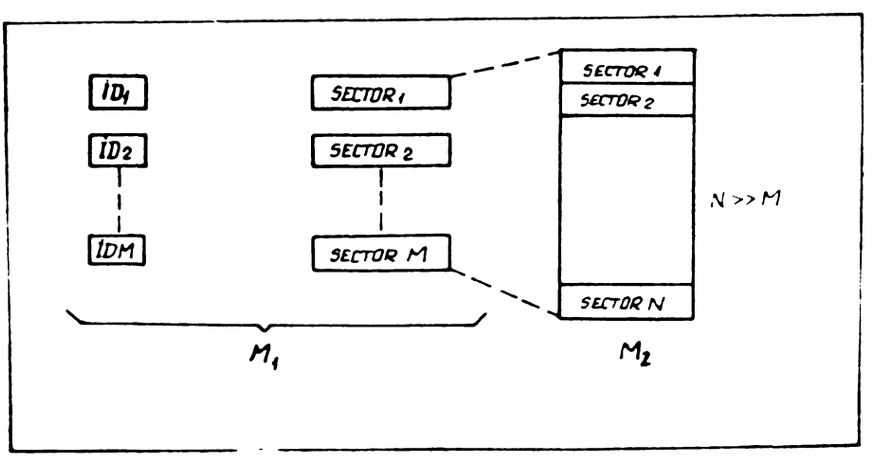


Fig.9. Organizare în sector.

Deoarece sectoarele sînt mari şi M_1 este, în general, doar o fracţie din M_2, în M_1 există relativ puţine sectoare şi

./.

prin urmare puţini identificatori. Aceasta permite ca toţi identificatorii să fie examinaţi asociativ simultan.

Într-un sector există, asociat cu fiecare bloc, un bit de "validitate" ce certifică prezenţa blocului în sectorul de care el aparţine. Aceasta este realizată din următorul motiv: lipsa unui sector în M_1 va provoca înlocuirea unui alt sector prezent în M_1, dar, deoarece dimensiunea unui sector este mare, blocurile sînt aduse în sector la cerere, unul cîte unul, în loc de a transfera întregul sector. Prin urmare, dacă un bloc este prezent într-un sector, bitul de validitate trebuie să fie poziţionat 1.

Orice sector din M_2 poate fi dispus în orice sector al lui M_1; identificatorul stabileşte adresa primului cuvînt într-un sector; la compararea cu succes, blocul dorit este identificat prin anumiţi biţi din compunerea adresei, iar cuvîntul sau octetul dorit de asemenea prin biţi corespunzători din adresă.

Avantajul principal al acestor organizări constă în viteza cu care cuvîntul poate fi extras din M_1, ca o consecinţă a numărului redus de identificatori; acest lucru implică la rîndul lui cheltuieli mici de timp în execuţia algoritmului de înlocuire.

Un dezavantaj serios apare din necesitatea de a înlocui un întreg sector, ori de cîte ori un sector nou trebuie să fie adus în memorie. Aceasta ar putea fi deosebit de critic dacă sistemul este multiprogramat sau dacă operează în regim cu diviziune în timp; în acest caz programele şi-ar înlocui sectoarele unele altora şi performanţa s-ar înrăutăţi.

IBM 360-85 a fost primul calculator ce a utilizat această schemă de organizare (115), căreia i-a fost dat numele de cache. Dimensiunea sectorului a fost aleasă de 64 blocuri a 16 cuvinte fiecare şi dimensiunea lui M_1 este de 16 sectoare.

(2) transpunere directă.

Ambele memorii M_1 şi M_2 sînt împărţite în blocuri, a căror dimensiune este apropiată de dimensiunea optimă de bloc; fiecare bloc are un identificator, ce precizează biţii de or-

./.

din superior ai adresei primului cuvînt în bloc.

Dispunerea blocurilor din M_2 în M_1 este realizată conform unei reguli anumite stabilite; corespunzător raportului între capacităţile memoriilor c_1 şi c_2 vor exista mai multe sau mai puţine blocuri din M_2 ce vor concura pentru aceeaşi poziţie în M_1.

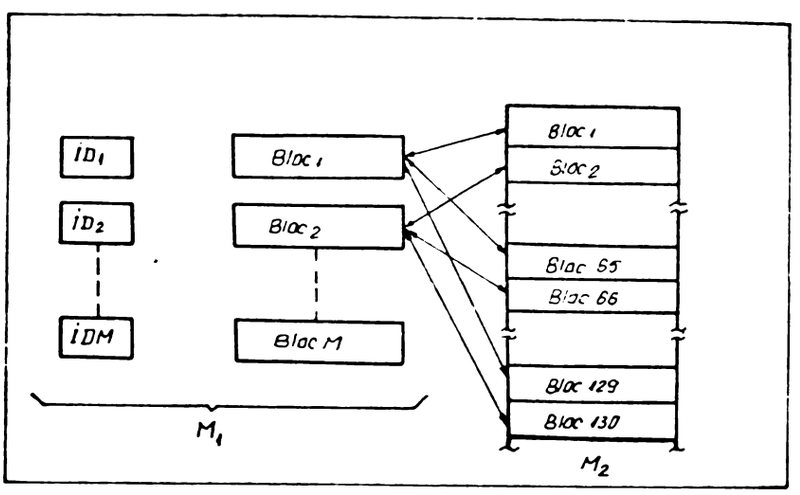


Fig.10. Transpunere directă.

Fiind dată o anumită referire la memorie, există una şi numai o poziţie unde ea ar putea fi găsită în M_1; identificatorul ce corespunde acestei poziţii este examinat şi dacă se constată concordanţa, cuvîntul sau octetul dorit este precizat cu ajutorul unor anumiţi biţi de ordin inferior din compunerea adresei. Dacă blocul nu este găsit în M_1, el este scos din M_2 şi transferat în M_1 în locaţia de bloc corespunzătoare.

Avantajul principal al acestei scheme de organizare este faptul că nu este solicitată nici o cercetare asociativă şi prin urmare adresarea M_1 este foarte rapidă.

Există totuşi o deficienţă importantă a acestei organizări: este foarte posibil ca două referiri de memorie să conducă la transferuri din M_2 în acelaşi bloc din M_1, ce ar implica necesitatea unui ciclu de M_2 pentru o referire. Această situaţie este referită ca "dispută";

(3) organizarea complet asociativă.

Ca şi în cadrul schemei de organizare precedente, ambele M_1 şi M_2 sînt împărţite în blocuri de dimensiune egală. Fiecărui

./.

bloc din M_1 îi este asociat un identificator. Acest identificator este alcătuit din biţii de ordin superior ai adresei primului cuvînt prezent în bloc. Fiecare bloc din M_2 poate fi transpus în orice bloc din M_1. Ca o consecinţă, pentru a stabili prezenţa unui anumit bloc în M_1 este necesară examinarea tuturor identificatorilor. Această procedură de examinare este realizată într-o memorie asociativă într-un mod serial-paralel.

Din punctul de vedere al " disputei", aceasta este organizarea ideală; de asemenea, dacă cîteva programe îşi divid M_1, fiecare ar putea avea M_1 efectivă maximă cu acest mod de organizare.

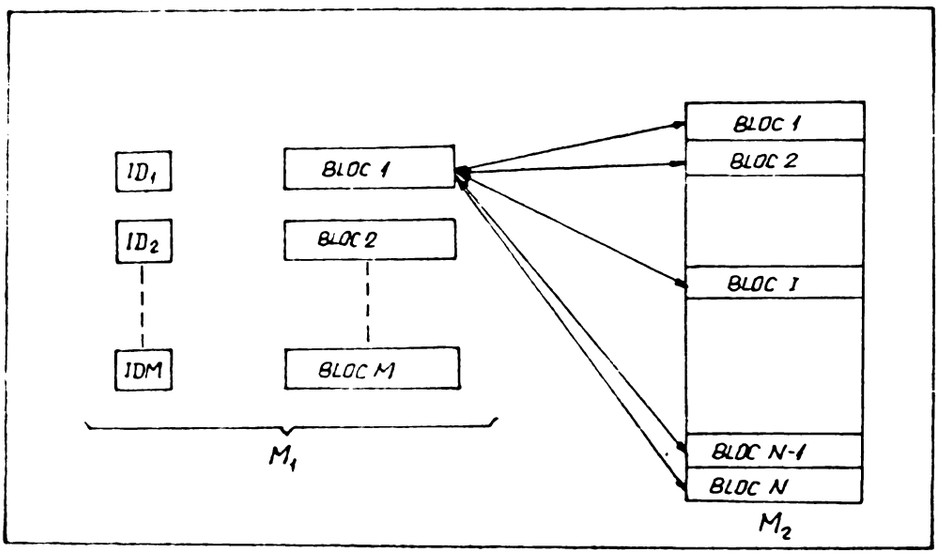


Fig.11. Organizare complet-asociativă.

Singura deficienţă a organizării complet asociative este timpul cerut pentru cercetarea asociativă pentru toţi identificatorii; în particular, dacă M_1 este foarte mare, cheltuielile implicate ar putea fi prohibitive.

(4) organizare grup-asociativă.

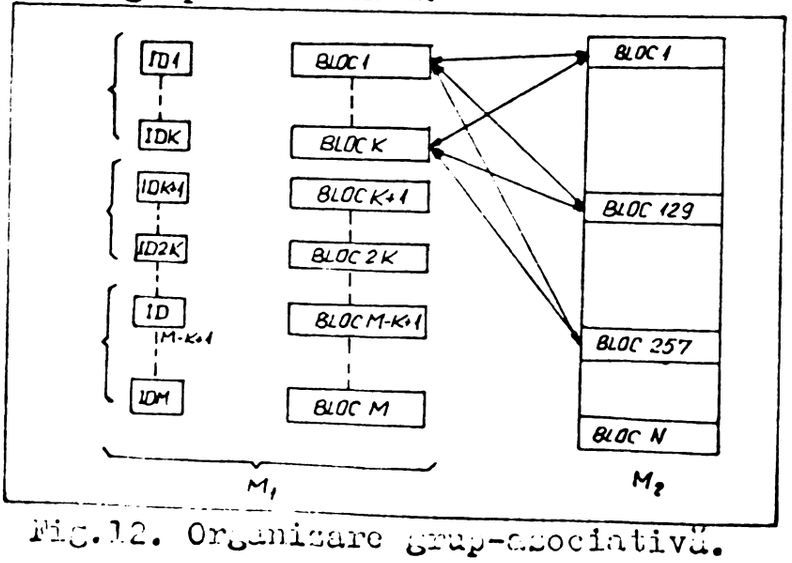


Fig.12. Organizare grup-asociativă.

Această organizare combină tehnicile de transpunere directă şi complet asociativă. Ambele M_1 şi M_2 sînt împărţite în blocuri de dimensiuni egale. Blocurile din M_1 sînt clasificate în grupuri de blocuri, fiecare grup conţinînd acelaşi număr de blocuri; blocurile din M_2 dispun în blocurile lui M_1 într-un grup anumit,dat. Deşi este obligat să selecteze un anumit grup, blocul din M_2 poate dispune în oricare din blocurile acestui grup.

Dacă numărul blocurilor cuprinse într-un grup este egal cu 1, schema este identică cu transpunerea directă. Dacă pe de altă parte dimensiunea unui grup este egală cu dimensiunea M_1, se regăseşte schema complet asociativă.

După cum poate fi văzut, această organizare reţine avantajul transpunerii directe şi a organizării complet asociative; dacă cheltuielile implicate în schema complet asociativă sînt mari, această organizare ar putea fi preferată.

Aceasta este organizarea folosită în ierarhia de memorii a calculatorului IBM 370/155 (104). In acest calculator dimensiunea fiecărui grup este de 4 blocuri a 8 cuvinte logice fiecare; numărul total de grupuri existente în M_1 este 64.

Proiectanţii acestor sisteme cache au reuşit să reducă drastic proporţia $c_1/(c_1+c_2)$ pînă la 1% şi să menţină încă proporţia succeselor s_1 la peste 90%.

Rezultate similare au fost publicate de Bell şi Casasent (21), Mattson (120), Meade (124) şi Seligman (148).

O îmbunătăţire structurală, relativă la utilizarea memoriilor cache este realizată în cadrul sistemului ICL 2900 (100), instrucţiile şi operanzii sînt adresaţi de către procesor din trei memorii rapide slave. Fiecare din aceste trei memorii există pentru a servi o anumită fază de tratare a lanţului de instrucţii: prima memorie conţine instrucţii, a doua operanzi primi şi ultima operanzi secunzi. Această organizare asigură o viteză sporită de tratare a lanţului de instrucţii.

6.3. Sisteme cu trei nivele.

In literatură au fost publicate puţine lucrări, ce să cuprindă cercetarea unor asemenea sisteme.

Au fost studiate însă cel puţin trei tipuri de astfel de ierarhii:

./.

(1) ierarhie de memorii, bazată pe utilizarea dispozitivelor 2,3 şi 9 din tabelul 1. Cea mai consistentă cercetare este cea întreprinsă la Universitatea Carnegie-Mellon (71). Rezultate au fost de asemenea publicate de Williams (166);

(2) ierarhie de memorii bazată pe utilizarea dispozitivelor 3,8 şi 9 din tabelul 1. Cercetarea a avut loc în legătură cu exploatarea sistemului Multics din cadrul proiectului MAC al MIT. Ca o indicaţie a efectului ei, s-a arătat că noua strategie a condus la creşterea proporţiei succeselor s_2 de la 20% la mai mult de 90%;

(3) ierarhie de memorii bazată pe utilizarea următoarelor dispozitive: primul nivel corespunde dispozitivului 3 din tabelul 1, nivelul secund corespunde la o combinaţie a dispozitivelor 8 şi 9 şi al treilea nivel poate fi aproximat prin dispozitivul lo al tabelului 1. Deşi din datele lucrării lui Considine şi Weis (43) este imposibil să se calculeze proporţiile succeselor, rezultă totuşi că pentru $c_2/(c_2+c_3)=0,5$, s_2 este foarte mare (apropiat de unitate).

CAP.3. ANOMALIA DIMENSIUNII DE PAGINA

1. Alegerea dimensiunii paginii utilizate în cadrul unei ILVM.

Un parametru important al unei ILVM (din categoria celor de configuraţie - vezi cap.2 - secţiunea 3.2.) este dimensiunea paginii, unitatea de informaţii transferată între nivelele vecine ale ierarhiei. Intrucît acest parametru afectează performanţa unei ILVM, s-a acordat o atenţie deosebită determinării unei dimensiuni optime de pagină. Astfel s-a constatat că factorii principali ce influenţează alegerea dimensiunii de pagină sînt (1) fragmentarea memoriei şi (2) eficienţa opêraţiei de transport a paginii.

1.1. Fragmentarea memoriei.

Prin fragmentarea memoriei se are în vedere împărţirea memoriei disponibile într-un număr mare de blocuri, separate prin domenii ocupate (nedisponibile). Acest fenomen este de natură statistică şi apare cînd blocuri de dimensiuni variabile sînt alocate în memorie şi apoi eliberate după intervale variabile de timp.

După cum se observă în cadrul unor sisteme de operare convenţionale, aceste spaţii nefolosite din memoria principală situate între partiţii au tendinţa de a se dezvolta pe măsură ce sarcina de lucru zilnică se amplifică. Fiecare din aceste domenii libere disponibile este prea mic pentru a dispune în el unul din programele plasate în şirul de aşteptare cerînd spaţiu de execuţie continuu de o anumită dimensiune, deşi suma capacităţilor domeniilor libere este suficientă. Această situaţie conduce la scăderi în utilizarea memoriei utilizate, care este măsurată prin procentul din capacitatea totală a memoriei , a capacităţii memoriei ocupate de activităţile curente şi există două alternative:

(1) să se admită această utilizare mai scăzută a memoriei;

(2) apelînd la o anumită cheltuială de resurse ale sistemului, să se realizeze reorganizarea conţinutului memoriei astfel încît să se unească blocurile separate de memorie disponibilă.

După cum s-a arătat în secţiunea 6.1.2.a cap.2 o soluţie de tipul (2) a fost adoptată în cadrul schemei de segmentare

./.

Burroughs - funcţia de comprimare a spaţiului ocupat de lucrările în tratare şi de formare a unui spaţiu liber continuu din fragmentele mici dispersate este realizată de MCP-Master Control Program.

O lucrare ce se ocupă cu studiul fenomenului de fragmentare al memoriei este (137). Vom prezenta pe scurt conţinutul şi concluziile acestei lucrări.

Pentru cercetarea fragmentării memoriei s-a recurs la o serie de experimente de simulare. In cadrul modelului utilizat memoria principală este reprezentată printr-o listă de blocuri de diferite dimensiuni, ficeare element din listă fiind însoţit de o indicaţie, dacă blocul este ocupat şi, în caz afirmativ, cînd va fi eliberat. Elementele vecine în această listă se referă la blocuri din memorie vecine fizic. Cererile de spaţiu de memorie sînt caracterizate prin mărimea spaţiului cerut şi lungimea de timp cerută, ambele valori fiind alese din distribuţii statistice. Sirul de cereri a fost considerat ca fiind ordonat şi nu s-a căutat intercalarea unor cereri de spaţiu mai mic, în situaţia în care cererea actuală nu putea fi satisfăcută.

Prin model s-a cercetat efectul diferiţilor factori asupra utilizării memoriei, caracterizată de capacitatea memoriei ocupate de cererile alocate. Unul din aceşti factori indică ce algoritm de plasare (strategia pentru a decide unde ar trebui alocată în memorie o cerere pentru spaţiu de memorie) este utilizat în cadrul experimentului. In model au fost cuprinşi trei algoritmi distincţi de plasare:

(a) MIN - acest algoritm întreţine o listă a blocurilor disponibile, ordonate după creşterea dimensiunii. O cerere de memorie este alocată în blocul cel mai mic disponibil, ce are o dimensiune suficientă pentru a satisface cererea. Blocurile ocupate de memorie nu sînt deplasate pentru a uni porţiunile separate de memorie disponibilă;

(b) ALEATOR - alegerea blocului pentru a aloca cererea se face în mod aleator din întreaga mulţime a blocurilor disponibile, ce sînt suficient de mari. La fel ca în cazul algoritmului (a), blocurile ocupate nu sînt deplasate;

./.

(c) RELOC - de fiecare dată ce un bloc ocupat devine disponibil, blocurile alăturate la dreapta lui sînt deplasate astfel încît spaţiul disponibil să fie mereu menţinut într-un bloc continuu, la sfîrşitul memoriei. Fiecare cerere este alocată în memorie, începînd cu cuvîntul de la începutul acestui bloc disponibil.

Algoritmii MIN şi ALEATOR au fost aleşi ca fiind situaţi în apropierea celor două extreme ale spectrului algoritmilor de plasare fără relocare, în ceea ce priveşte complexitatea şi nivelul performanţei. Algoritmul RELOC a fost ales în special ca termen de comparaţie, indicînd cîştigul maxim posibil ce s-ar obţine utilizînd relocarea.

Se consideră ca scăderi în utilizarea memoriei acele situaţii în care o cerere de memorie nu poate fi alocată în memoria principală, deşi cererile deja alocate în memorie în acest moment nu ocupă în întregime memoria.

In lucrare se face distincţia între două tipuri diferite de fragmentare a memoriei:

(1) fragmentare externă: scăderea în utilizarea memoriei cauzată de imposibilitatea de a face uz de toată memoria disponibilă, după ce ea a fost fragmentată într-un număr mare de blocuri separate. Astfel, diferenţa între utilizarea memoriei obţinută în cadrul unui experiment utilizînd ca algoritmi de plasare MIN sau ALEATOR şi cea obţinută în cadrul unui experiment utilizînd algoritmul RELOC este datorată fragmentării externe, pe care cei doi algoritmi o cauzează;

(2) fragmentarea internă, ce se referă la scăderea în utilizarea memoriei cauzată prin rotunjirea unei cereri de memorie la o valoare de alocare, în locul unei alocări de numai exact numărul cerut de cuvinte.

Cererile de memorie au fost rotunjite la un multiplu (cel mai apropiat) a unei unităţi de alocare Q.

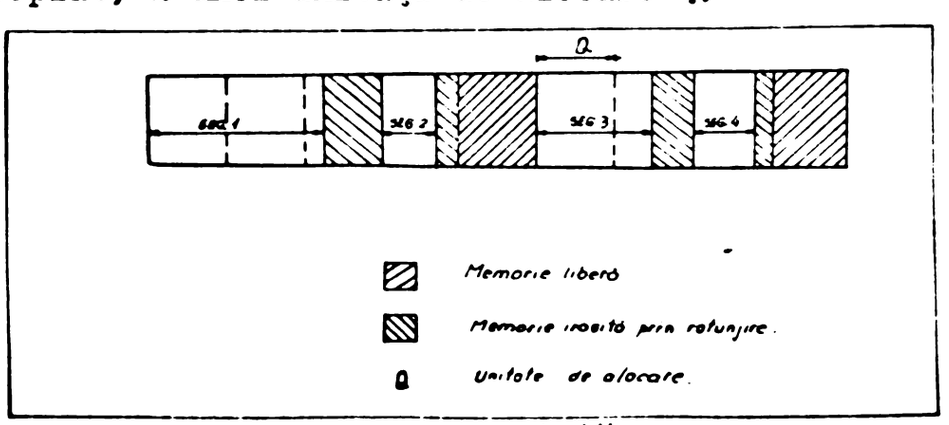


Fig.13 Memorie fragmentată

Pe măsura creşterii dimensiunii unităţii de alocare Q, fragmentarea externă se micşorează, dar se măreşte fragmentarea internă; ea corespunde spaţiului de memorie irosit în cadrul fiecărui bloc prin procesul de rotunjire a cererii de memorie.

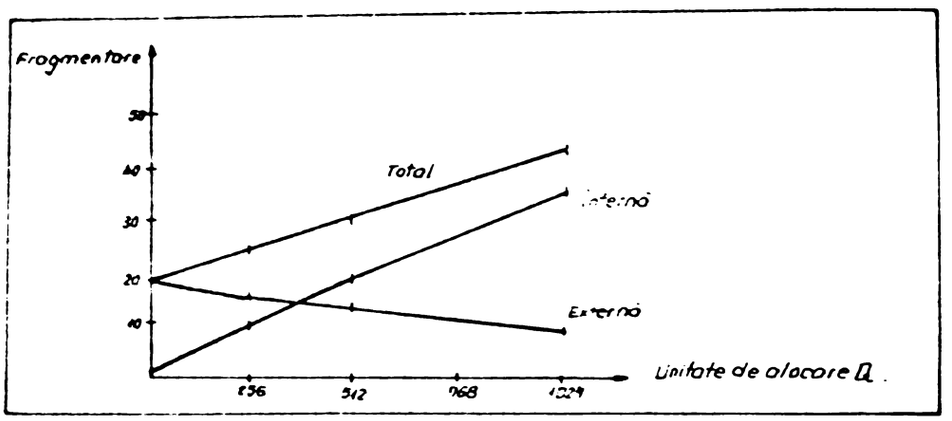


Fig.14. Efectele rotunjirii cererilor de memorie

Figura 14 reprezintă o curbă reprezentativă a tipului de rezultate obţinute. Ea a fost obţinută în cadrul unui experiment utilizînd o distribuţie exponenţială a dimensiunilor cererilor de memorie, cu o medie de 1024 cuvinte, dimensiunea memoriei fiind de 32.768 cuvinte.

Rezultate similare au fost obţinute utilizînd algoritmii FIN şi ALEATOR şi de asemenea pentru diferite distribuţii şi valori medii ale dimensiunilor cererilor de memorie.

Rezultatul cel mai interesant este că pe măsura creşterii lui Q, scăderea în utilizarea memoriei datorată fragmentării interne mărite are o valoare mai mare decît cîştigul datorat fragmentării externe scăzute. Acest efect se datorează faptului că pe măsura creşterii dimensiunii unităţii de alocare se măreşte de asemenea spaţiul de memorie alocat, dar nefolosit (deci irosit).

3.2. Eficienţa operaţiei de transport a paginii.

Timpul t_{di} necesar pentru a deplasa o pagină între două nivele ale ierarhici (de la nivelul i la nivelul i-1) constă din:

(1) timpul mediu de acces t_i;

(2) timpul de transfer: p_i/r_i, p_i fiind dimensiunea paginii, iar r_i fiind rata de transfer, relativă la deplasarea

./.

informaţiilor între cele două nivele.

Dacă toate dimensiunile de pagină au fost alese pentru a prevedea volumul de informaţii p_i cerut de procesor, timpul de deplasare al paginii între nivele este:

$$t_{di} = t_i + p_i/r_i$$

Pentru referiri concrete, vom considera un sistem de memorii compus din dispozitivele 1,3,8,9 şi 10 din cadrul tabelului 1 (mai frecvent întîlnite la sistemele de calcul curente). In tabelul 2 au fost indicate timpul de acces şi rata maximă de transfer pentru fiecare din aceste dispozitive:

Nr.	Dispozitivul	Timpul de acces aleator (sec.)	Rata max. de transfer (octeţi/sec.)
1	RAM-TTL	$60 \cdot 10^{-9}$	$1 \cdot 10^{8}$
3	RAM-MIEZURI	$500 \cdot 10^{-9}$	$2 \cdot 10^{7}$
8	DISC CAPETE FIXE	$8 \cdot 10^{-3}$	$3 \cdot 10^{6}$
9	DISC CAPETE MOBILE	$50 \cdot 10^{-3}$	$8 \cdot 10^{5}$
10	BANDA MAGNETICA	10	$6 \cdot 10^{5}$

Tabelul 2. Timpul de acces şi rata de transfer a unor dispozitive de memorii

Examinînd tabelul 2 se constată că timpul de acces variază considerabil mai mult decît rata de transfer: timpul de acces acoperă 9 ordine de mărime, în timp ce rata de transfer doar 3.

Această diferenţă pune problema avantajelor posibile de obţinut prin transferul, în cadrul unui singur acces, a unui volum sporit de informaţii. Astfel, dorim a stabili creşterea lui t_{di}, dacă, suplimentar la cei p_1 octeţi ai paginii, se realizează transferul a încă p_1 octeţi, imediat următori. Presupunînd că p_1 este destul de mic, de exemplu 8 octeţi (am ales această valoare ţinînd seama de lăţimea benzii de date între nivelele 1 şi 2 ale unei ierarhii), creşterile corespunzătoare ale timpilor t_{di} sînt trecute în tabelul 3.

./.

Tabelul 3

Transport între nivele	Timp transport pentru 1 bloc	Timp transport pentru 2 blocuri	Creşterea procentuală a timpului de transport
3 la 1	0,9 μs	1,3 μs	44%
8 la 3	8003 μs	8005 μs	0,03%
9 la 8	50010 μs	50020 μs	0,02%
10 la 9	10000013 μs	10000027 μs	0,0002%

Creşterea timpilor de transport a paginilor.

Din tabelul 3 constatăm că creşterea procentuală a timpului de transport la transferul suplimentar a p_1 octeţi scade de la o valoare ridicată de 44%, obţinută în cadrul transferului de la nivelul 3 spre nivelul 1, la o valoare mică de 0,00002% în cadrul transferului de la nivelul 10 la nivelul 9.

Această situaţie este deosebit de favorabilă, avînd în vedere comportarea localizată a programului (vezi secţiunea privind conceptul de localizare) şi anume: avînd în vedere acest caracter de localizare în mostra de referiri a programului la spaţiul adreselor ne putem aştepta ca probabilitatea p_r ca procesorul să facă referiri în intervalul de timp imediat următor la cei p_1 octeţi alăturaţi transmişi suplimentar să fie considerabil mai mare decît inversul dimensiunii spaţiului de adrese logice. Astfel, pentru un nivel dat al ierarhiei, dacă probabilitatea p_r este mai mare decît creşterea procentuală a timpului de transport, este avantajos să transferăm p_1 octeţi suplimentari şi prin aceasta să evităm necesitatea de a cheltui t_{ci} sec. pentru a transporta aceşti p_1 octeţi separat, mai tîrziu.

După cum rezultă din tabelul 3 aceste avantaje sînt substanţiale în cadrul transferurilor între nivelele mai joase ale ierarhiei, caracterizate în special prin timpi de acces lenţi. Pentru nivelele superioare ale ierarhiei, în speţă dispozitive de memorii cu timpi de acces medii rapizi, creşterea timpului de transport la transferul suplimentar a p_1 octeţi este considerabilă şi avantajele posibile de obţinut în urma acestui transfer mai voluminos sînt mult mai mici.

./.

O ultimă observaţie. Deoarece creşterea procentuală a t_{di} descreşte monoton ca funcţie de nivelul dispozitivului de memorie, numărul de blocuri a câte p_1 octeţi, urmînd a fi transferaţi ca o singură pagină, ar trebui să crească uniform. Astfel, în cadrul unei ILVM ar fi avantajos ca între dimensiunile de pagini p_i utilizate să existe relaţia de ordonare:

$$p_1 < p_2 < \cdots < p_l$$

1.3. Alte consideraţii privind dimensiunea paginii. Concluzii.

Joseph (102) cercetează variaţia proporţiei eşecurilor (vezi cap.2, secţiunea 4.1.) în funcţie de numărul de pagini din program aflate în memorie (alocare), în cazul unor diferite dimensiuni de pagină: (32, 128 şi 1024 cuvinte) - în fig.15. proporţia eşecurilor e reprezentată într-o scară logaritmică.

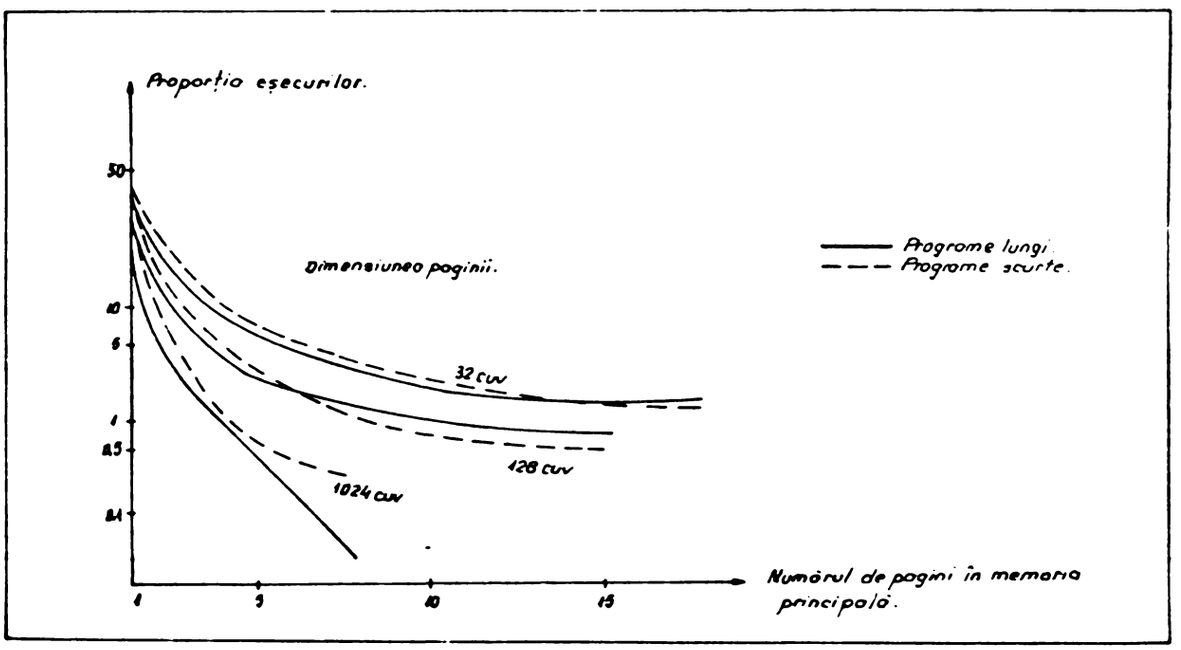


Fig.15. Proporţia eşecurilor.

Curbele reprezentate prin linii continue se referă la programe lungi, iar cele reprezentate prin linii întrerupte se referă la programe scurte; se observă încă că între ele nu există o diferenţă substanţială.

Un fapt important de constatat este cel, de exemplu, că o alocare de 5 pagini a 32 cuvinte fiecare, conduce la o propor-

./.

ţie a eşecurilor mai mică decît o alocare de 2 pagini fiecare a 1024 cuvinte. Deci în situaţia în care se dispune în memoria principală de un spaţiu mic pentru alocarea unui program (lucru curent în sistemele cu multiprogramare), paginile de dimensiuni mici se prezintă ca mai avantajoase. In ceea ce priveşte dimensiunea de 1024 cuvinte, utilizarea unei asemenea dimensiuni de pagină aduce o îmbunătăţire a performanţei faţă de dimensiunile mai mici de pagină numai după ce numărul de pagini alocate în memorie depăşeşte 5 (în cazul cercetat, o alocare de 5 K pentru un program de 8 K).

In figura 16 este reprezentată variaţia proporţiei eşecurilor în funcţie de dimensiunea paginii, pentru un număr fix de cuvinte în memorie (alocare în cuvinte). Din nou se observă clar că în timp ce pentru alocări mici de memorie este mai eficientă utilizarea unor pagini de dimensiuni mai mici, la o alocare de peste 4 K utilizarea paginilor mai mari tinde să devină mai eficientă.

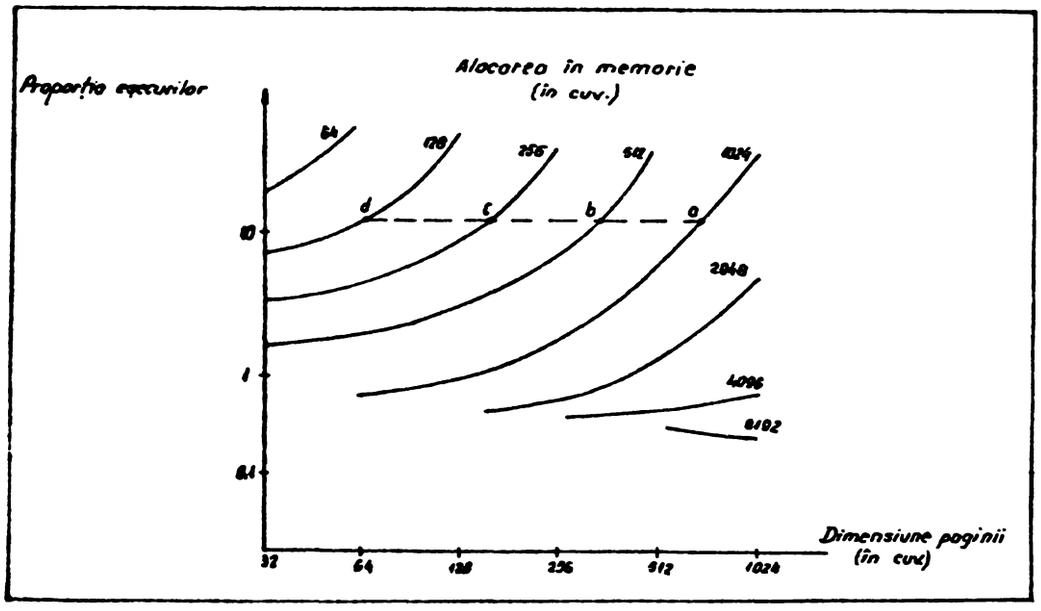


Fig.16. Variaţia proporţiei eşecurilor pentru diferite dimensiuni de pagină.

Astfel, din figura 16 rezultă că aceeaşi proporţie a eşecurilor, realizată în cazul utilizării unor pagini cu dimensiunea de 1 K, poate fi obţinută în cazul utilizării unui spaţiu de memorie (alocare) considerabil mai mic şi prin folosirea unor pagini de dimensiune mai mică (conform punctelor a,b,c şi d pe curbele din figură).

./.

Un alt factor implicat în problema alegerii dimensiunii de pagină este consumul neproductiv de timp utilizat pentru gestiunea software a paginilor. Un asemenea consum se referă la:

(1) întreţinerea tabelelor descriptoare de segmente şi pagini, utilizate în cursul procedurii de translatare a adreselor;

(2) actualizarea informaţiilor privind prezenţa în memoria principală, utilizarea şi modificarea paginilor;

(3) gestiunea şi alocarea memoriei paginate interne (prin activarea rutinei pentru execuţia algoritmului de înlocuire) şi externe;

(4) detectarea unui defect de pagină, echivalent cu o cerere de introducere/extragere (I/E).

Ori de cîte ori apare un asemenea eveniment, supervizorul trebuie să întreprindă următoarele acţiuni:

- selectarea rutinei de supervizor ce tratează comunicaţiile de canal;

- plasarea unei intrări în şirul de aşteptare a cererilor de I/E pentru canalul selectat;

- memorarea stării programului ce generează defectul de pagină; aceasta echivalează cu memorarea conţinutului contorului de instrucţii, acumulatorului, registrelor aritmetice etc. într-un spaţiu rezervat al memoriei UCP;

- comutarea procesorului la o nouă lucrare;

(5) detectarea terminării transferului unei pagini, echivalent cu un serviciu I/E.

Se întreprind următoarele acţiuni:

- actualizarea şirului I/E al canalului respectiv;

- plasarea unei noi intrări în şirul de aşteptare al cererilor pentru utilizarea UCP;

Este dificil a determina acest consum neproductiv de timp implicat în gestiunea software a paginilor ca funcţie de dimensiunea paginii. In cazul utilizării paginilor de dimensiune mică există mai multe pagini de cercetat, dar condiţia pentru care se realizează această căutare poate fi distribuită în acelaşi manieră pentru paginile mari şi mici, astfel încît adîncimea de

./.

cercetare ar fi aceeaşi. Deoarece se pot formula condiţii ce ar favoriza oricare dimensiune, timpul software utilizat la înlocuirea unei pagini se presupune independent de dimensiunea paginii.

Cercetarea unor componente ale consumului neproductiv de timp, utilizat pentru realizarea hardware a operaţiilor de paginare (timpul I/E consumat pentru accesul şi transferul paginilor) şi pentru gestiunea software a paginilor este realizată în (150): în lucrare este descris un model ce include explicit şi acest consum, şi care se referă la un sistem cu memorie virtuală cu două nivele, operînd într-un mediu multiprogramat; nivelul de multiprogramare este fix. Consumul de timp asociat cu detectarea unei cereri I/E este presupus avînd o distribuţie exponenţială. Toate serviciile I/E şi consumul de timp asociat cu terminarea unei I/E sînt presupuse constante. Rezultatele expuse indică că consumul neproductiv de timp nu este în nici un caz neglijabil; valoarea lui variază nelinear, în funcţie de raportul timpilor medii de servire I/E şi UCP.

In cadrul unui exemplu consumul neproductiv de timp echivalează cu cca.13% din timpul total de rulare şi aprox. 12% din utilizarea totală a UCP.

Succesul sistemelor cache (cap.2 secţiunea 6.2.) indică, că principiul de localizare se aplică atît la scară microscopică, cît şi la scară macroscopică a sistemelor cu paginare. Acest succes este un argument în favoarea utilizării unor dimensiuni mici de pagină.

In cadrul modelelor 155 şi 165 ale sistemului IBM/370 dimensiunea blocurilor, utilizate pentru transferul între nivelele M_1 şi M_2 ale ierarhiei de memorii, este de 32 octeţi (8, 104).

Avantaje ale utilizării unor pagini de dimensiuni mici sînt discutate de asemenea în (18),(159); în (53) se constată îmbunătăţirile importante aduse în utilizarea memoriei prin folosirea unor pagini de dimensiuni mici, de exemplu 200 octeţi.

./.

Putem constata de asemenea, că beneficiul utilizării unor dimensiuni mari de pagină, legat de eficienţa operaţiei de transport a paginii, este redus drastic prin introducerea recentă a unor noi tehnologii de dispozitive de memorii (memorii CCD, bubble şi altele - vezi tabelul 1 din cap.2) cu timpi de acces rapizi.

Astfel introducerea acestor noi tehnologii în cadrul sistemelor utilizînd ierarhii de memorii constituie un argument în favoarea utilizării unor dimensiuni mici de pagină.

Pe baza consideraţiilor expuse se poate trage concluzia că folosirea unor pagini de dimensiuni mici conduce în sistemele moderne utilizînd MLVM, la îmbunătăţiri în utilizarea sistemului de memorii. Aceste îmbunătăţiri se referă la utilizarea mai eficientă a spaţiului de memorie şi la valoarea mică a consumului neproductiv de timp utilizat pentru accesul hardware şi gestiunea software a paginilor.

2. Anomalia dimensiunii de pagină

Anomalia dimensiunii de pagină se referă la raportul numărului de defecte de pagină generate în cadrul execuţiei unui program şi utilizării succesive a două dimensiuni de pagină.

Ea a fost observată în cadrul unor experienţe pentru cercetarea eficienţei reîmpachetării automate a programelor şi secţiunilor de programe în pagini de memorie virtuală, astfel ca să se obţină o reducere a numărului de defecte de pagină.S-a pus problema dacă îmbunătăţirile de performanţă (mai puţine defecte de pagină pentru o alocare fixă de memorie) obţinute prin această împachetare pentru o anumită dimensiune de pagină, s-ar aplica de asemenea în cazul utilizării unor pagini de dimensiuni duble sau jumătate, fără a mai proceda la o reîmpachetare suplimentară. S-a cercetat efectul utilizării unor pagini de dimensiune dublă sau jumătate avînd în vedere faptul că dimensiunile paginilor utilizate în sisteme reale sînt puteri ale lui 2.

Rezultatele obţinute în urma acestor studii au confirmat, în general, presupunerile cercetătorilor: anume, că şi în cazul utilizării unor pagini, avînd dimensiunile dublă şi jumătate faţă de dimensiunea de pagină folosită la aplicarea algo-

./.

ritmului de împachetare, se pot constata îmbunătăţiri ale performanţei. Insă, s-au obţinut rezultate surprinzătoare la compararea numărului de defecte de pagină generate în cazul utilizării dimensiunilor de pagină de p octeţi şi, corespunzător, $\frac{p}{2}$ octeţi, surprinzătoare întrucît ele contrazic atît intuiţia, cît şi considerentele expuse în secţiunea precedentă. Anume, dacă un program este împărţit în pagini de dimensiunea p şi apoi în pagini de dimensiunea $\frac{p}{2}$, execuţia programului avînd pagini de dimensiune mai mică ar părea să implice un transfer mai mic de date, rezultînd din defecte de pagină. Motivul pentru acest lucru este că nu ne-am aştepta ca programul să folosească întotdeauna ambele jumătăţi ale unei pagini de dimensiune mare.

Considerăm o primă situaţie, de extremă, în cadrul căreia ar fi întotdeauna folosită numai o jumătate din conţinutul paginilor de dimensiune mai mare. In această situaţie lungimea secvenţei paginilor, pentru ambele dimensiuni de pagină, ar fi aceeaşi; deoarece presupunem că unei cereri de pagină (i) de dimensiune mare, i-ar corespunde o cerere de pagină mică (2 i) sau (2 i - 1). Dacă această corespondenţă de la paginile mari la cele mici ar implica întotdeauna aceeaşi jumătate a unei pagini mari (i) şirul de referiri de pagină ar fi exact acelaşi (exceptînd renumerotarea) şi ar rezulta acelaşi număr de defecte de pagină, pentru o alocare de acelaşi număr de cadre de pagină. Dar acelaşi număr de cadre de pagină implică jumătate de spaţiu în cazul utilizării dimensiunii mici de pagină, aşa că pentru un spaţiu de aceeaşi capacitate, numărul de defecte de pagină ar fi considerabil mai mic. De asemenea, timpul pentru transferul paginilor ar fi mai mic, căci fiecare pagină este pe jumătate de lungă. Această situaţie favorizează în mod evident dimensiunea mai mică de pagină.

La cealaltă extremă (adică, cazul în care întotdeauna sînt utilizate ambele jumătăţi ale unei pagini de dimensiune mare), la fiecare referire a unei pagini (i) de dimensiune mare, secvenţa corespunzătoare a paginilor mici

../.

referite ar fi (2i-1), (2i). Sirul de referiri pentru paginile mici ar avea o lungime exact dublă, faţă de lungimea şirului de referiri de pagini mari; pentru aceeaşi capacitate de spaţiu de memorie reală, numărul de defecte de pagini mici ar fi exact dublul numărului de defecte de pagini mari. Dar din nou, deoarece paginile mici sînt pe jumătate de lungi, numărul total de octeţi transferaţi ar fi acelaşi. Consumul neproductiv de timp ar fi mai mare pentru paginile mici, datorită creşterii corespunzătoare a timpului de acces şi a timpului software asociat cu actualizarea şi căutarea în tabelele de pagini (creştere datorată dublării numărului de defecte de pagină).

Presupunem deci că limita superioară efectivă a numărului de defecte de pagini mici este dublul numărului de defecte de pagini mari; de asemenea, presupunem că, pe măsură ce densitatea de utilizare a memoriei în cadrul unei pagini scade, raportul între numărul de defecte de pagini mici şi numărul de defecte de pagini mari scade de asemenea. Ne aşteptăm ca avantajul utilizării unei dimensiuni mici de pagină să crească pe măsură ce spaţiul disponibil se micşorează.

Referindu-ne acum la rezultatele experimentale, ele au confirmat aceste presupuneri, în cazul unor programe caracterizate ca manifestînd o utilizare a memoriei de joasă densitate (o distribuţie în timp aprox. uniformă de utilizare a spaţiului de memorie reală alocat).

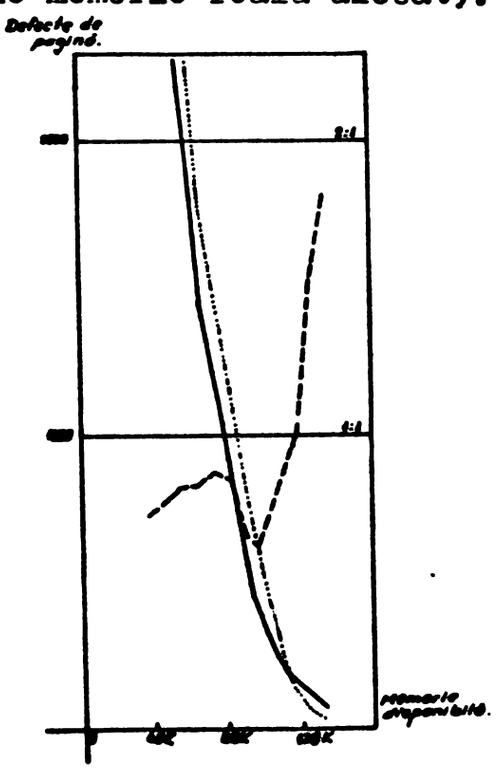


Fig.17.

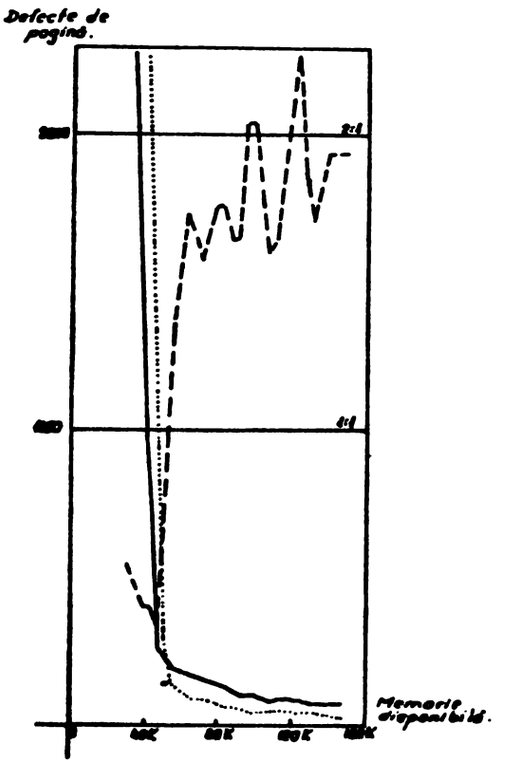


Fig.18.

In figura 17 sînt reprezentate curbele caracterizînd variaţia numărului de defecte de pagină în funcţie de alocarea de memorie (spaţiu de memorie reală disponibil) pentru pagini de 2 Ko (linie continuă) şi pagini de 4 Ko (linie întreruptă). Algoritmul de înlocuire utilizat a fost FIFO. Linia punctată reprezintă raportul între numărul de defecte de pagină pentru pagini de 2 Ko şi numărul de defecte de pagină pentru pagini de 4 Ko. Prin linii orizontale sînt reprezentate raportul de 1:1 şi raportul 2:1.

Din figura 17 rezultă că, pentru majoritatea valorilor capacităţii memoriei alocate, paginile de 2 Ko conduc la un număr mai mic de defecte de pagină decît paginile de 4 Ko. Numai în situaţia în care programului îi este alocat întreg spaţiul de memorie pe care îl necesită, raportul defectelor de pagină tinde spre valoarea 2:1. Înălţimea la care se opreşte curba reprezentînd acest raport indică raportul între numărul total de pagini de 2 Ko necesitate şi numărul total de pagini de 4 Ko necesitate.

In figura 18 sînt reprezentate aceleaşi curbe, dar pentru un program caracterizat ca manifestînd o localizare puternică a utilizării memoriei (programul îşi distribuie în timp referirile la spaţiul de memorie neuniform, localizat la anumite domenii). După cum se observă din figură, pe o porţiune considerabilă a domeniului de variaţie a memoriei disponibile şi anume în jumătatea dreaptă a domeniului de variaţie - regiunea de activitate de paginare moderată şi scăzută - utilizarea paginilor de 2 Ko conduce la un număr de defecte de pagină mai mare decît numărul de defecte de pagină rezultate prin utilizarea unor pagini de 4 Ko. Si - contrar aşteptărilor intuitive - raportul dintre numerele defectelor de pagină, rezultate prin utilizarea unor pagini de 2 Ko şi 4 Ko depăşeşte valoarea de 2:1, limita superioară presupusă. O asemenea situaţie este definită ca reprezentînd anomalia dimensiunii de pagină.

Prin cercetarea unor şiruri de referiri de pagină se pot identifica situaţii ce modelează cazuri de apariţie a

./.

unei asemenea anomalii.

Fie, în acest sens, un sistem ierarhic de memorii cu două nivele, în cadrul căruia utilizăm, succesiv, pagini cu dimensiunea de p octeţi şi $\frac{p}{2}$ octeţi. Fie e_p şi $e_{\frac{p}{2}}$ numărul defectelor de pagină (eşecuri) obţinute în cele două situaţii. Notăm prin ς raportul acestor două numere:

$$\varsigma = \frac{e_{\frac{p}{2}}}{e_p}$$

Valorile posibile ale acestui raport ς le repartizăm în trei domenii de interes:

(1) $\varsigma < 1$

(2) $1 \leq \varsigma \leq 2$

(3) $\varsigma > 2$

Vom discuta, pe rînd, aceste cazuri.

Cazul 1. $\varsigma < 1$

In acest caz, numărul de pagini aduse, prin defecte de pagină, în memoria principală în cazul utilizării paginii mai mici, este mai mic. De asemenea, ţinînd seama şi de faptul că transferul paginii de dimensiuni mai mici se realizează într-un timp mai scurt, rezultă că această situaţie ilustrează un caz de comportare de acces a programelor, ce favorizează utilizarea paginilor de dimensiuni mai mici.

In figura 19 este reprezentat un exemplu de asemenea caz. Similar cu secţiunea 4.1. din cap.2 am notat prin:

λ = lungimea şirului de referiri de pagină;

γ = grupul de pagini distincte în şir;

η = numărul de pagini distincte în grupul γ.

Definim m_i ca fiind dimensiunea lui M_i exprimată în unităţi de pagină, recepţionate de la nivelul inferior vecin. Astfel:

$$m_1 = \frac{c_1}{p_2}$$

La utilizarea unei asemenea dimensiuni de pagină de $\frac{p}{2}$ octeţi în loc de p octeţi, M_1 va conţine de două ori mai multe pagini, fiecare din aceste pagini fiind însă pe jumătate de mare:

./.

$$(m_1)_p = \frac{c_1}{p}$$

$$(m_1)_{\frac{p}{2}} = \frac{c_1}{\frac{p}{2}} = 2 \cdot \frac{c_1}{p}$$

deci:

$$(m_1)_{\frac{p}{2}} = 2 \cdot (m_1)_p$$

In convertirea şirului de referiri de pagină de dimensiuni p în şir de referiri de pagină de dimensiune $\frac{p}{2}$, pentru reprezentarea celor două jumătăţi ale unei pagini A de dimensiune p, se folosesc adresele logice de pagină A_1 şi A_2.

Parametrii sistemului, în cazul utilizării paginii p:

. x = A, B, C, A, B, C
. $\lambda = 6$
. $\Gamma = \{A, B, C\}$
. $\eta = 3$
. $m_1 = 2$
. algoritm înlocuire: FIFO

Parametrii sistemului în cazul utilizării paginii $\frac{p}{2}$:

. $x = A_1, B_1, C_1, A_1, B_1, C_1$
. $\lambda = 6$
. $\Gamma = \{A_1, B_1, C_1\}$
. $\eta = 3$
. $m_1 = 4$
. algoritm înlocuire: FIFO

Pagină p	Şir referiri		A	B	C	A	B	C
	Defecte pagină		*	*	*	*	*	*
	Conţinut M_1	1	A	B	C	A	B	C
Pagină $\frac{p}{2}$	Şir referiri		A_1	B_1	C_1	A_1	B_1	C_1
	Defecte pagină		*	*	*			
	Conţinut M_1	1	A_1	B_1	C_1	A_1	B_1	C_1
		2		A_1	C_1	A_1	B_1	C_1
		3			C_1	A_1	B_1	A_1
		4						

./.

Rezultate:

- $e_p = 6$
- $e_{\frac{p}{2}} = 3$
- $\varsigma = \frac{3}{6} = 0,5$

Fig.19. Exemplu pentru cazul 1.

In exemplul din fig.19 raportul $\varsigma = 0,5$ ceea ce înseamnă că numărul de pagini aduse prin defecte de pagină a fost redus la jumătate, prin folosirea dimensiunii de pagină mai mică. Rezultatul se referă evident la acea situaţie de extremă, discutată mai sus, în cadrul căreia programul foloseşte doar prima jumătate din conţinutul paginilor de dimensiune mai mare. Un asemenea tip de rezultat poate fi aşteptat de la un program ce manifestă o comportare de referiri destul de răzleaţă şi nelocalizată. Astfel, în cadrul unui sistem tipic cu paginare cu două nivele, este adusă în memoria principală o pagină cu dimensiunea de 4096 octeţi la prima referire pe care o face programul la această pagină. Dacă în intervalul de timp următor programul nu face alte referiri la această pagină, s-a creat o situaţie în care s-au adus în memoria principală multe date, dar au rămas nefolosite. In aceste circumstanţe utilizarea lui M_1 ar putea fi îmbunătăţită prin păstrarea unei colecţii mai mari şi mai diversificate de pagini, deşi fiecare pagină ar fi mai mică.

Cazul 2. $1 \leqslant \varsigma \leqslant 2$

Acesta este un domeniu de tranziţie. Pentru $\varsigma = 1$ utilizarea paginilor mai mici va fi mai avantajoasă, deoarece numărul de defecte de pagină fiind acelaşi în ambele cazuri, totuşi timpul consumat pentru transferul paginilor mai mici va fi mai scurt. Pentru $\varsigma = 2$ utilizarea paginilor mai mici va conduce la un număr dublu de defecte de pagină. Dublarea acestui număr conduce la creşterea corespunzătoare a timpului consumat pentru accesul hardware şi gestiunea software a acestor pagini. De aici rezultă comportarea mai bună, în această situaţie, a paginilor de dimensiune mai mare.

./.

Punctul de tranziţie poate fi determinat pe baza următoarelor considerente. Timpul de tratare a unui defect de pagină este compus din timpul consumat pentru transferul datelor, timpul de acces al dispozitivului de memorii, unde rezidă pagina respectivă şi timpul software implicat în tratarea întreruperii generate de un defect de pagină şi determinarea paginii ce urmează a fi înlocuită (algoritmul de înlocuire). Presupunînd că performanţa dispozitivului de memorii şi procedura generală de paginare sînt identice pentru ambele dimensiuni de pagină, timpul de acces t_a va fi acelaşi pentru ambele pagini. Timpul de transfer este t_t pentru pagina de dimensiune mare şi $\frac{1}{2} \cdot t_t$ pentru pagina de dimensiune pe jumătate. In ceea ce priveşte consumul de timp software t_s implicat în înlocuirea unei pagini, conform celor discutate în secţiunea 1.3 a acestui capitol, se presupune acelaşi în ambele cazuri.

In fig. 20 este reprezentat un exemplu al acestui caz pentru care valoarea raportului $\varsigma = 2,0$ deci se observă dublarea numărului de defecte de pagină în cadrul utilizării unei dimensiuni mai mici de pagină. Un asemenea tip de rezultat ar putea fi aşteptat de la programe ce manifestă o comportare în mostra de referiri la memorie densă, localizată şi secvenţială.

Parametrii sistemului, în cazul utilizării paginii p:

- $x = A,A,B,B,C,C$
- $\lambda = 6$
- $\int = \{A,B,C\}$
- $\eta = 3$
- $m_1 = 2$
- algoritm înlocuire FIFO

Parametrii sistemului, în cazul utilizării paginii $\frac{p}{2}$:

- $x = A_1,A_2,B_1,B_2,C_1,C_2$
- $\lambda = 6$
- $\int = \{A_1,A_2,B_1,B_2,C_1,C_2\}$
- $\eta = 6$
- $m_1 = 4$
- algoritm înlocuire FIFO

./.

Pagina p	Sir referiri	A	A	B	B	C	C
	Defecte pagină	*		*		*	
	Conţinut M_1 1	A	A	B	B	C	C
	2			A	A	B	B

Pagina $\frac{p}{2}$	Sir referiri	A_1	A_2	B_1	B_2	C_1	C_2
	Defecte pagină	*	*	*	*	*	*
	Conţinut M_1 1	A_1	A_2	B_1	B_2	C_1	C_2
	2		A_1	A_2	B_1	B_2	C_1
	3			A_1	A_2	B_1	B_2
	4				A_1	A_2	B_1

Rezultate:

- $e_p = 3$
- $e_{\frac{p}{2}} = 6$
- $\varsigma = \frac{6}{3} = 2,0$

Fig. 20. Exemplu pentru cazul 2.

Intuitiv, rezultatul obţinut pentru raportul $\varsigma = 2,0$ este cazul cel mai defavorabil, căci în cazul acesta sîntem nevoiţi să aducem întotdeauna în memoria principală ambele jumătăţi A_1 şi A_2 ale fiecărei pagini originale mari A, prin aceasta pierzînd toate beneficiile utilizării unei dimensiuni mai mici de pagină $\frac{p}{2}$ şi cauzînd un număr dublu de defecte de pagină. Insă această observaţie bazată pe intuiţie nu e corectă: cazul $\varsigma = 2,0$ nu este cel mai defavorabil.

Cazul 3. $\varsigma > 2$

In acest al treilea domeniu, utilizarea paginilor de $\frac{p}{2}$ octeţi ar conduce la apariţia unui număr mai mare decît dublu de defecte de pagină decît în cazul utilizării paginilor de p octeţi şi, ca urmare a acestui lucru, este defavorizantă.

Parametrii sistemului, în cazul paginii p:

- x = A,B,A,B,C,C,B,A,A,C,C
- $\lambda = 11$

./.

- $\Gamma = \{A, B, C\}$
- $\eta = 3$
- $m_1 = 2$
- algoritm de înlocuire FIFO

Parametrii sistemului, în cazul utilizării paginii $\frac{p}{2}$

- $x = A_1, B_1, A_2, B_2, C_1, C_2, B_1, A_1, A_2, C_1, C_2$
- $\lambda = 11$
- $\Gamma = \{A_1, A_2, B_1, B_2, C_1, C_2\}$
- $\eta = 6$
- $m_1 = 4$
- algoritm de înlocuire FIFO

Pagina p	Sir referiri		A	B	A	B	C	C	B	A	A	C	C
	Defecte pagină		*	*			*			*			
	Conținut M_1	1	A	B	B	B	C	C	C	A	A	A	C
		2		A	A	A	B	B	B	C	C	C	B
Pagina $\frac{p}{2}$	Sir referiri		A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1	C_2
	Defecte pagină		*	*	*	*	*	*	-	*	*	*	*
	Conținut M_1	1	A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1	C_2
		2		A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1
		3			A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2
		4				A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1

Rezultate:

- $e_p = 4$
- $c_{\frac{p}{2}} = 11$
- $\varrho = \frac{11}{4} = 2,75$

Fig.21. Exemplu pentru cazul 3.

In fig.21 se ilustrează un model de referiri la memorie ce conduce la un raport al numărului de defecte de pagină de 2,75. Alte asemenea exemple de șiruri de referiri de pagini, ce conduc la $\varrho > 2$, utilizînd ca algoritmi de înlocuire atît FIFO, cît și LRU, sînt date în (88).

./.

3. Consideraţii finale. Necesitatea unor studii suplimentare

Anomalia dimensiunii de pagină a fost detectată în cadrul unor studii legate de posibilitatea obţinerii unor îmbunătăţiri suplimentare ale performanţei unui sistem ierarhic de memorii prin utilizarea unor dimensiuni mai mici de pagină. Anume, contrar aşteptărilor, numărul de defecte de pagină, apărute în cazul utilizării unei dimensiuni de pagină de $\frac{p}{2}$ octeţi depăşeşte valoarea dublă a numărului de defecte de pagină, apărute în cadrul execuţiei programului şi utilizării unei dimensiuni de pagină de p octeţi.

Anomalia descrisă în (88) a fost constatată pentru valori ale dimensiunii de pagină cuprinse între p = 2 Ko şi p = 16 Ko. Insă din considerarea exemplelor date în secţiunea precedentă, exemple ce modelează cazuri de apariţie a anomaliei de pagină, se poate constata că apariţia anomaliei este legată de mostra şirului de referiri de pagină şi nu de dimensiunea paginii. Ne putem deci aştepta la o astfel de anomalie şi pentru dimensiuni mici ale paginii (de 64 şi 32 octeţi). Rezultă că anomalia dimensiunii de pagină poate fi întîlnită atît la calculatoarele ce folosesc hardware de relocare, pentru translatarea adreselor, cît şi la cele ce utilizează memorii locale (cache).

Anomalia dimensiunii de pagină afectează toţi algoritmii de înlocuire utilizaţi în prezent. Există o excepţie şi ea se referă la algoritmul MIN, descris de Belady (20), care conduce la numărul minim de defecte de pagină, pentru orice şir de referiri. Se poate uşor arăta (88) că acest algoritm nu poate produce un număr mai mare decît dublu de defecte de pagină, pentru o dimensiune de pagină redusă la jumătate. Interesul pentru acest algoritm este de natură teoretică numai, căci implementarea practică a acestui algoritm este greu de realizat. Anume, algoritmul necesită cunoaşterea exactă a întregului şir de referiri la memorii, generat la execuţia lui, înainte de execuţie. Ori realizarea acestei condiţii este practic imposibilă pentru un program, căci acesta poate interacţiona dinamic cu mediul (intrări şi ieşiri variabile în timp şi ca volum) şi cu resursele sistemului (utilizarea unor resurse software şi hardware ale sistemului

./.

într-un mod greu de precizat apriori, mai ales într-un mediu multiprogramat sau cu diviziune în timp).

In ceea ce priveşte interesul prezentat de algoritmul MIN, acesta constă în următoarele:

După cum s-a arătat în (18), fiind date un program caracterizat de şirul lui de referiri de pagină şi dimensiunea memorici reale alocate, este util să se cunoască numărul minim de defecte de pagină necesare pentru a rula programul, cu scopul de a evalua diferite configuraţii de memorii şi scheme de gestiune a paginilor. De exemplu, eficienţa unui algoritm de înlocuire a paginilor este definită ca raportul dintre numărul minim de defecte de pagină şi numărul de defecte de pagină generate în cadrul utilizării unui anumit algoritm. Algoritmul MIN este o metodă pentru a determina numărul minim de defecte de pagină. Anume, algoritmul descris în (18) tratează şirul de referiri generat de program şi, pentru o anume dimensiune fixă a memoriei reale alocate, determină numărul minim asociat de defecte de pagină prin construcţia, însă numai după o întîrziere necesară şi variabilă, a stărilor memoriei şi tranziţiilor lor.

Algoritmul MIN cu mai multe valori descris în (20) permite obţinerea în mod practic simultan a numerelor minime de defecte de pagină pentru întregul domeniu de alocări posibile de memorie reală, eliminîndu-se astfel dezavantajul algoritmului MIN, descris în (18) şi care, la un moment dat, avea ca obiect o singură dimensiune de memorie. Lucrarea (20) prezintă, de asemenea, o posibilă implementare hardware a algoritmului MIN cu mai multe valori şi anume un dispozitiv care, ataşat unui calculator cu memorie virtuală, produce în mod continuu la ieşire un şir de valori ale capacităţilor de memorie minimă (MMC), în timp ce calculatorul execută programul respectiv. O valoare p a MMC are următorul sens: înainte de a fi referit, elementul şirului de referiri de pagină asociat cu această ieşire era conţinut în memorii de capacitate p sau mai mare şi de aceea, în cazul referirii la memorii de capacitate mai mică decît p, produce un defect în pagină. Pe baza şirului de valori a MMC şi a alocării

./.

reale se poate determina numărul minim de defecte de pagină.

In final, anomalia dimensiunii de pagină a fost constatată pentru programe manifestînd un caracter localizat de utilizare al memoriei şi anume în regiunea de activitate scăzută de paginare.

Aceasta este starea prezentă în cunoaşterea acestei probleme. Anomalia dimensiunii de pagină a fost descrisă, dar nu cunoaştem nici o lucrare dedicată studiului acestei probleme şi clarificării factorilor implicaţi în ea. Ca atare, lipseşte o înţelegere mai precisă a mecanismului de apariţie.

In aceste condiţii rezultă necesitatea unor studii suplimentare care să cerceteze mai îndeaproape această problemă.

Importanţa unor asemenea studii rezultă din următoarele considerente. Pe baza cunoaşterii mecanismului de apariţie al acestei anomalii, se vor putea elabora metode sau procedee de prevenire a apariţiei. Acest lucru se va dovedi de interes practic imediat. El va permite obţinerea unor îmbunătăţiri suplimentare în gestiunea unui sistem ierarhic de memorii prin utilizarea unor dimensiuni reduse de pagină, fără riscul unui consum mare de timp neproductiv datorat activităţii exagerate de paginare.

In încheiere cîteva consideremte privind posibilitatea apariţiei anomaliei dimensiunii de pagină în cazul unei programări structurate. S-a arătat că această anomalie a dimensiunii de pagină s-a detectat în cazul utilizării, pentru un program structurat pentru o anumită dimensiune de pagină p, a unei pagini de dimensiune înjumătăţită, $\frac{p}{2}$. Pentru a beneficia de avantajele utilizării unei dimensiuni reduse de pagină şi a nu mai recurge la o nouă restructurare a programului pentru o asemenea nouă dimensiune, este posibil să se ruleze programul structurat iniţial în noile condiţii, dar adoptînd măsuri de evitare a apariţiei anomaliei. Evident este posibilă şi soluţia de a recurge la o nouă restructurare a programului pentru situaţia utilizării noii dimensiuni de pagină. Alegerea uneia din aceste soluţii este dependentă de facilităţile hardware şi software ale sistemului concret considerat.

CAP.4. MODELE ANALITICE ALE COMPORTĂRII DE ACCES A UNUI PROGRAM

1. Modele existente.

Modelele analitice de bază ale comportării de acces a unui program studiate pînă în prezent, sînt funcţia de timp de existenţă (17) şi modelul de grup de lucru (52). Le vom prezenta în continuare.

1.1. Funcţia de timp de existenţă.

In cadrul sistemelor ce utilizează o gestiune dinamică la cerere a memoriei, este dispusă în memoria de execuţie doar o fracţiune dintr-un program. Apoi, pe măsură ce sînt necesitate proceduri sau date suplimentare, acestea sînt încărcate în memorie dinamic. Această schemă creează o secvenţă alternantă de intervale de execuţie (e) şi întîrzieri (u).

Intr-un sistem dat, la un anumit moment în timpul execuţiei unui anumit program, în domeniul de memorie alocat acestui program există o mulţime particulară de unităţi informaţionale ce sînt asociate programului. Volumul şi conţinutul acestui domeniu de memorie alocat determină timpul de execuţie ce va fi realizat pînă la apariţia necesităţii unor informaţii suplimentare (defect de pagină). In lucrarea (17), Belady şi Kuehner fac presupunerea că lungimea medie a intervalelor de execuţie (e) este funcţie doar de mărimea spaţiului de memorie ocupat.

Funcţia de timp de existenţă, introdusă în lucrarea amintită, stabileşte o relaţie între lungimea medie a intervalelor de execuţie (e) şi dimensiunea memoriei (s), care este alocată programului pentru desfăşurare.

Factori ca modul de programare şi tipul problemei, întîlnite în cadrul unor programe concrete, ca şi algoritmul de înlocuire utilizat de sistem, influenţează valorile locale (e), dar nu sînt de natură să modifice comportamentul general al funcţiei de timp de existenţă. Asupra acestui comportament se pot face unele consideraţii intuitive. Astfel, dacă alocarea de memorie este mică, fracţiunea de program dispusă în memorie va realiza intervale de execuţie foarte scurte. Dacă însă alocarea permite dispunerea întregului program în memorie, intervalul de execuţie mediu este de ordinul timpului total de

./.

rulare. Ar fi de aşteptat ca această capacitate de execuţie să fie monoton crescătoare între aceste extreme.

Ipoteza cea mai simplă relativă la modul în care sînt distribuite referirile la memorie ale unui program, este că aceste referiri urmează o distribuţie aleatoare.

Notăm cu:

r - domeniul de memorie, exprimat în pagini, referit de către program;

s - alocarea de memorie, exprimată de asemenea în pagini ($s < r$)

Atunci poate fi arătat că numărul aşteptat de referiri consecutive la memorie, înaintea apariţiei unui nou defect de pagină, este:

$$e = \frac{q}{1 - q} = q + q^2 + q^3 + \ldots$$

unde:

$$q = \frac{s}{r}$$

e reprezintă deci lungimea medie a unui interval de execuţie, între două defecte succesive de pagină, exprimată în unităţi de referiri la memorie a unui program, manifestînd o distribuţie aleatoare a referirilor la memorie.

Funcţia de mai sus poate fi reprezentată printr-o curbă convexă, ca în fig.22.

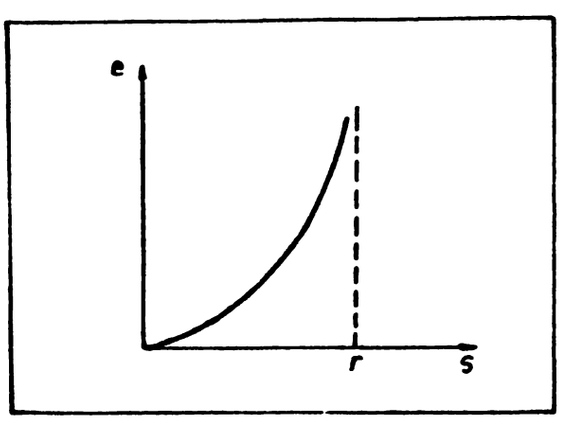


Fig.22. Funcţia de timp de existenţă pentru accese aleatoare.

Referirile la memorie ale programelor reale nu urmează însă o distribuţie aleatoare. Forma mai generală a unei funcţii de timp de existenţă mai realistă este reprezentată în fig.23 cu linie

./.

continuă.

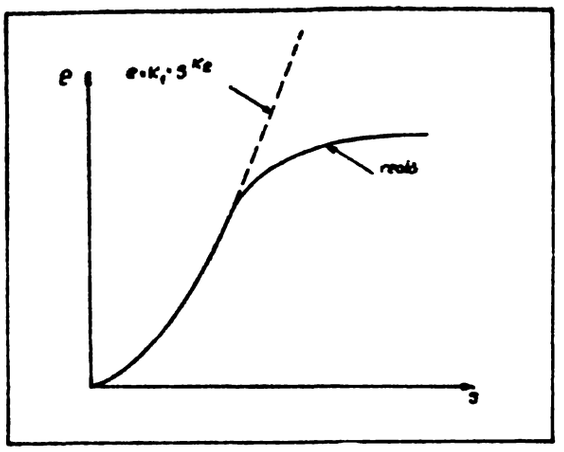


Fig.23. Funcţia generală de timp de existenţă.

Unele observaţii au arătat că pentru multe programe, relaţia între e şi s este neliniară şi că o porţiune a ei poate fi aproximată prin:

$$e = k_1 \cdot s^{k_2}$$

Relaţia de mai sus este ea însăşi o aproximaţie, iar determinarea valorilor pentru constantele k_1 şi k_2, astfel încît să se obţină o corespondenţă satisfăcătoare cu un anumit program real, este o a doua aproximaţie. In fig.23 este de asemenea reprezentată - prin linie punctată - aproximarea propusă în (17).

In acest model efectul dimensiunii paginii nu este considerat.

1.2. Modelul de grup de lucru.

Acest model este bazat pe recunoaşterea unei proprietăţi importante a comportării programelor şi anume localizarea. Localizarea este proprietatea ca, în decursul oricărui interval de execuţie, programul va favoriza unele din paginile lui într-o măsură mai mare decît pe altele. Factorii ce justifică această proprietate sînt discutaţi în secţiunea ce se referă la analiza comportamentului localizat al unui program (secţiunea 2 a cap.5).

Grupul de lucru de informaţii $W_p(t,\tau)$ a procesului p la timpul t este definit în (52) a fi colecţia de informaţii pe care procesul p le-a referit în decursul intervalului de timp

./.

virtual $(t - \tau, t)$. Ideea este ilustrată în fig.24.

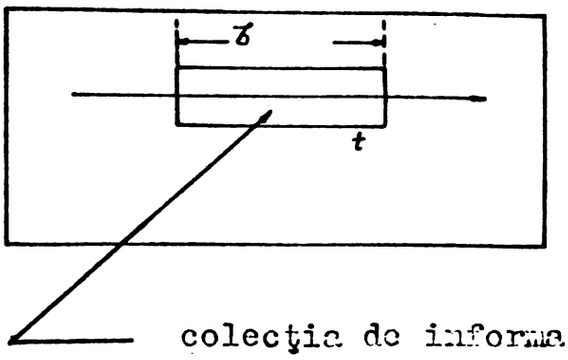


Fig.24. Definirea grupului de lucru.

Astfel, informaţiile referite de proces în timpul ultimilor τ unităţi de timp virtual ale execuţiei lui alcătuiesc grupul lui de lucru. τ este numit parametrul grupului de lucru.

Considerînd elementele lui $W(t, \tau)$ ca fiind pagini, se defineşte dimensiunea grupului de lucru $\omega(t, \tau)$ ca fiind numărul de pagini în $W(t, \tau)$. Presupunînd că dimensiunea grupului de lucru $\omega(t, \tau)$ este un proces stochastic staţionar, rezultă că aşteptarea matematică $\overline{\omega(t, \tau)}$ este independentă de t şi deci se poate scrie:

$$w(\tau) = \overline{\omega(t, \tau)}$$

Presupunînd unităţi de timp discret, Denning şi Schwartz (56) dau următoarea expresie pentru dimensiunea medie a grupului de lucru:

$$w(\tau) = \lim_{k \to \infty} \left[\frac{1}{k} \sum_{t=1}^{t=k} \omega(t, \tau) \right]$$

A fost arătat în (39,56) că dimensiunea medie a grupului de lucru a unui program este distribuită normal, lucru ce simplifică calculul algebric.

In (52) este arătat că valoarea aşteptată a dimensiunii grupului de lucru are următoarele proprietăţi:

(1) $w(\tau) \leq \tau$

Deoarece numărul maxim de referiri distincte ce pot apărea în τ unităţi de timp virtual este τ, avem $\omega(t, \tau) \leq \tau$ şi de aceea $w(\tau) \leq \tau$

./.

(2) $w(0) = 0$

Acest lucru este evident, căci nici o pagină nu poate fi referită la timpul zero.

(3) $w(\tau + \lambda) \geqslant w(\tau)$; $\lambda \geqslant 0$

Această proprietate ce arată că $w(\tau)$ este o funcţie nedescrescătoare, rezultă din faptul că în intervale de timp mai lungi pot fi referite un număr mai mare de pagini.

(4) $w(\tau)$ e convexă.

In fig.25 a fost reprezentată forma curbelor $w(\tau)$ pentru două tipuri de program.

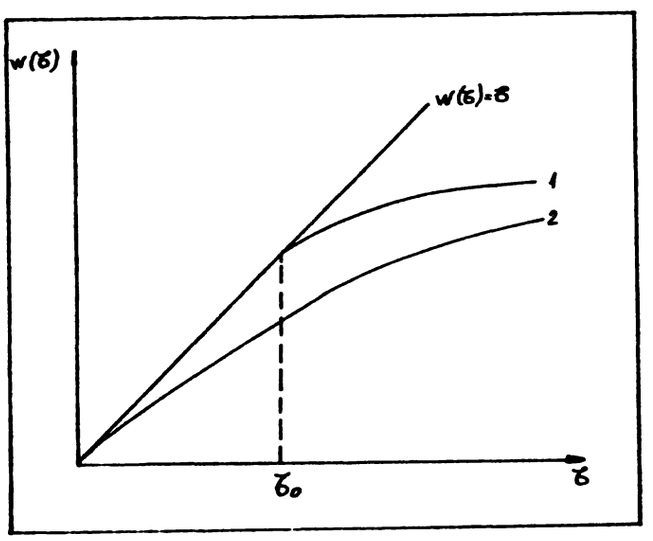


Fig.25. Dimensiunea aşteptată a grupului de lucru.

Programul de tipul 1 are un grup bine definit de pagini favorizate; el are tendinţa de a-şi distribui majoritatea referirilor lui uniform asupra unui grup de τ_0 pagini favorizate. Pentru intervale $\tau < \tau_0$ pentru un asemenea program ne putem aştepta ca programul să facă în mod frecvent referiri la pagini distincte noi, iar $w(\tau)$ să crească aproape linear cu τ. Pentru $\tau \gg \tau_0$ grupul de lucru va conţine multe pagini nefolositoare. Pentru un program cu o asemenea comportare de acces, alegerea lui τ aproape de valoarea τ_0 nu va afecta eficienţa operării, dar va diminua volumul de memorie alocat programului.

Programul de tipul 2 are un grup vag definit de pagini favorizate.

Fie x o variabilă aleatoare ce caracterizează intervalele de timp virtual între referirile succesive la aceeaşi

./.

pagină;

Fie: $F_x(u) = Pr\ [x \leq u]$

funcţia ei de distribuţie şi

$$f_x(u) = \frac{dF_x(u)}{du}$$

funcţia ei de densitate. In (41) se arată că acest tip de distribuţie a intervalelor de interreferiri se poate modela printr-o distribuţie hiperexponenţială. Fie numărul de referiri la memorie (intervalul de execuţie în u.t.v.). Atunci:

$$g(y_i = \tau) = \alpha_i(1 - \alpha_i)^{\tau - 1} \qquad i = 1,2,\ldots,n$$

unde:

$g(y_i = \tau)$ = probabilitatea că intervalul de interreferire al paginii i este egal cu τ;

α_i = probabilitatea de referire a paginii i ;

n = numărul total de pagini referite de un program.

Probabilitatea unui defect de pagină (sau a unei pagini lipsă) $\lambda(\alpha)$ este definită ca probabilitatea ca un proces să facă următoarea lui referire la o pagină ce nu este conţinută în grupul de lucru $W(t, \tau)$. Probabilitatea ca pagina referită să nu fie cuprinsă în $W(t,\tau)$ este de fapt probabilitatea ca intervalul de interreferire cel mai recent să satisfacă $x > \tau$, astfel că:

$$\lambda(\tau) = Pr\ [x > \tau] = 1 - F_x(\tau)$$

O pagină i care este referită numai o dată va avea intervalul ei de interreferire de pagină egal cu infinit.

Dimensiunea grupului de lucru la timpul t constă din paginile a căror intervale de interreferire sînt mai mici sau egale cu τ, o valoare finită; de aceea această determinare a dimensiunii grupului de lucru nu reuşeşte să includă paginile referite o singură dată. De asemenea, modelul de grup de lucru nu consideră efectul variaţiei dimensiunii paginii. Dimensiunile paginilor au un rol definit în determinarea ratelor de paginare şi utilizare memoriei.

După cum s-a arătat în cap.2 secţiunea 3.4. strategia de grup de lucru poate fi utilizată ca algoritm de înlocuire şi anume: într-un mediu multiprogramat, numai acele programe sînt eligibile de a putea fi rulate, a căror grup de lucru nu depăşeşte spaţiul propriu disponibil în memoria principală. A fost

./.

remarcată deficienţa legată de utilizarea unui asemenea algoritm şi care constă în dificultatea de a măsura dinamic parametrul de grup de lucru, cu scopul de a estima dimensiunea grupului de lucru.

Conceptul de grup de lucru este frecvent utilizat în studiile de alocare a resurselor. Coffman şi Ryan (39) au comparat efectul partiţionării fixe şi dinamice a memoriei, bazaţi pe cerinţele de grup de lucru ale unui program; este presupus că dimensiunea grupului de lucru urmează o distribuţie normală. Rodriguez şi Dupuy (141) prezintă proiectarea, implementarea şi evaluarea unui dispecer pentru grupuri de lucru. Ferrari (67) cercetează posibilitatea îmbunătăţirii comportării localizate a programelor prin utilizarea grupurilor de lucru critice. Ghanem (81) propune un algoritm pentru partiţia memoriei principale între N programe competitive, cu caracteristici diferite, algoritm bazat pe o strategie optimă de alocare, obţinută prin utilizarea conceptului de grup de lucru. Bryant (28) analizează rezultate empirice a unor dimensiuni ale grupurilor de lucru şi ajunge la concluzia că ipoteza privind distribuţia normală a dimensiunilor grupurilor de lucru nu este justificată.

1.3. Modele stochastice.

Vom descrie cîteva modele stochastice, ce se referă la reprezentarea analitică a şirului de referiri de pagină a unui program (vezi cap.2 secţiunea 3.3.1.).

1.3.1. Model cu probabilităţi constante.

Fiind date cele N pagini ale unui program, numerotate de la 1 la N, modelul de comportament cel mai simplu constă în a ataşa fiecăreia din aceste pagini o probabilitate de referire p_i. Evident:

$$\sum_{i=1}^{N} p_i = 1$$

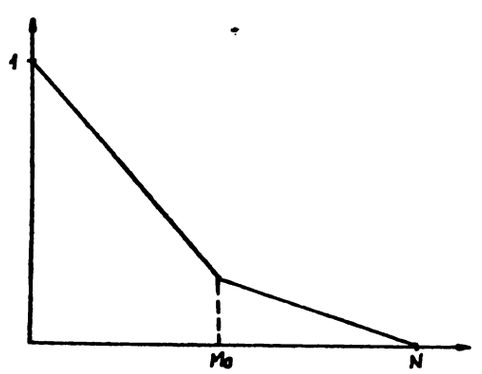
p_i determină probabilitatea pentru ca pagina i să facă obiectul celei de a k-a referiri. Această probabilitate este constantă în timp, deci independentă de k.

./.

Un asemenea model poate să fie utilizat pentru a reprezenta într-un mod aproximativ fenomenul de scădere a performanţei în cazul unei alocări date de memorie; pentru aceasta este suficient să se aleagă un subansamblu de M pagini, fiecăreia din care i se atribuie o probabilitate p, restul paginilor primind o probabilitate q

$$Mp + (N - m)\, q = 1$$

Executînd programul într-un mediu cu o anumită alocare de memorie, se obţine pentru reprezentarea probabilităţii unui defect de pagină curba din fig.26, care corespunde suficient de precis rezultatelor raportate în (26).



Probabilitatea unui defect de pagină (în regim staţionar)

dimensiunea memoriei alocate

parachor **dimensiunea programului**

Fig.26. Asupra probabilităţii unui defect de pagină.

Dimensiunea m_0 a memoriei alocate, la care se constată o schimbare clară în modul de variaţie a curbei corespunde dimensiunii de alocare a memoriei, definite de Belady (19) drept **parachor: Acesta din urmă reprezintă volumul de informaţii care trebuie să fie rezident în memoria principală, astfel încît programul să cheltuiască nu mai mult de jumătate din timpul de execuţie în aşteptări de pagină.**

Cele m_0 pagini a căror probabilitate de referire este $Mp > 0{,}5$ reprezintă un "nucleu" al programului, al cărui parametru $\frac{p_m}{1 - p_m}$ poate fi pus în legătură cu modul în care programul îşi distribuie referirile: valori mari ale acestui raport caracterizează un program cu referiri localizate; valori apropiate de unitate corespund unui program ce-şi distribuie în mod uniform referirile.

./.

In cadrul acestui model, programul este caracterizat prin două numere fără dimensiune: $\frac{m}{M}$ raportul dintre dimensiunea nucleului şi dimensiunea totală a programului şi de raportul $\frac{p_{mm}}{1-p_{mm}}$

1.3.2. Model markovian staţionar.

In cazul acestui model se consideră constante probabilităţile de tranziţie p_{ij} între pagini:

$$p_{ij} = Pr(r_{k+1} = j \mid r_k = i) \qquad i,j \in [1,M]$$

şi

$$\sum_{j=1}^{M} p_{ij} = 1 \qquad i \in [1,M]$$

Matricea de tranziţie p_{ij} o notăm cu P.

Vom arăta cîteva proprietăţi ale acestui model.

Presupunem că toate paginile programului fac obiectul unei referiri într-un interval de timp suficient şi că nu există "stare de absorbire" (adică $p_{ii} \neq 1$). Aceste două condiţii se traduc prin ipoteza că lanţul Markov reprezentînd comportarea de acces a programului este regulat. Probabilitatea de referire se exprimă prin:

$$Pr(r_{k+1}=i) = \sum_{j=1}^{n} p_{ij}^{(k)} \, b_j$$

unde $|b_i|$ este vectorul probabilităţilor iniţiale (b_i = probabilitatea ca prima referire să fie făcută la pagina i), iar $p_{ij}^{(k)}$ reprezintă elementul (i,j) al matricii P^k. Comportarea puterilor succesive ale matricii de tranziţie joacă un rol determinant în acest model. Asupra acestei comportări s-a arătat că pentru un lanţ Markov regulat, puterile succesive P^n a matricii de tranziţie converg spre o matrice limită A, a cărei linii sînt identice cu un vector a. Vectorul a este soluţia ecuaţiei.

$$aP = a$$

şi este un vector al probabilităţilor ($\sum a_i = 1$) care condiţionează comportarea asimptotică a lanţului Markov, în sensul că:

./.

$$\lim_{k \to \infty} Pr(r_k = i) = a_i$$

Utilizarea acestui model este justificată în cazul unor studii precise privind comportarea de acces a programelor sau în cazul unui număr redus de tranziţii.

Considerînd referirea la pagini a unui program un proces în puncte, în (111) se propune un model semi-Markov cu două stări ce descrie apariţia defectelor de paginã. Modelul este definit pe baza şirului de valori $\{x_i\}$ ce caracterizează intervalele între defectele de paginã. Se presupune cã existã două tipuri de asemenea intervale, cu funcţiile de densitate probabilisticã $p_1(x)$ şi $p_2(x)$. Modelul postulează cã existã un lanţ Markov cu două stări, cu matricea de tranziţie:

$$P = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix} = \begin{pmatrix} \alpha_1 & 1-\alpha_1 \\ 1-\alpha_2 & \alpha_2 \end{pmatrix}$$

astfel încît fiind dat cã X_{j-1} are f.d.p. $p_1(x)$, probabilitatea ca X_j sã aibã f.d.p. $p_2(x)$ este $1-\alpha_1$; probabilitatea ca X_j sã aibã f.d.p. $p_1(x)$ este α_1, independent de tipul sau lungimea intervalelor anterioare.

Pentru f.d.p. a primului tip de intervale se propune distribuţia geometricã $p_1(x)$ avînd media μ_1:

$$p_1(x) = p_1^{x-1}(1 - p_1) \qquad (0 < p_1 < 1;\ x = 1,2,\ldots)$$

$$\mu_1 = \frac{1}{1-p_1}$$

Pentru f.d.p. a celui de al doilea tip de intervale se propune distribuţia negativ-binominalã:

$$p_2(x,k) = \binom{k + x - 2}{x - 1} p_2^{x-1} (1 - p_2)^k$$

$$(0 < p_2 < 1; \quad k > 0; \quad x = 1,2,\ldots)$$

Modelul impune estimarea unui număr de 6 parametri.

O extensie a acestui model pentru o ierarhie cu 3 nivele este datã în (76).

1.4. Critica modelelor prezentate.

Modelele prezentate nu pot fi folosite ca instrument de cercetare în analiza anomaliei dimensiunii de paginã, cãci ele

./.

nu consideră efectul variaţiei dimensiunii de pagină. In plus, mai pot fi semnalate şi alte deficienţe ale acestor modele.

Astfel, în cadrul modelului bazat pe funcţia de timp de existenţă:

- parametrii modelului k_1 şi k_2 nu sînt în mod clar definiţi pe baza proprietăţilor programului care este modelat;

- modelul este valid pentru programe reale numai pe o porţiune a domeniului de variaţie a lui s (alocarea de memorie).

In privinţa modelului bazat pe grup de lucru se poate constata:

- dificultatea de a măsura dinamic parametrul de grup de lucru (cu scopul de a estima dimensiunea grupului de lucru);

- modul cum este definită dimensiunea grupului de lucru nu permite includerea în această dimensiune a paginilor care sînt referite o singură dată.

In cazul modelelor stochastice, cele ce presupun probabilităţi constante de referire a paginilor sau probabilităţi constante de tranziţie între pagini impun un grad de simplificare prea mare al comportării de acces a unui program, iar modelele ce utilizează proprietăţile unui proces semi-Markov conduc la necesitatea estimării unui număr relativ mare de parametri, care nu pot fi puşi în legătură directă cu proprietăţile programului modelat.

In această situaţie, pentru analiza anomaliei dimensiunii de pagină apare **necesitatea elaborării unui instrument de cercetare corespunzător, un model al comportării de acces a unui program construit în aşa manieră, încît să ia în consideraţie şi efectul variaţiei dimensiunii de pagină, iar parametrii modelului să fie în mod clar definiţi pe baza proprietăţilor programului care este modelat.**

2. <u>Modelul propus al comportării de acces a programului</u>.

In cele ce urmează vom construi un model analitic, ce presupune următoarea comportare de acces a programului:

./.

(1) selectarea aleatoare a unei locaţii "a" în spaţiul de adrese virtuale al programului;

(2) selectarea aleatoare a unui ciclu cu lungimea α;

(3) accesul la α locaţii în ciclul (2) de mai sus secvenţial de σ ori;

(4) repetarea paşilor de mai sus.

Un ciclu ce se extinde în afara ultimei adrese virtuale disponibile programului este continuat la începutul spaţiului de adrese virtuale. Deci, în conformitate cu cele de mai sus, comportarea de acces a programului constă în apelarea unei serii de cicluri, poziţia şi dimensiunea cărora se schimbă în mod aleator.

Vom fi interesaţi de numărul defectelor de pagină generate de un program cu o asemenea comportare de acces.

Fie:

P = dimensiunea programului (în cuvinte sau octeţi);

p = dimensiunea paginii;

$\lceil P/p \rceil$ = dimensiunea aparentă a programului considerată de un sistem cu dimensiunea de pagină p.

Notaţia $\lceil x \rceil$ reprezintă cel mai **mic** întreg, care este mai mare sau egal cu x.

2.1. Efectele lungimii ciclului de program asupra numărului defectelor de pagină.

Primul lucru pe care îl determinăm este numărul de pagini **distincte referite în cadrul accesului la un ciclu. Acest număr este funcţie de lungimea ciclului şi de poziţia ciclului în spaţiul de adrese virtuale.**

Spaţiul de adrese virtuale poate fi reprezentat ca în fig. 27, în care linia A-B descrie un segment al spaţiului de adrese virtuale; liniile de diviziune sînt limite de pagini, iar o limită de pagină este presupusă făcînd parte din pagina din partea dreaptă.

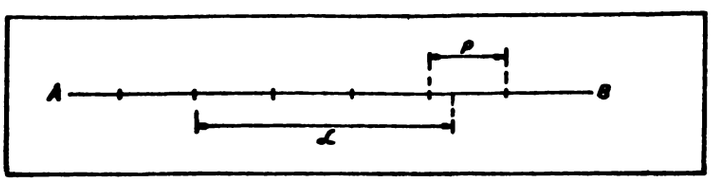


Fig.27. Reprezentarea spaţiului de adrese virtuale.

Considerăm un ciclu cu lungimea α ce începe la o adresă limită de pagină. Numărul de pagini distincte referite în cadrul

./.

ciclului este $\left\lceil \frac{\alpha}{p} \right\rceil$. Din fig.27 rezultă că, mutînd poziţia ciclului spre dreapta, numărul de pagini distincte referite va rămîne $\left\lceil \frac{\alpha}{p} \right\rceil$ pînă în momentul în care capătul din dreapta al ciclului traversează o limită de pagină. In acel moment numărul de pagini referite va deveni $\left\lceil \frac{\alpha}{p} \right\rceil + 1$ şi rămîne la această valoare pînă în momentul în care capătul din stînga al ciclului traversează o limită de pagină. Astfel, numărul de pagini distincte referite de un ciclu cu lungimea α va fi $\left\lceil \frac{\alpha}{p} \right\rceil$ sau $\left\lceil \frac{\alpha}{p} \right\rceil + 1$ în funcţie de poziţia ciclului în spaţiul adreselor virtuale.

Vrem să determinăm poziţiile ciclului ce conduc la referirea a $\left\lceil \frac{\alpha}{p} \right\rceil$ şi respectiv $\left\lceil \frac{\alpha}{p} \right\rceil + 1$ pagini distincte. Considerăm poziţia ciclului în spaţiul adreselor virtuale determinată de adresa de început a ciclului sau marginea din stînga pe fig.28. Vom arăta că, în cazul poziţionării adresei de început a ciclului la oricare din primele $\left\lceil \frac{\alpha}{p} \right\rceil . p - \alpha$ adrese, vor fi referite în ciclu un număr de $\left\lceil \frac{\alpha}{p} \right\rceil$ pagini distincte.

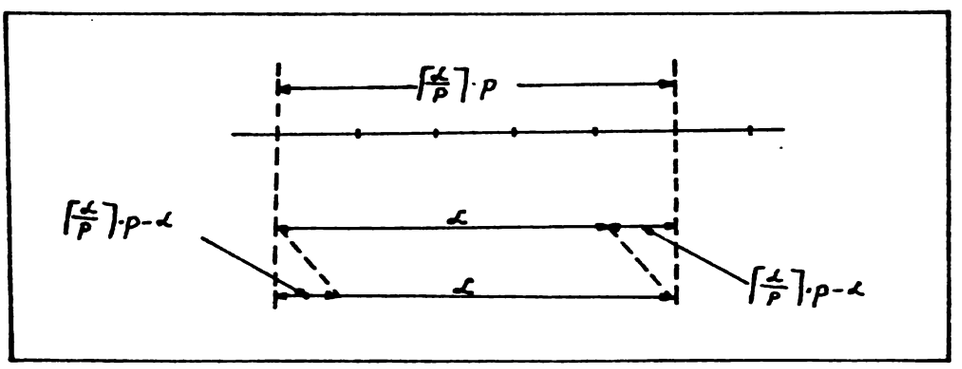


Fig.28. Poziţia ciclului în spaţiul de adrese virtuale.

Dimensiunea aparentă a ciclului, considerată de sistem, este de $\left\lceil \frac{\alpha}{p} \right\rceil . p$ cuvinte: în cadrul acestei dimensiuni rămîn neutilizate, în execuţia ciclului, un număr de $\left\lceil \frac{\alpha}{p} \right\rceil p - \alpha$ adrese. Astfel se poate poziţiona ciclul la oricare din primele $\left\lceil \frac{\alpha}{p} \right\rceil . p - \alpha$ adrese a oricărei pagini şi se vor referi numai $\left\lceil \frac{\alpha}{p} \right\rceil$ pagini distincte. Corespunzător, dacă ciclul este poziţionat la oricare din $p - (\left\lceil \frac{\alpha}{p} \right\rceil . p - \alpha)$ sau

./.

$p - \left\lceil \frac{\alpha}{p} \right\rceil \cdot p + \alpha$ adrese a oricărei pagini, ciclul va referi un număr de $\left\lceil \frac{\alpha}{p} \right\rceil + 1$ pagini distincte.

Considerăm poziţia "a" a fiecărui ciclu ca fiind o variabilă aleatoare uniform distribuită, al cărui domeniu este întreg spaţiul de adrese virtuale al sistemului. Astfel, pentru probabilităţile de referire de către un ciclu a unui număr de pagini N distincte avem următoarele expresii:

$$Pr\ (\text{referire } \left\lceil \frac{\alpha}{p} \right\rceil \text{pagini distincte}) = \left\lceil \frac{\alpha}{p} \right\rceil - \frac{\alpha}{p} \quad ; \qquad (1)$$

$$Pr\ (\text{referire } \left\lceil \frac{\alpha}{p} \right\rceil + 1 \text{ pagini distincte}) = 1 - \left\lceil \frac{\alpha}{p} \right\rceil + \frac{\alpha}{p} \qquad (2)$$

2.2. Consideraţii preliminare privind numărul de defecte de pagină.

Sîntem interesaţi de numărul de defecte de pagină generate în decursul accesului la un asemenea ciclu. Există doi factori importanţi ce trebuie consideraţi. Primul trebuie determinat dacă numărul de pagini distincte referite de ciclu este mai mare decît spaţiul alocat $\frac{s}{p}$. Dacă avem o asemenea situaţie, vom avea de-a face cu o paginare aproape continuă. Această activitate va fi consecinţa faptului că, în timpul accesului secvenţial la un asemenea ciclu, paginile ce compun ciclul vor trebui înlocuite din memoria principală înainte de a se completa ciclul şi a fi referite din nou. In termenii modelului de grup de lucru, într-un asemenea caz se constată că grupul de lucru al ciclului este mai mare decît spaţiul alocat.

In cazul în care spaţiul alocat $\frac{s}{p}$ este mai mare decît sau egal cu numărul de pagini unice referite de ciclu, vom constata o activitate intensă de paginare pe timpul cît paginile ce compun ciclul sînt aduse în memoria principală, urmată de o perioadă de execuţie, neîntreruptă de defecte de pagină suplimentare.

Al doilea factor ce trebuie avut în vedere este acela al probabilităţii de a găsi deja aduse în memoria principală (sau alt echivalent similar în ierarhie) paginile necesitate pentru execuţia ciclului, ca rezultat al activităţii de paginare ce a avut loc mai devreme. Aceasta înseamnă că, în momentul în care începem fiecare procedură nouă de acces la un

./.

nou ciclu, există probabilitatea de a găsi unele pagini necesitate de ciclu deja rezidente în memoria principală.

Pentru a determina numărul de defecte de pagină DP ce apar în timpul accesului la un nou ciclu cu lungimea de α cuvinte de σ ori va fi necesar să considerăm două cazuri distincte:

$DP(\alpha) = DP_1(\alpha)$ pentru cazul în care adresa a de început a ciclului este situată la primele $\left\lceil \frac{\alpha}{p} \right\rceil . p - \alpha$ adrese ale unei pagini; (3)

$= DP_2(\alpha)$ pentru cazul în care adresa a de început a ciclului este situată în ultimile $p - \left\lceil \frac{\alpha}{p} \right\rceil . p + \alpha$ adrese ale unei pagini; (4)

Necesitatea celor două cazuri este determinată de numărul diferit de pagini distincte referite în fiecare caz şi anume, în primul caz sînt referite $\left\lceil \frac{\alpha}{p} \right\rceil$ pagini distincte, iar în al doilea caz sînt referite $\left\lceil \frac{\alpha}{p} \right\rceil + 1$ pagini distincte.

Expresia pentru $DP_2(\alpha)$ va putea fi obţinută din expresia pentru $DP_1(\alpha)$, ţinînd seama de această diferenţă privind numărul paginilor distincte referite.

In ceea ce priveşte $DP_1(\alpha)$ putem constata că ea însăşi se poate găsi în două situaţii şi anume:

$$DP_1(\alpha) = DP_{1a}(\alpha) \quad \text{pentru} \quad \left\lceil \frac{\alpha}{p} \right\rceil \leq \frac{s}{p} \; ; \qquad (5)$$

$$= DP_{1b}(\alpha) \quad \text{pentru} \quad \left\lceil \frac{\alpha}{p} \right\rceil > \frac{s}{p} \; ; \qquad (6)$$

In cazul (1a) numărul de pagini distincte referite este mai mic sau egal cu numărul de pagini alocate în memorie. Astfel, după referirea iniţială a tuturor paginilor ce alcătuiesc ciclul, nu vom mai avea nici o altă activitate de paginare. Este evident că vor exista, cel mult $\left\lceil \frac{\alpha}{p} \right\rceil$ defecte de pagină în decursul accesului la acest ciclu, căci după ce a avut loc primul acces la acest ciclu, toate paginile cerute se vor găsi în memorie şi următoarele accese la acest ciclu nu vor produce alte defecte de pagină.

In cazul (1b), în cadrul ciclului sînt referite mai multe pagini distincte decît spaţiul alocat în memorie. In această situaţie, presupunînd LRU, FIFO sau alt algoritm similar de înlocuire a paginilor (vezi cap.2 secţiunea 3.4),

./.

vom constata că după referirea primelor $\frac{s}{p}$ pagini distincte din cadrul ciclului, va apărea un defect de pagină de fiecare dată ce secvenţa referirilor traversează o limită de pagină. Aceasta va continua atîta timp cît are loc accesul la ciclu. Deoarece accesul la acest ciclu va avea loc de σ ori, vor apărea un număr maxim de $\left\lceil \frac{\alpha}{p} \right\rceil . \sigma$ defecte de pagină.

2.3. Efectele paginării anterioare.

Un alt factor ce trebuie considerat este acela al găsirii în memorie a uneia sau a mai multor din paginile necesitate $\left\lceil \frac{\alpha}{p} \right\rceil$ distincte. Cînd începe accesul la ciclul considerat în memorie se găsesc $\frac{s}{p}$ pagini, ca rezultat al execuţiei unuia sau mai multor cicluri de acces precedente. Există posibilitatea ca între aceste pagini să se afle şi unele din cele necesitate de noul ciclu şi ca atare pot rămîne în memorie (să nu fie înlocuite).

Este evident că numărul defectelor de pagină evitate în acest mod este o variabilă aleatoare. Se poate de asemenea presupune că media şi funcţia de densitate a acestei variabile aleatoare sînt funcţii de toţi **a** şi α precedenţi. Vom evita aceste complicaţii, făcînd cîteva ipoteze şi înlocuind această variabilă aleatoare cu media ei. Astfel, aproximăm probabilitatea ca pagina i a unui ciclu curent să se afle deja dispusă în memorie, prin expresia:

$$\Pr(\text{pagina } i \text{ în memorie}) = \frac{\frac{s}{p} - i + 1}{\left\lceil \frac{P}{p} \right\rceil} \quad \text{pentru } s < p \cdot \left\lceil \frac{P}{p} \right\rceil \qquad (7)$$

$$= 1 \quad \text{pentru } s \geqslant p \cdot \left\lceil \frac{P}{p} \right\rceil \qquad (8)$$

Sînt cîteva motive pentru care această aproximaţie poate fi considerată corespunzătoare. In primul rînd, ea este suficient de precisă pentru valori mici ale lui i, iar aceste valori mici ale lui i se întîlnesc mai frecvent. Valoarea lui i nu poate depăşi $\frac{s}{p}$ sau $\left\lceil \frac{\alpha}{p} \right\rceil + 1$. Aceasta înseamnă că, pentru a putea implica valori mari ale lui i, trebuie să avem o alocare s mare şi o lungime de ciclu α mare. In al doilea rînd, vom utiliza o însumare a acestor probabilităţi pentru **i** = 1 pînă la **i** = $\left\lceil \frac{\alpha}{p} \right\rceil$ sau în unele cazuri i = $\left\lceil \frac{\alpha}{p} \right\rceil + 1$. Valorile acestor termeni în relaţia de aproximare propusă scad cu creşterea lui i. Astfel, termenii dominanţi în serie sînt acei cu i mici, pentru care apro-

./.

mineţia este mai precisă.

Putem scrie valoarea aşteptată a numărului acestor pagini, ce se găsesc deja în memorie, pentru fiecare nou ciclu, sub următoarea formă:

$$\sum_{i=1}^{k} \frac{\frac{s}{p} - i + 1}{\left\lceil \frac{P}{p} \right\rceil} = \frac{k\left(\frac{s}{p} + 1\right) - \sum_{i=1}^{k} i}{\left\lceil \frac{P}{p} \right\rceil} = \frac{k \cdot \frac{s}{p} + k - \frac{k(k+1)}{2}}{\left\lceil \frac{P}{p} \right\rceil}$$

pentru $s < p\left\lceil \frac{P}{p} \right\rceil$ (9)

şi

$$\sum_{i=1}^{k} 1 = k \qquad \text{pentru } s \geq p\left\lceil \frac{P}{p} \right\rceil \qquad (10)$$

unde:

$k = \left\lceil \frac{\alpha}{p} \right\rceil$ în cazul 1a

şi

$k = \frac{s}{p}$ în cazul 1b

Valorile lui k provin din următoarele considerente:

În cazul (1a) numărul de pagini referite este mai mic sau egal cu numărul alocat de pagini. În acest caz trebuie să terminăm numerea la numărul de pagini distincte referite, care este $\left\lceil \frac{\alpha}{p} \right\rceil$. În cazul (1b) numărul de pagini distincte referite este mai mare decît numărul alocat de pagini. Numărul alocat de pagini este $\frac{s}{p}$ şi ne putem aştepta să găsim pagini, ca rezultat al paginării anterioare, numai în timpul referirii primelor $\frac{s}{p}$ pagini distincte.

Făcînd înlocuirile, se obţin următoarele expresii:

Cazul 1 a.

$$\left\lceil \frac{\alpha}{p} \right\rceil \leq \frac{s}{p} \qquad \text{şi} \qquad k = \left\lceil \frac{\alpha}{p} \right\rceil$$

subcazul $s < p\left\lceil \frac{P}{p} \right\rceil$

$$\sum_{i=1}^{\left\lceil \frac{\alpha}{p} \right\rceil} \frac{\frac{s}{p} - i + 1}{\left\lceil \frac{P}{p} \right\rceil} = \frac{\left\lceil \frac{\alpha}{p} \right\rceil \cdot \frac{s}{p} + \left\lceil \frac{\alpha}{p} \right\rceil - \frac{1}{2} \cdot \left\lceil \frac{\alpha}{p} \right\rceil^2 - \frac{1}{2}\left\lceil \frac{\alpha}{p} \right\rceil}{\left\lceil \frac{P}{p} \right\rceil} =$$

$$= \frac{\left\lceil \frac{\alpha}{p} \right\rceil \left(\frac{s}{p} - \frac{1}{2}\left\lceil \frac{\alpha}{p} \right\rceil + \frac{1}{2}\right)}{\left\lceil \frac{P}{p} \right\rceil}; \qquad (11)$$

./.

Subcazul $s \geqslant p\left\lceil \frac{P}{p} \right\rceil$

$$\sum_{i=1}^{\left\lceil \frac{\alpha}{p} \right\rceil} 1 = \left\lceil \frac{\alpha}{p} \right\rceil \qquad (12)$$

Cazul 1 b.

$$\left\lceil \frac{\alpha}{p} \right\rceil > \frac{s}{p}$$

Ținînd seama și de relația evidentă:

$$\left\lceil \frac{\alpha}{p} \right\rceil \leqslant \left\lceil \frac{P}{p} \right\rceil$$

rezultă că:

$$s < p \cdot \left\lceil \frac{\alpha}{p} \right\rceil \leqslant p \cdot \left\lceil \frac{P}{p} \right\rceil$$

Deci, în cazul 1 b nu sînt posibile două subcazuri, ci doar unul singur:

$$s < p \cdot \left\lceil \frac{P}{p} \right\rceil$$

$$\sum_{i=1}^{\frac{s}{p}} \frac{\frac{s}{p} - i + 1}{\left\lceil \frac{P}{p} \right\rceil} = \frac{\frac{s}{p}\left(\frac{s}{p} + 1\right) - \frac{s}{p}\left(\frac{s}{p} + 1\right) \cdot \frac{1}{2}}{\left\lceil \frac{P}{p} \right\rceil} = \frac{1}{2\left\lceil \frac{P}{p} \right\rceil}\left(\frac{s^2}{p^2} + \frac{s}{p}\right) \qquad (13)$$

Scăzînd numărul mediu al paginilor ce se găsesc deja în memorie din numărul defectelor posibile de pagină, se obține valoarea reală a numărului defectelor de pagină.

Cazul 1 a.

Subcazul: $s < p\left\lceil \frac{P}{p} \right\rceil$

$$DP_{1a} = \left\lceil \frac{\alpha}{p} \right\rceil - \frac{\left\lceil \frac{\alpha}{p} \right\rceil\left(\frac{s}{p} - \frac{1}{2}\left(\left\lceil \frac{\alpha}{p} \right\rceil\right)\right) + \frac{1}{2}\left\lceil \frac{\alpha}{p} \right\rceil}{\left\lceil \frac{P}{p} \right\rceil} =$$

$$= \left\lceil \frac{\alpha}{p} \right\rceil\left(1 - \frac{s}{\left\lceil \frac{P}{p} \right\rceil} - \frac{1}{2\left\lceil \frac{P}{p} \right\rceil} + \frac{1}{2\left\lceil \frac{P}{p} \right\rceil} \cdot \left\lceil \frac{\alpha}{p} \right\rceil\right) \qquad (14)$$

Subcazul: $s \geqslant p\left\lceil \frac{1}{p} \right\rceil$

$$DP_{1a} = \left\lceil \frac{\alpha}{p} \right\rceil - \left\lceil \frac{\alpha}{p} \right\rceil = 0 \qquad (15)$$

./.

Cazul 1 b.

$$s < p \left\lceil \frac{p}{p} \right\rceil$$

$$DP_{1b} = \left\lceil \frac{\alpha}{p} \right\rceil - \frac{1}{2\left\lceil \frac{p}{p} \right\rceil} \left(\frac{s^2}{p^2} + \frac{s}{p} \right) \qquad (16)$$

Expresiile pentru DP_{2a} şi DP_{2b} se obţin direct din expresiile DP_{1a} şi DP_{1b} prin înlocuirea lui $\left\lceil \frac{\alpha}{p} \right\rceil$ prin $\left\lceil \frac{\alpha}{p} \right\rceil + 1$

Astfel se obţine:

$$DP_{2a} = \left(\left\lceil \frac{\alpha}{p} \right\rceil + 1 \right) \left[1 - \frac{s}{p\left\lceil \frac{p}{p} \right\rceil} - \frac{1}{2\left\lceil \frac{p}{p} \right\rceil} + \frac{1}{2\left\lceil \frac{p}{p} \right\rceil} \left(\left\lceil \frac{\alpha}{p} \right\rceil + 1 \right) \right] \qquad (17)$$

$$\text{pentru } \left\lceil \frac{\alpha}{p} \right\rceil + 1 \leq \frac{s}{p} \quad \text{şi} \quad s < p \left\lceil \frac{p}{p} \right\rceil$$

$$DP_{2b} = \left(\left\lceil \frac{\alpha}{p} \right\rceil + 1 \right) - \frac{1}{2\left\lceil \frac{p}{p} \right\rceil} \left(\frac{s^2}{p^2} + \frac{s}{p} \right) \qquad (18)$$

$$\text{pentru } \left\lceil \frac{\alpha}{p} \right\rceil + 1 > \frac{s}{p}$$

2.4. Considerarea repartiţiei lungimii ciclurilor.

Pentru a calcula valoarea medie a defectelor de pagină trebuie cunoscută funcţia de densitate de repartiţie pentru α, lungimea unui ciclu.

Se poate considera că un program real s-ar putea modela printr-un număr de cicluri de lungimi fixe la care s-ar adăuga cîteva cicluri de lungimi variabile. Un ciclu de lungime fixă este format de exemplu în cazul unui ciclu DO real. Ciclurile de lungimi variabile sînt generate în cadrul unor activităţi, precum căutarea într-o listă a unei anumite valori. Aceste consideraţii ar conduce la o funcţie de densitate, conţinînd un număr de "vîrfuri". Nu urmărim însă în cadrul acestui model realizarea unei descrieri atît de detaliate a programului.

Putem considera însă o funcţie de densitate de repartiţie cu un singur maximum dominant, cu parametrii corespunzători, care permit controlul asupra alurii curbei şi poziţiei maximum-ului. O asemenea flexibilitate privind controlul asupra alurii curbei şi poziţiei maximum-ului este oferită de utilizarea repartiţiei de tip Γ. Parametrii aces-

./.

tei repartiţii trebuie aleşi astfel încît concordanţa cu repartiţia reală a lungimii ciclurilor să fie satisfăcătoare, îndeosebi pe intervalul ce prezintă interes $[1,P]$.

Pentru funcţia de repartiţie de tip Γ densitatea de repartiţie este determinată prin:

$$f(u) = \begin{cases} \frac{\varsigma^{\beta}}{\Gamma(\beta)} u^{\beta-1} e^{-\varsigma u} & \text{dacă } u > 0 ; \\ 0 & \text{dacă } u \leq 0 ; \end{cases} \qquad \varsigma > 0; \ \beta > 0$$

unde: $\Gamma(\beta)$ este funcţia Γ definită ca

$$\Gamma(\beta) = \int_0^{\infty} x^{\beta-1} e^{-x} dx$$

iar $\Gamma(\beta+1) = \beta\Gamma(\beta)$ pentru orice β.

O curbă ce indică modul de variaţie al funcţiei de densitate pentru repartiţia de tipul Γ având parametrul $\beta > 1$ este reprezentată în fig.29.

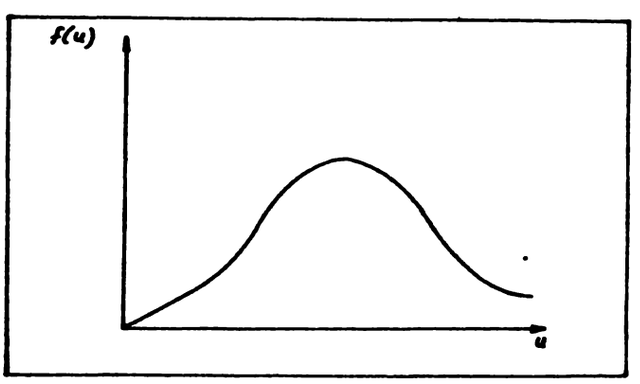


Fig.29. Funcţia de densitate pentru repartiţia de tipul Γ având $\beta > 1$

Cei doi parametri ς şi β ai repartiţiei de tip Γ îi vom determina, egalînd valorile medii şi dispersiile funcţiilor de repartiţie reală (r) şi aproximantă (a), considerate pe intervalul $[0, P]$.

Astfel:

$$M_r[u] = M_a[u] = \int_0^P u f(u) du \qquad (19)$$

$$M_r[u^2] = M_a[u^2] = \int_0^P u^2 f(u) du \qquad (20)$$

şi

$$D_r^2[u] = D_a^2[u] = M_r[u^2] - M_r^2[u] = M_a[u^2] - M_a^2[u] \qquad (21)$$

./.

Pe baza valorilor reale ale lungimilor ciclurilor programului modelat, valorile pentru $M_r[u]$ şi $M_r[u^2]$ pot fi determinate numeric. Pe de altă parte, pentru repartiţia aproximantă de tipul Γ valorile $M_a[u]$ şi $M_a[u^2]$ pot fi exprimate ca funcţii de parametri ς şi β

$$M_a[u] = \int_0^P u \cdot \frac{1}{\Gamma(\beta)} \varsigma^\beta u^{\beta-1} e^{-\varsigma u} du = \frac{1}{\varsigma \Gamma(\beta)} \int_0^P (\varsigma u)^\beta e^{-\varsigma u} d(\varsigma u) =$$

$$= \frac{1}{\varsigma \Gamma(\beta)} \int_0^{\varsigma P} v^\beta e^{-v} dv = \frac{\beta}{\varsigma} \frac{1}{\Gamma(\beta+1)} \int_0^{\varsigma P} v^{(\beta+1)-1} e^{-v} dv =$$

$$= \frac{\beta}{\varsigma} I[\varsigma P ; \beta+1] \qquad (22)$$

$$M_a[u^2] = \int_0^P u^2 \cdot \frac{1}{\Gamma(\beta)} \varsigma^\beta u^{\beta-1} e^{-\varsigma u} du = \frac{1}{\varsigma^2 \Gamma(\beta)} \int_0^{\varsigma P} v^{(\beta+2)-1} e^{-v} dv =$$

$$= \frac{(\beta+1)\beta}{\varsigma^2 \Gamma(\beta+2)} \int_0^{\varsigma P} v^{(\beta+2)-1} e^{-v} dv = \frac{(\beta+1)\beta}{\varsigma^2} I[\varsigma P, \beta+2] \qquad (23)$$

unde:

$$I[x,z] = \frac{1}{\Gamma(z)} \int_0^x u^{z-1} e^{-u} du$$

este funcţia Γ incompletă, tabulată.

Dacă valoarea P este suficient de mare, atunci $I[\varsigma P; \beta+1]$ şi $I[\varsigma P; \beta+2]$ pot fi considerate egale cu unitatea. Atunci se poate considera că:

$$M_a[u] \approx \frac{\beta}{\varsigma} ; \qquad (24)$$

$$M_a[u^2] \approx \frac{\beta(\beta+1)}{\varsigma^2} , \qquad (25)$$

$$D_a^2[u] \approx \frac{\beta}{\varsigma^2} . \qquad (26)$$

De aici se obţin următoarele estimări pentru ς şi β :

$$\varsigma = \frac{M_a[u]}{D_a^2[u]} = \frac{M_r[u]}{D_r^2[u]} ; \qquad (27)$$

şi

$$\beta = \varsigma M_a[u] = \frac{M_a^2[u]}{D_a^2[u]} = \frac{M_r^2[u]}{D_r^2[u]} \qquad (28)$$

./.

In final, repartiţia de tipul Γ având parametrii ρ şi β determinaţi conform relaţiilor (27) şi (28) este utilizată pentru aproximarea repartiţiei lungimii ciclurilor programului în intervalul $[1, P]$.

2.5. Elaborarea modelului final.

Valoarea medie a numărului de defecte de pagină DP (α) poate fi exprimată ca:

$$M[DP(\alpha)] = \int_0^{p\left\lceil \frac{P}{p} \right\rceil} \left[DP_1(\alpha)\ P(\text{Caz } 1|\alpha).f(\alpha) + DP_2(\alpha)\ P(\text{Caz } 2|\alpha).f(\alpha) \right] d\alpha \quad (29)$$

Pentru a ţine seama de schimbările ce apar în reprezentarea funcţiilor $DP_1(\alpha)$ şi $DP_2(\alpha)$ la punctele $\alpha = s - p$ şi $\alpha = s$ este necesară împărţirea integralei (29) în trei integrale separate.

$$M[DP(\alpha)] = \int_0^{s-p} \left[DP_{1a}(\alpha) P(\text{Caz } 1|\alpha).f(\alpha) + DP_{2a}(\alpha)\ P(\text{Caz } 2|\alpha).f(\alpha) \right] d\alpha +$$

$$+ \int_{s-p}^{s} \left[DP_{1a}(\alpha)\ P(\text{Caz } 1|\alpha).f(\alpha) + DP_{2b}(\alpha)\ P(\text{Caz } 2|\alpha).f(\alpha) \right] +$$

$$+ \int_{s}^{p\left\lceil \frac{P}{p} \right\rceil} \left[DP_{1b}(\alpha)\ P(\text{Caz } 1|\alpha).f(\alpha) + DP_{2b}(\alpha)\ P(\text{Caz } 2|\alpha).f(\alpha) \right] \quad (30)$$

Probabilităţile $P(\text{Caz } 1|\alpha)$ şi $P(\text{Caz } 2|\alpha)$ pot fi luate din (1) şi (2). Astfel:

$$P(\text{Caz } 1|\alpha) = \left\lceil \frac{\alpha}{p} \right\rceil - \frac{\alpha}{p} ; \quad (31)$$

$$P(\text{Caz } 2|\alpha) = 1 - \left\lceil \frac{\alpha}{p} \right\rceil + \frac{\alpha}{p} \quad (32)$$

Examinăm în continuare cele trei integrale.

In prima integrală dăm factor comun $f(\alpha)$, obţinînd:

$$I_1 = \int_0^{s-p} \left[DP_{1a}(\alpha)\ P(\text{Caz } 1|\alpha) + DP_{2a}(\alpha)\ P(\text{Caz } 2|\alpha) \right] f(\alpha)\, d\alpha \quad (33)$$

DP_{1a} este definită prin:

$$DP_{1a}(\alpha) = \left\lceil \frac{\alpha}{p} \right\rceil \left(1 - \frac{s}{p\left\lceil \frac{1}{p} \right\rceil} - \frac{1}{2\left\lceil \frac{P}{p} \right\rceil} + \frac{1}{2\left\lceil \frac{P}{p} \right\rceil} \cdot \left\lceil \frac{\alpha}{p} \right\rceil \right) \quad (34)$$

./.

Se poate observa că între valorile lui α în care $\left\lceil \frac{\alpha}{p} \right\rceil$ se schimbă brusc, funcţia $DP_{1a}(\alpha)$ are valori constante, deci reprezentarea grafică a acestei funcţii constă într-o serie de segmente paralele cu axa α, iar punctele de discontinuitate (aflate la valori ale lui α ce sînt multiplii întregi de p) sînt situate pe curba:

$$F_{11}(\alpha) = \frac{\alpha}{p}\left[1 - \frac{s}{p\left\lceil \frac{T}{p} \right\rceil} - \frac{1}{2\left\lceil \frac{T}{p} \right\rceil} + \frac{1}{2\left\lceil \frac{T}{p} \right\rceil} \cdot \frac{\alpha}{p}\right]$$

Forma lui $DP_{2a}(\alpha)$ este aceeaşi cu a funcţiei $DP_{1a}(\alpha)$, cu excepţia faptului că $\left\lceil \frac{\alpha}{p} \right\rceil$ este înlocuit cu $\left\lceil \frac{\alpha}{p} \right\rceil + 1$. Deci consideraţiile privind reprezentarea funcţiei $DP_{1a}(\alpha)$ sînt aplicabile şi pentru $DP_{2a}(\alpha)$, punctele de discontinuitate fiind situate pe curba:

$$F_{12}(\alpha) = \left(\frac{\alpha}{p} + 1\right)\left[1 - \frac{s}{p\left\lceil \frac{T}{p} \right\rceil} - \frac{1}{2\left\lceil \frac{T}{p} \right\rceil} + \frac{1}{2\left\lceil \frac{T}{p} \right\rceil}\left(\frac{\alpha}{p} + 1\right)\right]$$

In fig.30 sînt reprezentate grafic funcţiile:

$DP_{1a}(\alpha)$ şi $DP_{2a}(\alpha)$

Vom examina expresia:

$$DP_{1a}(\alpha)\ P(\text{Caz } 1|\alpha) + DP_{2a}(\alpha)\ P(\text{Caz } 2\,|\alpha) \qquad (35)$$

pe un domeniu de variaţie a lui α cuprins între np şi (n+1)p.

In fig.31 sînt trasate 3 curbe: în primă $DP_{1a}(\alpha)$ şi $DP_{2a}(\alpha)$, iar în următoarele două $P(\text{Caz } 1|\alpha)$ şi $P(\text{Caz } 2\,|\alpha)$.

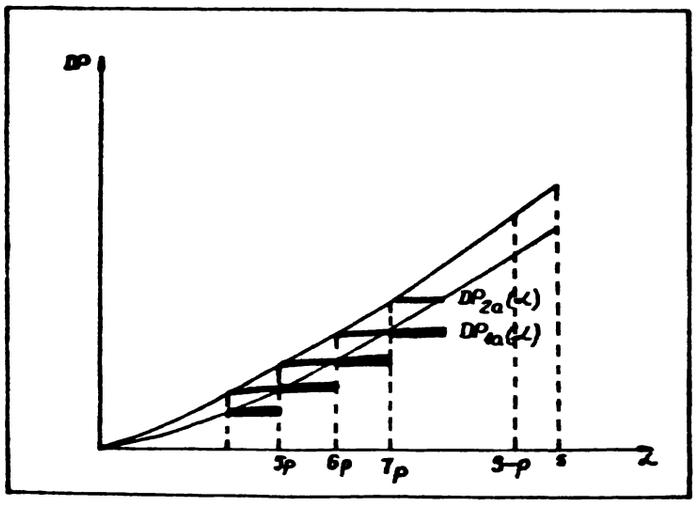


Fig.30. Reprezentarea grafică a funcţiilor $DP_{1a}(\alpha)$ şi $DP_{2a}(\alpha)$.

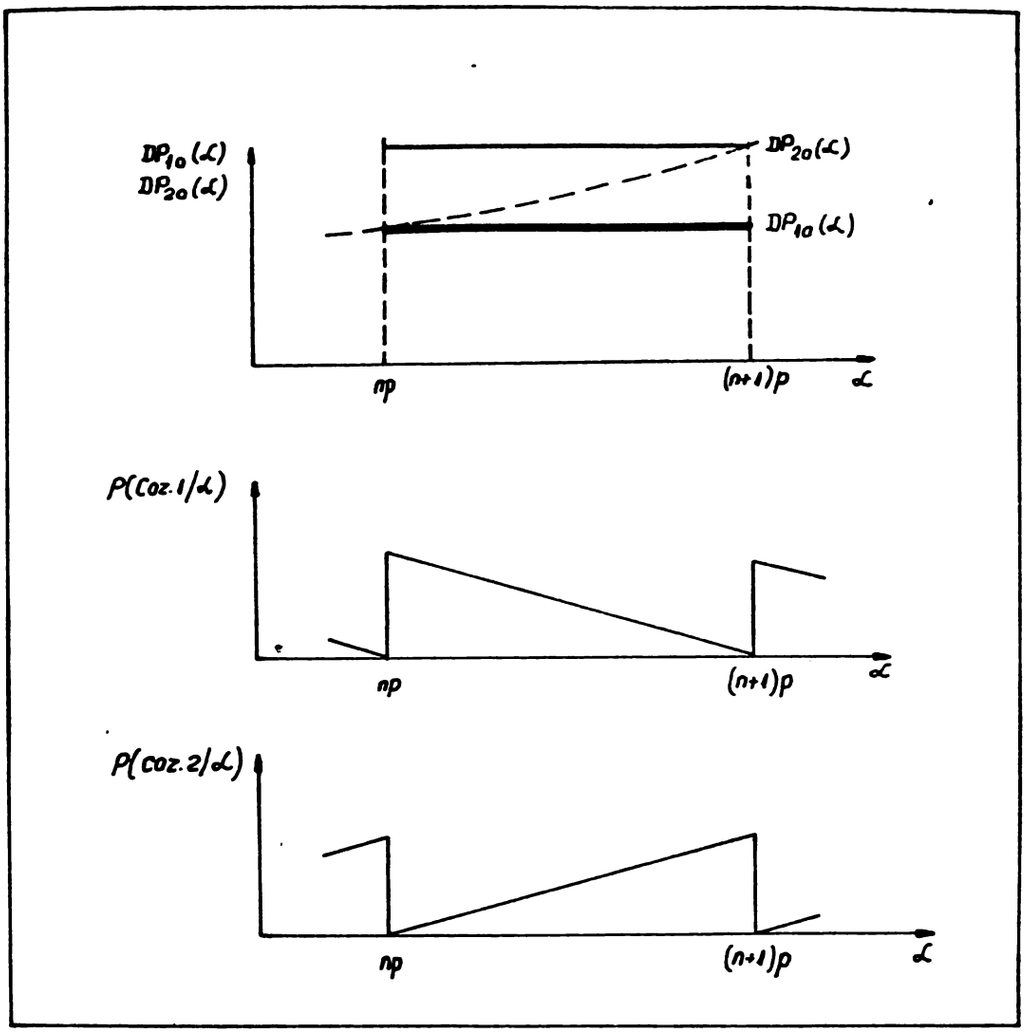


Fig.31. Relativ la analiza expresiei (35).

Valoarea expresiei (35) începe la stînga cu valoarea lui $DP_{1a}(\alpha)$ şi progresează între cele două funcţii $DP_{1a}(\alpha)$ şi $DP_{2a}(\alpha)$, egalînd pe $DP_{2a}(\alpha)$ la limita din dreapta a intervalului considerat. Este evident că expresia nu este niciodată mai mare ca $DP_{2a}(\alpha)$ sau mai mică decît $DP_{1a}(\alpha)$.

Înlocuim (35) cu funcţia:

$$H_1(\alpha) = \frac{\alpha}{p}\left[1 - \frac{s}{p\left[\frac{p}{p}\right]} - \frac{1}{2\left[\frac{p}{p}\right]} + \frac{1}{2\left[\frac{p}{p}\right]} \cdot \frac{\alpha}{p}\right] \qquad (36)$$

$H_1(\alpha)$ reprezintă $DP_{1a}(\alpha)$, în care $\left[\frac{\alpha}{p}\right]$ a fost înlocuit prin $\frac{\alpha}{p}$. Se observă că valorile acestei funcţii sînt egale cu cele ale funcţiilor $DP_{1a}(\alpha)$ şi $DP_{2a}(\alpha)$ pentru valori ale lui α multiplii întregi de dimensiunea paginii (limite de pagină). $H_1(\alpha)$ este o funcţie nedescrescătoare, care satisface

$$DP_{1a}(\alpha) \leq H_1(\alpha) \leq DP_{2a}(\alpha)$$

Deoarece noi sîntem interesaţi în continuare de integrarea produsului $H_1(\alpha)$. $f(\alpha)$ este clar că erorile introduse prin folosirea funcţiei $H_1(\alpha)$ ca aproximaţie pentru expresia (35) sînt triviale cînd sînt comparate cu alegerea destul de arbitrară a funcţiei $f(\alpha)$.

Examinînd celelalte două integrale din relaţia (30) se observă că situaţia, în cadrul fiecăreia, este aproximativ similară. In fiecare avem două funcţii care sînt linii drepte între limitele de pagină şi cu discontinuităţi la limitele de pagină.

Fig.32 reprezintă forma generală a funcţiilor $DP_{1a}(\alpha)$, $DP_{2a}(\alpha)$, $DP_{1b}(\alpha)$ şi $DP_{2b}(\alpha)$.

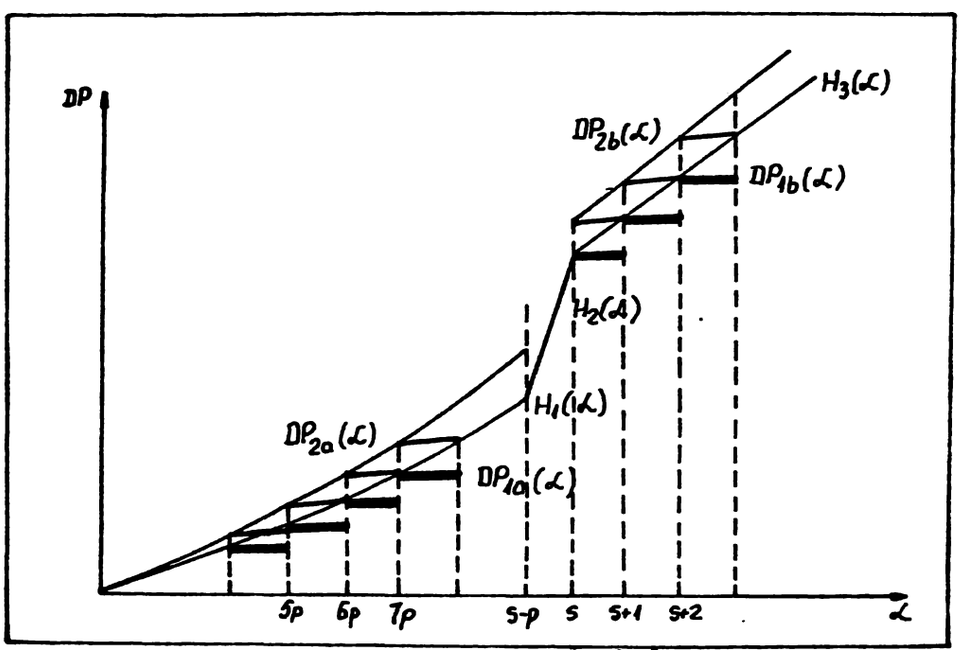


Fig.32. Obţinerea funcţiilor $H_i(\alpha)$.

In integrala a treia din relaţia (30) înlocuim termenul

$$\left[DP_{1b}(\alpha)\ P(\text{Cas } 1|\alpha) + DP_{2b}(\alpha)\ P(\text{Cas } 2|\alpha)\right] \qquad (37)$$

cu:

$$H_3(\alpha) = \frac{\alpha}{p}\sigma - \frac{1}{2\left\lceil\frac{\sigma}{p}\right\rceil}\left(\frac{s^2}{p^2} + \frac{s}{p}\right) \qquad (38)$$

Funcţia $H_3(\alpha)$ poate fi recunoscută ca $DP_{1b}(\alpha)$, unde $\left\lceil\frac{\alpha}{p}\right\rceil$ a fost înlocuit cu $\frac{\alpha}{p}$. Din nou putem arăta că $H_3(\alpha)$ este egală cu expresia (37) în discuţie, la limitele de pagină şi satisface:

$$DP_{1b}(\alpha) \leq H_3(\alpha) \leq DP_{2b}(\alpha)$$

./.

In integrala a doua din relaţia (30) considerăm termenul:

$$DP_{1a}(\alpha)\ P(\text{Con } 1|\alpha) + DP_{2b}(\alpha)\ P(\text{Con } 2|\alpha) \qquad (39)$$

Acest termen combină ambele cazuri a şi b ilustrate în fig.32 în domeniul α cuprins între s - p şi s. Vom înlocui acest termen cu o linie dreaptă ce uneşte valorile termenului la limitele de pagină. Forma generală a acestei aproximaţii este:

$$H_2(\alpha) = (\text{panta}) \cdot \alpha + (\text{const}) \qquad (40)$$

Deoarece valorile funcţiilor $H_1(\alpha)$ şi $H_3(\alpha)$ sînt cunoscute la limitele de pagină, putem calcula panta:

$$\text{panta} = \frac{H_3(s) - H_1(s-p)}{s - (s-p)} = \frac{H_3(s) - H_1(s-p)}{p} \qquad (41)$$

iar constanta se determină prin:

$$\text{constanta} = H_3(s) - (\text{panta}) \times s \qquad (41\ a)$$

Înlocuind în (41) valorile pentru $H_3(s)$ şi $H_1(s-p)$ determinate conform (38) şi (36), se obţine:

$$\text{panta} = \frac{1}{p}\left\{\frac{s}{p}\sigma - \frac{1}{2\left\lceil\frac{P}{p}\right\rceil}\left(\frac{s^2}{p^2} + \frac{s}{p}\right) - \frac{s-p}{p}\left[1 - \frac{s}{p\left\lceil\frac{P}{p}\right\rceil} - \frac{1}{2\left\lceil\frac{P}{p}\right\rceil} + \frac{1}{2\left\lceil\frac{P}{p}\right\rceil}\cdot\frac{s-p}{p}\right]\right\}$$

Efectuînd:

$$\text{panta} = \frac{s}{p^2}(\sigma - 1) + \frac{1}{p}\left(1 - \frac{1}{\left\lceil\frac{P}{p}\right\rceil}\right) \qquad (42)$$

Pentru constantă se obţine:

$$\text{constanta} = \frac{s}{p}\sigma - \frac{1}{2\left\lceil\frac{P}{p}\right\rceil}\left(\frac{s^2}{p^2} + \frac{s}{p}\right) - \frac{s^2}{p^2}(\sigma - 1) - \frac{s}{p}\left(1 - \frac{1}{\left\lceil\frac{P}{p}\right\rceil}\right)$$

sau:

$$\text{constanta} = \left(\sigma - 1 + \frac{1}{2\left\lceil\frac{P}{p}\right\rceil}\right)\left(-\frac{s^2}{p^2} + \frac{s}{p}\right) \qquad (43)$$

Avînd valorile (42) şi (43) pentru pantă şi constantă, forma relaţiei (40) de aproximare a funcţiei $H_2(\alpha)$ devine:

$$H_2(\alpha) = \left[\frac{s}{p^2}(\sigma - 1) + \frac{1}{p}\left(1 - \frac{1}{\left\lceil\frac{P}{p}\right\rceil}\right)\right]\alpha + \left(\sigma - 1 + \frac{1}{2\left\lceil\frac{P}{p}\right\rceil}\right)\left(-\frac{s^2}{p^2} + \frac{s}{p}\right) \qquad (44)$$

Utilizînd funcţiile de aproximare $H_1(\alpha)$ (36), (44) şi (38) valoarea medie (30) a numărului de defecte de pagină se

./.

poate scrie:

$$M\left[DP(\alpha)\right] = \int_{0}^{s-p} H_1(\alpha)\, f(\alpha)\, d\alpha + \int_{s-p}^{s} H_2(\alpha)\, f(\alpha)\, d\alpha +$$

$$+ \int_{s}^{s+p\left[\frac{r}{p}\right]} H_3(\alpha) f(\alpha) d\alpha \qquad (45)$$

Funcţiile $H_1(\alpha)$ sînt de forma:

$$H_1(\alpha): a\alpha^2 + b\alpha$$

$$H_2(\alpha): c\alpha + d \qquad (46)$$

$$H_3(\alpha): e\alpha + f$$

In relaţia (45) intervin deci următoarele 3 tipuri de integrale:

$$\int_0^L C_1 f(\alpha)\, d\alpha \;;$$

$$\int_0^L C_2 \alpha f(\alpha)\, d\alpha \;; \qquad \text{şi}$$

$$\int_0^L C_3 \alpha^2 f(\alpha)\, d\alpha$$

Ţinînd seama că $f(\alpha)$ este funcţia de densitate pentru repartiţia de tip Γ :

$$f(\alpha) = \frac{1}{\Gamma(\beta)} \varsigma (\varsigma\alpha)^{\beta-1} e^{-\varsigma\alpha}$$

pentru cele 3 tipuri de integrale se obţin valorile următoare:

Pentru primul tip:

$$\int_0^L C_1 \frac{1}{\Gamma(\beta)} \varsigma (\varsigma\alpha)^{\beta-1} e^{-\varsigma\alpha} d\alpha = C_1 \frac{1}{\Gamma(\beta)} \int_0^L (\varsigma\alpha)^{\beta-1} e^{-\varsigma\alpha} d(\varsigma\alpha) = C_1 I\left[\varsigma L; \beta\right] \qquad (47)$$

Pentru următoarele două tipuri de integrale, conform relaţiilor (22) şi (23) se obţin:

$$\int_0^L C_2 \alpha f(\alpha) d\alpha = C_2 \frac{\beta}{\varsigma} I\left[\varsigma L, \beta + 1\right] \qquad \text{şi} \qquad (48)$$

$$\int_0^L C_3 \alpha^2 f(\alpha)\, d\alpha = C_3 \frac{(\beta+1)\beta}{2} I\left[\varsigma L; \beta+2\right] \qquad (49)$$

./.

Aplicînd relaţiile (47) - (49) pentru integralele din partea dreaptă a expresiei (45), funcţiile $M_i(\alpha)$ avînd forma (46), se obţine:

$$M[DP(\alpha)] = a\,\frac{(\beta+1)\beta}{2}\,I[\varsigma(s-p);\beta+2] + b\,\frac{\beta}{\varsigma}\,I[\varsigma(s-p);\beta+1] +$$

$$+ c\,\frac{\beta}{\varsigma}\left\{I[\varsigma s;\beta+1] - I[\varsigma(s-p;\beta+1]\right\} +$$

$$+ d\left\{I[\varsigma s;\beta] - I[\varsigma(s-p);\beta]\right\} + e\,\frac{\beta}{\varsigma}\left\{I[\varsigma p\lceil\tfrac{p}{p}\rceil;\beta+1] - I[\varsigma s;\beta+1]\right\}$$

$$+ f\left\{I[\varsigma p\lceil\tfrac{p}{p}\rceil,\beta] - I[\varsigma s;\beta]\right\} \qquad (50)$$

Coeficienţii a - f au valorile:

$$a = \frac{1}{p^2}\cdot\frac{1}{2\lceil\frac{p}{p}\rceil}\;;$$

$$b = \frac{1}{p}\left(1 - \frac{s}{p\lceil\frac{p}{p}\rceil} - \frac{1}{2\lceil\frac{p}{p}\rceil}\right),$$

$$c = \frac{s}{p^2}(\sigma-1) + \frac{1}{p}\left(1 - \frac{1}{\lceil\frac{p}{p}\rceil}\right); \qquad (51)$$

$$d = \left(\sigma - 1 + \frac{1}{2\lceil\frac{p}{p}\rceil}\right)\left(-\frac{s^2}{p^2} + \frac{s}{p}\right);$$

$$e = \frac{\sigma}{p}$$

$$f = \frac{1}{2\lceil\frac{p}{p}\rceil}\left(\frac{s^2}{p^2} + \frac{s}{p}\right)$$

In continuare vom obţine o expresie echivalentă cu (50) pentru determinarea numărului mediu de defecte de pagină, dar în care toate funcţiile Γ incomplete de forma $I[x;\alpha]$ să aibă parametrii α egali. Deci dorim a determina expresii echivalente pentru integralele (48) şi (49).

Astfel:

$$I_{48} = \int_0^I C\alpha f(\alpha)d\alpha = C\,\frac{1}{\Gamma(\beta)}\int_0^L (\varsigma\alpha)^\beta e^{-\varsigma\alpha}d\alpha = C\,\frac{1}{\varsigma}\cdot\frac{1}{\Gamma(\beta)}\int_0^I x^\beta e^{-x}dx$$

./.

Integrînd prin părţi:

$x^{\beta} = u$ $\qquad e^{-x}dx = dv$

$du = \beta x^{\beta-1}dx$ $\qquad v = -e^{-x}$

$$I_{43} = C\frac{1}{\varsigma}\frac{1}{\Gamma(\beta)}\left\{\left|-x^{\beta}e^{-x}\right|_0^{\varsigma L} + \beta\int_0^{\varsigma L} x^{\beta-1}e^{-x}dx\right\} =$$

$$= -C\frac{1}{\varsigma}\frac{1}{\Gamma(\beta)}(\varsigma L)^{\beta}e^{-\varsigma L} + C\frac{\beta}{\varsigma}I[\varsigma L;\beta] \qquad (48\ a)$$

Iar

$$I_{49} = \int_0^L C\alpha^2 f(\alpha)d\alpha = C\frac{1}{\varsigma}\frac{1}{\Gamma(\beta)}\int_0^L (\varsigma\alpha)^{\beta+1}e^{-\varsigma\alpha}d\alpha =$$

$$= C\frac{1}{\varsigma^2}\frac{1}{\Gamma(\beta)}\int_0^{\varsigma L} x^{\beta+1}e^{-x}dx$$

Integrînd prin părţi:

$x^{\beta+1} = u$ $\qquad e^{-x}dx = dv$

$du = (\beta+1)x^{\beta}dx$ $\qquad v = -e^{-x}$

$$I_{49} = C\frac{1}{\varsigma^2}\frac{1}{\Gamma(\beta)}\left\{\left|-x^{\beta+1}e^{-x}\right|_0^{\varsigma L} + (\beta+1)\int_0^{\varsigma L} x^{\beta}e^{-x}dx\right\} =$$

$$= -C\frac{1}{\varsigma^2}\frac{1}{\Gamma(\beta)}(\varsigma L)^{\beta+1}e^{-\varsigma L} + C\frac{\beta+1}{\varsigma^2}\cdot\frac{1}{\Gamma(\beta)}\int_0^{\varsigma L} x^{\beta}e^{-x}dx$$

Pentru integrala din partea dreaptă putem aplica egalitatea (48 a) stabilită mai sus, deci:

$$I_{49} = -C\frac{1}{\varsigma^2}\frac{1}{\Gamma(\beta)}(\varsigma L)^{\beta+1}e^{-\varsigma L} + C\frac{\beta+1}{\varsigma^2}\left[-\frac{1}{\Gamma(\beta)}\cdot(\varsigma L)^{\beta}e^{-\varsigma L} + \beta I[\varsigma L;\beta]\right]$$

sau:

$$I_{49} = -C\frac{1}{\varsigma^2}\frac{1}{\Gamma(\beta)}e^{-\varsigma L}(\varsigma L)^{\beta}[\varsigma L + (\beta+1)] + C\frac{(\beta+1)\beta}{\varsigma^2}I[\varsigma L;\beta] \qquad (49\ a)$$

Aplicînd relaţiile (48 a) şi (49 a) pentru exprimarea funcţiilor Γ incomplete din relaţia (50) se obţine următoarea formă echivalentă:

./.

$$M[DP(\alpha)] = -a \cdot \frac{1}{\varsigma^2} \cdot \frac{1}{\Gamma(\beta)} [\varsigma(s-p)]^{\beta} e^{-\varsigma(s-p)} [\varsigma(s-p) + \beta + 1] +$$

$$+ a \frac{(\beta+1)\beta}{\varsigma^2} I[\varsigma(s-p);\beta] - b \cdot \frac{1}{\varsigma} \cdot \frac{1}{\Gamma(\beta)} [\varsigma(s-p)]^{\beta} e^{-\varsigma(s-p)} +$$

$$+ b \cdot \frac{\beta}{\varsigma} I[\varsigma(s-p);\beta] + c \left\{ -\frac{1}{\varsigma} \frac{1}{\Gamma(\beta)} [\varsigma s]^{\beta} e^{-\varsigma s} + \right.$$

$$\left. + \frac{1}{\varsigma} \frac{1}{\Gamma(\beta)} [\varsigma(s-p)]^{\beta} e^{-\varsigma(s-p)} + \frac{\beta}{\varsigma} I[\varsigma s,\beta] - \frac{\beta}{\varsigma} I[\varsigma(s-p),\beta] \right\} +$$

$$+ d \left\{ I[\varsigma s,\beta] - I[\varsigma(s-p),\beta] \right\} + e \left\{ -\frac{1}{\varsigma} \frac{1}{\Gamma(\beta)} \left[\varsigma p \left[\frac{p}{p}\right]\right]^{\beta} e^{-\varsigma p\left[\frac{p}{p}\right]} \right.$$

$$\left. + \frac{1}{\varsigma} \frac{1}{\Gamma(\beta)} [\varsigma s]^{\beta} e^{-\varsigma s} + \frac{\beta}{\varsigma} I\left[\varsigma p\left[\frac{p}{p}\right],\beta\right] - \frac{\beta}{\varsigma} I[\varsigma s;\beta] \right\} +$$

$$+ f \left\{ I\left[\varsigma p\left[\frac{p}{p}\right],\beta\right] - I[\varsigma s;\beta] \right\} \qquad (52)$$

Grupînd termenii în partea dreaptă:

$$M[DP(\alpha)] = \frac{1}{\varsigma} \frac{1}{\Gamma(\beta)} [\varsigma(s-p)]^{\beta} e^{-\varsigma(s-p)} \left[-a(s-p) - \frac{a(\beta+1)}{\varsigma} - b + c\right] +$$

$$+ \frac{1}{\varsigma} \cdot \frac{1}{\Gamma(\beta)} [\varsigma s]^{\beta} e^{-\varsigma s} [-c + e] + \frac{1}{\varsigma} \frac{1}{\Gamma(\beta)} \left[\varsigma p\left[\frac{p}{p}\right]\right]^{\beta} e^{-\varsigma p\left[\frac{p}{p}\right]} [-e +$$

$$+ I[\varsigma(s-p);\beta] \left[a \frac{(\beta+1)\beta}{\varsigma^2} + b \cdot \frac{\beta}{\varsigma} - c \frac{\beta}{\varsigma} - d\right] +$$

$$+ I[\varsigma s;\beta] \left[c \frac{\beta}{\varsigma} + d - e \frac{\beta}{\varsigma} - f\right] + I\left[\varsigma p\left[\frac{p}{p}\right],\beta\right] \left[e \frac{\beta}{\varsigma} + f\right] \qquad (53)$$

Relaţiile echivalente (50) şi (53) reprezintă expresia analitică căutată de noi care determină numărul de defecte de pagină care apar în decursul execută̆rii unui program într-un mediu cu memorie virtuală. Aceste expresii vor fi de importanţă centrală în cele ce urmează.

2.6. Validarea modelului elaborat.

Modelul comportă̆rii de acces a unui program într-un mediu cu memorie virtuală construit mai sus este verificat în continuare prin faptul că pe baza lui se regăsesc pe cale analitică:

(1) caracteristicile calitative ale funcţiei ce determină numărul defectelor de pagină generate la execuţia unui program, caracteristici observate în cadrul unor studii de simulare publicate;

./.

(2) rezultatele numerice experimentale.

2.6.1. Proprietăţile funcţiei DPA.

In cadrul unui sistem ce adoptă paginarea la cerere un program îşi începe execuţia avînd o singură pagină dispusă în memoria principală; pentru satisfacerea necesităţilor de instrucţii şi date restul de pagini ale programului va fi adus în memoria principală treptat. La începutul execuţiei programul produce defecte de pagină la intervale de execuţie foarte scurte; după ce programul a acumulat un număr suficient de pagini din cerinţa lui totală de pagini se observă intervale mai mari de execuţie.

Astfel, studiul de simulare (72) stabileşte că:

- programele tind să ceară pagini cu rate mari pînă la obţinerea unei suficienţe de pagini;

- de obicei programele nu rulează foarte mult timp după ce au obţinut o suficienţă de pagini;

- pentru acele cereri de pagini ale programului, care conduc la timpi de rulare continuă (fără alte cereri de pagini) importanţi, o suficienţă de pagini înseamnă o fracţiune considerabilă din cererea de pagini totală a programului.

Studiile de simulare (40,72,75,102) prezintă mostre de referire a paginilor prin analiza şirului de referiri de pagină a unor programe tipice. Rezultatele prezentate în (40,72) sînt larg acceptate în cercetările actuale asupra comportării de acces a programelor în cadrul unui sistem cu paginare la cerere; o însumare a acestor rezultate este prezentată în fig.33 şi 34.

Fig.33 este adaptată din studiile de simulare din (40). In lucrarea (40) au fost obţinute rezultate separate pentru pagini de instrucţii şi pagini de date şi ambele dovedesc caracteristici similare. Din fig.33 se observă că numărul de instrucţii executate între defecte succesive de pagină creşte de la R_1 la R_2 la o creştere a numărului de pagini acumulate de program de la 1 la C, la o valoare constantă F[x] a funcţiei de repartiţie cumulative (FRC) a instrucţiilor executate între defecte succesive de pagină.

./.

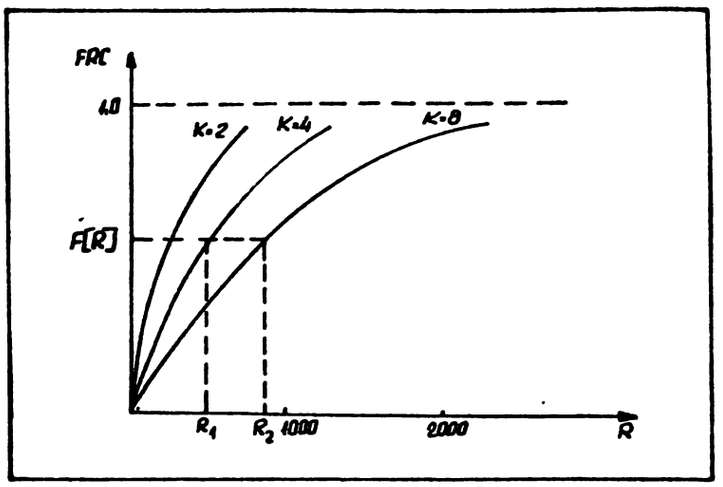


Fig.33. Funcţie de repartiţie cumulativă FRC a instrucţiilor executate între defecte succesive de pagină.

F[R] = **P** (numărul de instrucţii executate între defecte succesive de pagină $\leq R$);

K = numărul de pagini acumulate de program;

R = numărul de instrucţii executate.

Programe tipice analizate au fost compilatorul WATFOR FORMULE transformări FOURIER etc.

Fig.34 este adoptată din studiile de simulare din (72). Referirile la paginile de instrucţii şi la paginile de date nu au fost separate. Din fig.34 se observă că numărul paginilor cerute de program prin defecte de pagină este o funcţie concavă în dependenţă de numărul de instrucţii executate; această funcţie îşi atinge asimptotic limita superioară. De asemenea se constată că în faza de început a execuţiei programului referirea la noi pagini se face într-un ritm rapid.

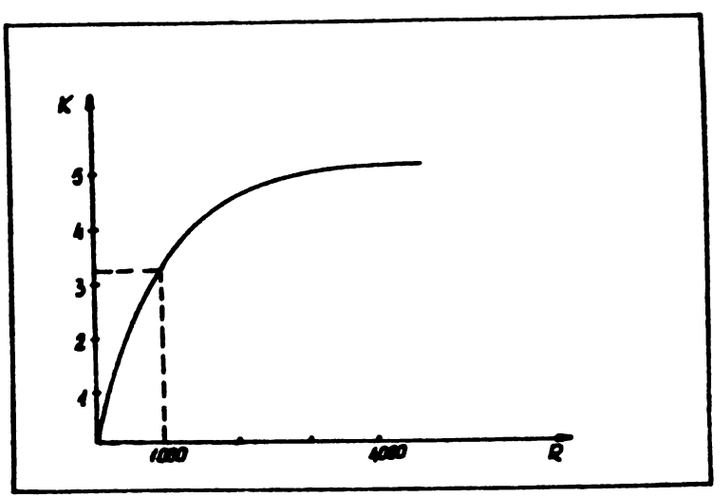


Fig.34. Cererea de pagini a unui program.

K = numărul de pagini referite;

R = numărul de instrucţii executate.

./.

Programe tipice analizate în obţinerea acestor rezultate au fost Program de tratarea listelor LISP, compilatorul LETA 5 etc.

In termenii modelului elaborat de noi pentru comportarea de acces a unui program, cererea de pagini a unui program reprezintă numărul defectelor de pagină generate până la un anumit moment în execuţia acestui program.

Ţinînd seama de rezultatele prezentate mai sus şi obţinute prin analiza unor şiruri de referiri de pagină, generate în cadrul execuţiei unor programe reale, pentru verificarea modelului elaborat va fi necesar să arătăm că el oferă posibilitatea de a stabili pe cale analitică următoarele 3 caracteristici ale funcţiei DPA (ce determină numărul defectelor de pagină generate la execuţia unui program):

1 - DPA este o funcţie concavă în dependenţa de numărul de instrucţii executate;

2 - DPA are o asimptotă pentru valori mari ale numărului de instrucţii executate;

3 - numărul de instrucţii executate de program între defecte succesive de pagină creşte cu numărul de pagini acumulate de program.

Vom stabili aceste proprietăţi ale funcţiei DPA.

2.6.1.1. Proprietatea 1: DPA este o funcţie concavă.

Fie relaţia (45) ce defineşte valoarea medie a numărului de defecte de pagină:

$$DPA = \int_0^{s-p} H_1(\alpha)f(\alpha)d\alpha + \int_{s-p}^{s} H_2(\alpha)f(\alpha)d\alpha + \int_{s}^{p\left\lceil\frac{p}{p}\right\rceil} H_3(\alpha)f(\alpha)d\alpha \quad (54)$$

funcţiile $H_1(\alpha)$ având forma:

$$H_1(\alpha) = a\alpha^2 + b\alpha$$

$$H_2(\alpha) = c\alpha + d$$

$$H_3(\alpha) = e\alpha + f$$

iar coeficienţii a - f având valorile (51).

Prin intermediul coeficienţilor **a-f**, DPA este o funcţie de dimensiunea programului (de numărul corespunzător de instrucţii).

./.

Calea ce o vom urma pentru a demonstra că DPA este o funcție concavă este de a arăta că $\frac{d^2(DPA)}{dP^2} < 0$ și ca atare tangenta, în orice punct al graficului DPA(P), se află situată deasupra graficului.

Pentru simplificarea expresiilor analitice și fără a influența în nici un fel generalitatea rezultatelor, se poate presupune că dimensiunea programului P este multiplu întreg de dimensiunea paginii p. În acest caz, valorile coeficienților a-f devin:

$$a = \frac{1}{2Pp}\ ;$$

$$b = \frac{1}{p} - \frac{s}{Pp} - \frac{1}{2P}\ ;$$

$$c = \frac{s}{p^2}(\sigma - 1) + \frac{1}{p} - \frac{1}{P}\ ;$$

$$d = -(\sigma - 1)\frac{s^2}{p^2} + \frac{s}{p}\left(\sigma - 1 - \frac{s}{2P}\right) + \frac{s}{2P}. \qquad (53)$$

$$e = \frac{\sigma}{p}\ ;$$

$$f = -\frac{s^2}{2Pp} - \frac{s}{2P}$$

Funcțiile $H_1(\alpha)$ se pot transcrie:

$$H_1(\alpha) = a\alpha^2 + b = \frac{1}{2Pp}\alpha^2 + \left(\frac{1}{p} - \frac{s}{Pp} - \frac{1}{2P}\right)\alpha = \frac{1}{P}\left(\frac{\alpha^2}{2p} - \frac{s\alpha}{p} - \frac{\alpha}{2}\right) + \frac{\alpha}{p}$$

$$H_2(\alpha) = c\alpha + d = \frac{s}{p^2}(\sigma - 1) + \frac{\alpha}{p} - \frac{\alpha}{P} - (\sigma - 1)\frac{s^2}{p^2} + \frac{s}{p}\left(\sigma - 1 - \frac{s}{2P}\right) + \frac{s}{2P}$$

$$= \frac{1}{P}\left(-\alpha - \frac{s^2}{2p} + \frac{s}{2}\right) + \left[\frac{s(\sigma - 1)}{p^2} + \frac{1}{p}\right]\alpha - (\sigma - 1)\frac{s^2}{p^2} + \frac{s}{p}(\sigma - 1)\ ;$$

$$H_3(\alpha) = e\alpha + f = \frac{\sigma}{p}\alpha - \frac{s^2}{2Pp} - \frac{s}{2P} = \frac{1}{P}\left(-\frac{s^2}{2p} - \frac{s}{2}\right) + \frac{\sigma}{p}\alpha. \qquad (56)$$

Expresia (54) devine, considerînd-o funcție de P:

$$DPA(P) = \int_0^{s-p}\left[\frac{1}{P}\left(\frac{\alpha^2}{2p} - \frac{s\alpha}{p} - \frac{\alpha}{2}\right) + \frac{\alpha}{p}\right]f(\alpha)d\alpha + \int_{s-p}^{s}\left\{\frac{1}{P}\left(-\alpha - \frac{s^2}{2p} + \frac{s}{2}\right) + \right.$$

$$\left. + \left[\frac{s(\sigma - 1)}{p^2} + \frac{1}{p}\right]\alpha - (\sigma - 1)\frac{s^2}{p^2} + \frac{s}{p}(\sigma - 1)\right\}f(\alpha)d\alpha +$$

$$+ \int_s^P\left[\frac{1}{P}\left(-\frac{s^2}{2p} - \frac{s}{2}\right) + \frac{\sigma}{p}\right]f(\alpha)d\alpha = T_1 + T_2 + T_3 \qquad (57)$$

./.

Calculăm coeficienții derivatelor secunde după x ale celor trei termeni T_1 din partea dreaptă a expresiei (37).

Astfel:

$$\frac{d^2T_1}{dP^2} = \int_0^{s-p} \frac{2}{p^3}\left(\frac{\alpha^2}{2p} - \frac{s\alpha}{p} - \frac{\alpha}{2}\right) f(\alpha)d\alpha =$$

$$= \frac{1}{p^3}\cdot\frac{1}{p}\int_0^{s-p}\alpha^2 f(\alpha)d\alpha - \frac{1}{p^3}\cdot\frac{2s+p}{p}\int_0^{s-p}\alpha f(\alpha)d\alpha =$$

$$= \frac{1}{p^3}\cdot\frac{1}{p}\int_0^{s-p}\alpha^2\frac{\rho^\beta}{\Gamma(\beta)}\alpha^{\beta-1}e^{-\rho\alpha}d\alpha - \frac{1}{p^3}\cdot\frac{2s+p}{p}\cdot$$

$$\cdot\int_0^{s-p}\alpha f(\alpha)\,d\alpha \qquad (39)$$

Primul integral devine:

$$\frac{1}{p^3}\frac{1}{p}\frac{1}{\Gamma(\beta)}\int_0^{s-p}\rho^\beta\alpha^{\beta+1}e^{-\rho\alpha}d\alpha = \frac{1}{p^3}\frac{1}{p}\frac{1}{\Gamma(\beta)}\frac{1}{\rho^2}\int_0^{s-p}(\rho\alpha)^{\beta+1}e^{-\rho\alpha}d(\rho\alpha)$$

$$= \frac{1}{p^3}\frac{1}{p}\frac{1}{\Gamma(\beta)}\frac{1}{\rho^2}\int_0^{\rho(s-p)}x^{\beta+1}e^{-x}dx$$

Integrînd prin părți:

$x^{\beta+1} = u$ $\qquad$ $e^{-x}dx = dv$

$du = (\beta+1)x^\beta dx$ $\qquad$ $v = -e^{-x}$

$$\frac{1}{p^3}\cdot\frac{1}{p}\frac{1}{\Gamma(\beta)}\frac{1}{\rho^2}\left\{\left|-x^{\beta+1}e^{-x}\right|_0^{\rho(s-p)} + \int_0^{\rho(s-p)}(\beta+1)x^\beta e^{-x}dx\right\} =$$

$$= -\frac{1}{p^3}\frac{1}{p}\frac{1}{\rho^2}\frac{1}{\Gamma(\beta)}\left[\rho(s-p)\right]^{\beta+1}e^{-\rho(s-p)} + \frac{1}{p^3}\frac{1}{p}\frac{1}{\rho^2}\frac{\beta+1}{\Gamma(\beta)}\cdot$$

$$\cdot\int_0^{\rho(s-p)}x^{(\beta+1)-1}e^{-x}dx = -\frac{1}{p^3}\frac{1}{p}\frac{1}{\rho^2}\frac{1}{\Gamma(\beta)}\left[\rho(s-p)\right]^{\beta+1}e^{-\rho(s-p)} +$$

$$+\frac{1}{p^3}\frac{1}{p}\frac{\beta(\beta+1)}{\rho^2}I\left[\rho(s-p);\beta+1\right] \qquad (39)$$

A doua integrală din expresia (39) devine, conform relației (43):

./.

$$\frac{1}{p^3}\,\frac{2s+p}{p}\int_0^{s-p}\alpha\,f(\alpha)d\alpha=\frac{1}{p^3}\,\frac{2s+p}{p}\,\frac{\beta}{\varsigma}\,I\left[\varsigma(s-p),\,\beta+1\right]\qquad(60)$$

Pe baza valorilor integralelor (59) şi (60) se poate determina valoarea expresiei (58):

$$\frac{d^2T_1}{d\,P^2}=-\frac{1}{p^3}\,\frac{1}{p}\,\frac{1}{\varsigma^2}\,\frac{1}{\Gamma(\beta)}\left[\varsigma(s-p)\right]^{\beta+1}e^{-\varsigma(s-p)}+\frac{1}{p^3}\,\frac{1}{p}\cdot$$

$$\cdot\,\frac{\beta(\beta+1)}{\varsigma^2}\,I\left[\varsigma(s-p);\,\beta+1\right]-\frac{1}{p^3}\,\frac{2s+p}{p}\,\frac{\beta}{\varsigma}\,I\left[\varsigma(s-p);\,\beta+1\right]=$$

$$=-\frac{1}{p^3}\,\frac{1}{p}\cdot\frac{1}{\varsigma^2}\,\frac{1}{\Gamma(\beta)}\left[\varsigma(s-p)\right]^{\beta+1}e^{-\varsigma(s-p)}-$$

$$-\frac{1}{p^3}\,\frac{1}{p}\,\frac{\beta}{\varsigma}\,I\left[\varsigma(s-p),\,\beta+1\right]\left[(2s+p)-\frac{\beta+1}{\varsigma}\right]$$

Întrucât, în mod obişnuit $2s+p>\frac{\beta+1}{\varsigma}$ ≈ lungimea medie a ciclurilor programului, rezultă că:

$$\frac{d^2T_1}{d\,P^2}<0\qquad(61)$$

Trecem la calculul derivatei secunde după P a termenului al doilea din (57):

$$\frac{d^2T_2}{d\,P^2}=\int_{s-p}^{s}\frac{2}{P^3}\left(-\alpha-\frac{s^2}{2p}+\frac{s}{2}\right)f(\alpha)d\alpha=$$

$$=-\frac{2}{P^3}\int_{s-p}^{s}\alpha\,f(\alpha)d\alpha-\frac{1}{P^3}\,\frac{s}{p}\cdot(s-p)\int_{s-p}^{s}f(\alpha)\,d\alpha\qquad(62)$$

Deoarece integralele $\int_{s-p}^{s}\alpha\,f(\alpha)d\alpha$ şi $\int_{s-p}^{s}f(\alpha)d\alpha$ sînt pozitive, conform (47)şi (48),

$$\int_{s-p}^{s}\alpha\,f(\alpha)d\alpha=\frac{\beta}{\varsigma}\left\{I\left[\varsigma s;\,\beta+1\right]-I\left[\varsigma(s-p);\,\beta+1\right]\right\}$$

$$\int_{s-p}^{s}f(\alpha)\,d\alpha=I\left[\varsigma s;\beta\right]-I\left[\varsigma(s-p),\beta\right]$$

./.

pe baza (62) rezultă că:

$$\frac{d^2 m_c}{d P^2} < 0 \qquad (63)$$

In final ne rămîne a determina derivata a doua a ultimului termen din expresia (57).

$$\frac{d^2}{dP^2} \int_s^P \left[\frac{1}{P} \left(- \frac{\alpha^2}{2p} - \frac{s}{2} \right) + \frac{\sigma}{p} \alpha \right] f(\alpha)\, d\alpha =$$

$$= - \frac{s}{2p} (s+p) \frac{d^2}{dP^2} \int_s^P \frac{1}{P} f(\alpha)\, d\alpha + \frac{\sigma}{p} \frac{d^2}{d P^2} \int_s^P \alpha f(\alpha)\, d\alpha =$$

$$= - \frac{s}{2p}(s+p) \left[\frac{2}{P^3} \int_{s_P}^P f(\alpha) d\alpha - \frac{2}{P^2} f(P) + \frac{1}{P} f'(P) \right] + \frac{\sigma}{p} \left[f(P) + P f'(P) \right] =$$

$$= - \frac{s}{2p} (s+p) \frac{2}{P^3} \int_s f(\alpha) d\alpha + \frac{s(s+p)}{p P^2} \left[f(P) - \frac{P}{2} f'(P) \right] + \frac{\sigma}{p} \left[f(P) + P f'(P) \right] \qquad (64)$$

In expresia de mai sus primul termen este evident negativ, căci integrala $\int_s^P f(\alpha) d\alpha$ este pozitivă, funcţia $f(\alpha)$ fiind pozitivă, iar $P > s$.

Relativ la ultimii doi termeni din expresia (64) ei pot fi transformaţi în modul următor:

$$\frac{s(s+p)}{p P^2} \left[f(P) - \frac{P}{2} f'(P) \right] + \frac{\sigma}{p} \left[f(P) + P f'(P) \right] =$$

$$= \frac{\sigma}{p} \left\{ \frac{1}{\sigma} \cdot \frac{s(s+p)}{P^2} \left[f(P) - \frac{P}{2} f'(P) \right] + \left[f(P) + P f'(P) \right] \right\} \qquad (65)$$

Notăm $\frac{1}{\sigma} \cdot \frac{s(s+p)}{P^2} = \Gamma$

Relativ la mărimea lui Γ se pot face următoarele consideraţii: Intrucît valoarea σ, care reprezintă numărul de ori de care un ciclu α este apelat, are valoarea:

$$\sigma >> 1,$$

iar

$$\frac{s(s+p)}{P^2} < 1 \qquad \text{(căci } s < P\text{)}$$

rezultă că:

$$\Gamma << 1$$

Expresia (65) devine:

$$\frac{\sigma}{p}\left\{\Gamma\left[f(P)-\frac{P}{\zeta}f'(P)\right]+\left[f(P)+Pf'(P)\right]\right\}=\frac{\sigma}{p}\left[f(P)(\Gamma+1)+Pf'(P)\left(1-\frac{\Gamma}{2}\right)\right] \tag{66}$$

Determinăm $f'(P)$

$f(P)$ este cunoscută ca:

$$f(P)=\frac{\zeta^{\beta}}{\Gamma(\beta)}P^{\beta-1}e^{-\zeta P} \tag{67}$$

deci:

$$f'(P)=\frac{\zeta^{\beta}}{\Gamma(\beta)}\left[(\beta-1)P^{\beta-2}e^{-\zeta P}+P^{\beta-1}(-\zeta)e^{-\zeta P}\right]=$$

$$=\frac{\zeta^{\beta}}{\Gamma(\beta)}P^{\beta-1}e^{-\zeta P}\left(\frac{\beta-1}{P}-\zeta\right)=f(P)\left(\frac{\beta-1}{P}-\zeta\right) \tag{68}$$

Înlocuind (68) în (66) se obține:

$$\frac{\sigma}{p}f(P)\left[(\Gamma+1)+P\left(\frac{\beta-1}{P}-\zeta\right)\left(1-\frac{\Gamma}{2}\right)\right]=\frac{\sigma}{p}f(P).$$

$$\cdot\left[(\Gamma+1)+(\beta-1)\left(1-\frac{\Gamma}{2}\right)-P\zeta\left(1-\frac{\Gamma}{2}\right)\right] \tag{69}$$

Se poate arăta ușor că parantesa mare este negativă, adică:

$$P>\frac{(\Gamma+1)+(\beta-1)\left(1-\frac{\Gamma}{2}\right)}{\zeta\left(1-\frac{\Gamma}{2}\right)} \tag{70}$$

Inegalitatea (70) rezultă astfel: S-a arătat mai sus că $\Gamma<<1$, deci Γ poate fi omis în parantezele $(\Gamma+1)$ și $\left(1-\frac{\Gamma}{2}\right)$; ca urmare trebuie arătat că:

$$P>\frac{\beta}{\zeta}$$

ori acest lucru este evident: $\frac{\beta}{\zeta}$ reprezintă valoarea medie a lungimii ciclurilor programului, iar dimensiunea programului este în mod firesc mai mare ca această lungime medie a ciclurilor componente.

Dovedind că suma celor trei termeni ai expresiei (64) este negativă, s-a demonstrat că:

$$\frac{d^2n_2}{dP^2}<0 \tag{71}$$

Ținînd seama de relațiile (61), (63) și (71) rezultă că:

$$\frac{d^2\left[DPA(P)\right]}{dP^2}<0$$

./.

şi ca urmare DPA(P) este o funcţie concavă relativ la numărul de instrucţii executate.

2.6.1.2. Proprietatea 2: DPA are o asimptotă pentru valori mari ale numărului de instrucţii executate.

Fie o dimensiune mare P a unui program ce se execută. In acest caz, coeficienţii a-f (55) devin:

$$a = 0;$$

$$b = \frac{1}{p};$$

$$c = \frac{s}{p^2}(\sigma-1) + \frac{1}{p};$$

$$d = -(\sigma-1)\frac{s}{p}\left(\frac{s}{p}-1\right); \qquad (72)$$

$$e = \frac{\sigma}{p} \text{ şi}$$

$$f = 0$$

Relativ la forma şi valoarea unor termeni în expresia (53), în această situaţie, se observă:

$$\lim_{P\to\infty} \frac{1}{\varsigma}\,\frac{1}{\Gamma(\beta)}\,\frac{\left[\varsigma p\left[\frac{P}{p}\right]\right]^{\beta}}{e^{\varsigma p\left[\frac{P}{p}\right]}} = 0$$

iar

$$-b + c = \frac{s}{p^2}(\sigma-1)$$

$$-c + e = -\frac{s}{p^2}(\sigma-1) - \frac{1}{p} + \frac{\sigma}{p} = -\frac{(\sigma-1)}{p}\left(\frac{s}{p}-1\right)$$

$$b\frac{\beta}{\varsigma} - c\frac{\beta}{\varsigma} - d = \frac{1}{p}\frac{\beta}{\varsigma} - \frac{\beta}{\varsigma}\left[\frac{s}{p^2}(\sigma-1) + \frac{1}{p}\right] + (\sigma-1)\frac{s}{p}\left(\frac{s}{p}-1\right) =$$

$$= \frac{s}{p}(\sigma-1)\left(-\frac{\beta}{\varsigma p} + \frac{s}{p} - 1\right)$$

$$c\frac{\beta}{\varsigma} + d - e\frac{\beta}{\varsigma} = d + \frac{\beta}{\varsigma}(c-e) = -(\sigma-1)\frac{s}{p}\left(\frac{s}{p}-1\right) + \frac{\beta}{\varsigma}\frac{\sigma-1}{p}\left(\frac{s}{p}-1\right)$$

$$= \frac{\sigma-1}{p}\left(\frac{s}{p}-1\right)\left(\frac{\beta}{\varsigma} - s\right)$$

$$\lim_{P\to\infty} I\left[\varsigma p\left[\frac{P}{p}\right];\beta\right] = \frac{1}{\Gamma(\beta)}\Gamma(\beta) = 1$$

./.

Expresia (53) devine:

$$\lim_{P\to\infty} M[DP(\alpha)] = \frac{1}{\varsigma}\,\frac{1}{\Gamma(\beta)}[\varsigma(s-p)]^{\beta} e^{-\varsigma(s-p)}\cdot\frac{s}{p^2}(\sigma-1) +$$

$$+ \frac{1}{\varsigma}\,\frac{1}{\Gamma(\beta)}[\varsigma s]^{\beta} e^{-\varsigma s}(-\frac{\sigma-1}{p})(\frac{s}{p}-1) +$$

$$+ I[\varsigma(s-p);\beta]\frac{s}{p}(\sigma-1)(-\frac{\beta}{\varsigma p}+\frac{s}{p}-1) +$$

$$+ I[\varsigma s,\beta]\frac{\sigma-1}{p}(\frac{s}{p}-1)(\frac{\beta}{\varsigma}-s)+\frac{\sigma}{p}\,\frac{\beta}{\varsigma}.$$

Se observă deci că această limită există și este finită; valoarea acestei limite reprezintă asimptota funcției DPi.

2.6.1.3. Proprietatea 3: Numărul de instrucțiuni executate de program între defecte succesive de pagină crește cu numărul de pagini acumulate de program.

Din expresia (53) se caracterizează numărul mediu al defectelor de pagină generate în cadrul execuției unui program cu dimensiunea de P instrucțiuni. Ca și în cadrul analizei proprietății (1) a funcției DPi, vom presupune că P este un multiplu întreg de dimensiunea paginii.

Fie P_{i-1}, P_i și P_{i+1} valorile dimensiunii P de program ce conduc la apariția a unui număr de (i-1), i și respectiv (i+1) defecte de pagină. Forma pe care o ia expresia (53) în cele trei cazuri va fi:

$$\begin{aligned}
i-1 &= k_1+k_2\psi(\frac{1}{P_{i-1}})+k_3\phi(\frac{1}{P_{i-1}})\int_0^{P_{i-1}} x^{\beta-1}e^{-\varsigma x}dx\\
i &= k_1+k_2\psi(\frac{1}{P_i})+k_3\phi(\frac{1}{P_i})\int_0^{P_i} x^{\beta-1}e^{-\varsigma x}dx \qquad (73)\\
i+1 &= k_1+k_2\psi(\frac{1}{P_{i+1}})+k_3\phi(\frac{1}{P_{i+1}})\int_0^{P_{i+1}} x^{\beta-1}e^{-\varsigma x}dx
\end{aligned}$$

unde k_1, k_2 și k_3 sînt expresii cu valori constante în cele trei relații (73), iar $\psi(\frac{1}{P})$ și $\phi(\frac{1}{P})$ sînt funcții de $\frac{1}{P}$, obținute prin intermediul coeficienților a-f (51).

Scăzînd, între ele, primele și ultimele două egalități (73) se obține:

$$k_2\left[\psi\left(\frac{1}{P_i}\right)-\psi\left(\frac{1}{P_{i-1}}\right)\right]\ldots c_2\left[\phi\left(\frac{1}{P_i}\right)\int_0^{P_i}x^{\beta-1}e^{-\gamma x}dx-\phi\left(\frac{1}{P_{i-1}}\right)\int_0^{P_{i-1}}x^{\beta-1}e^{-\gamma x}dx\right]$$

$$k_2\left[\psi\left(\frac{1}{P_{i+1}}\right)-\psi\left(\frac{1}{P_i}\right)\right]+\ldots c_2\left[\phi\left(\frac{1}{P_{i+1}}\right)\int_0^{P_{i+1}}x^{\beta-1}e^{-\gamma x}dx-\phi\left(\frac{1}{P_i}\right)\int_0^{P_i}x^{\beta-1}e^{-\gamma x}dx\right]=1 \qquad (74)$$

Ţinînd seama că $\psi\left(\frac{1}{P_i}\right)$ şi $\phi\left(\frac{1}{P_i}\right)$ sînt funcţii de $\frac{1}{P_i}$ diferenţele între două valori ale acestor funcţii pentru argumentele (i-1;i) şi (i; i+1) vor conţine în termeni de forma:

$$\sum_j\left(\frac{k_j}{P_{i+1}^{\gamma}}-\frac{k_j}{P_i^{\gamma}}\right)=\sum_j k_j\left(\frac{1}{P_{i+1}^{\gamma}}-\frac{1}{P_i^{\gamma}}\right) \qquad (75)$$

Deoarece: $P_{i+1}=P_i+\Delta P_i$, expresia (82) se poate scrie:

$$\sum_j k_j\,\frac{P_i^{\gamma}-(P_i+\Delta P_i)^{\gamma}}{P_i^{\gamma}\cdot P_{i+1}^{\gamma}}\approx\sum_j k_j\,\frac{(-\gamma P_i^{\gamma-1})\Delta P_i}{P_i^{2\gamma}}=-\sum_j k_j\,\frac{\Delta P_i}{P_i^{\gamma+1}}$$

Pentru dimensiuni P suficient de mari ale programului, aceşti termeni sînt foarte mici şi ca atare se poate aproxima, pentru P_i mari:

$$\psi\left(\frac{1}{P_{i-1}}\right)\approx\psi\left(\frac{1}{P_i}\right)\approx\psi\left(\frac{1}{P_{i+1}}\right)$$

$$\phi\left(\frac{1}{P_{i-1}}\right)\approx\phi\left(\frac{1}{P_i}\right)\approx\phi\left(\frac{1}{P_{i+1}}\right) \qquad (76)$$

Scăzînd acum între ele egalităţile (74) şi utilizînd (76) se obţine relaţia:

$$\int_{P_{i-1}}^{P_i}x^{\beta-1}e^{-\gamma x}dx=\int_{P_i}^{P_{i+1}}x^{\beta-1}e^{-\gamma x}dx \qquad (77)$$

Fie: $f(x)=x^{\beta-1}e^{-\gamma x}dx$

iar $f(x_i)$ şi $f(x_{i+1})$ valorile medii ale acestei funcţii pe intervalele:

$[P_{i-1};\ P_i]$

$[P_i;\ P_{i+1}]$ respectiv

Utilizînd pentru calculul integralelor din relaţia (77) o formulă de medie, obţinem egalitatea:

$$(P_i-P_{i-1})\,f(x_i)=(P_{i+1}-P_i)\,f(x_{i+1}) \qquad (78)$$

./.

Derivînd-ne pe rezultatul derivatei

$$f'(x) = \frac{\rho^{-2} e^{-\varphi x}}{x} \left[(\rho - 1) - \varphi x \right]$$

care pentru $x > \frac{\rho - 1}{\varphi}$ este negativă, rezultă că funcţia $f(x)$ este descrescătoare pentru valori ale lui P mai mari ca $P_0 = \frac{\rho - 1}{\varphi} \approx$ valoarea medie a lungimii ciclurilor programului.

Valorile P_i considerate de noi se referă la dimensiuni ale programului ce satisfac $P > P_0$. În această situaţie, valoarea medie a $f(x)$ pe intervalul (P_{i-1}, P_i) este mai mare ca valoarea medie a $f(x)$ pe intervalul (P_i, P_{i+1})

$$f(x_i) > f(x_{i+1}) \qquad (79)$$

Ţinînd seama de (79), din (78) rezultă că:

$$P_{i+1} - P_i > P_i - P_{i-1}$$

relaţie ce atestă că numărul de instrucţii executate de program între defectele succesive de paginã (i,i+1) este mai mare decît numărul de instrucţii executate de program între defectele succesive de paginã (i-1,i); deci acest număr de instrucţii creşte cu numărul de pagini acumulate de program.

2.6.2. Verificãri experimentale.

Rezultate experimentale privind comportarea unor programe într-un mediu cu memorie virtualã au fost publicate în numeroase studii (40,72,75,76,102,111).

Vom stabili în cele ce urmează în ce mãsurã rezultatele publicate în aceste lucrãri, ce se referã la numărul de defecte de paginã ce sînt generate în cadrul execuţiei unui program cu dimensiunea de P instrucţii, pot fi regãsite cu ajutorul modelului propus al comportãrii de acces a programului.

Vom descrie procedura de lucru în acest scop, pentru un exemplu de program luat din (72), urmînd ca rezultatele obţinute prin această procedurã pentru alte programe sã fie cuprinse într-un tabel (tabelul 5).

Astfel, pentru programul considerat:

p = 2 kbyte

P = 10 p = 20 kbyte

$L(\alpha) = 12 \text{ K} = 0{,}6 \text{ P}$

$$D^2(\alpha) = \sum_{1}^{n} \frac{[\alpha_i - m(\alpha)]^2 \cdot n_i}{n} = 0{,}021 \text{ P}^2$$

./.

$$\beta = \frac{V(\alpha)}{D^2(\alpha)} = \frac{0,36\ n^2}{0,021\ n^2} \approx 18$$

$$\varsigma = \frac{V(\alpha)}{D^2(\alpha)} = \frac{0,6\ n}{0,021\ n^2} \approx \frac{20}{n} \qquad (30)$$

$s = 10\ K = 0,5\ I$

$\sigma = 20$

Pentru comoditatea calculelor ce urmează, vom adopta dimensiunea de pagină p ca unitate de referinţă pentru mărimile calculate.

Coeficienţii a - f (31) au valorile:

$$a = \frac{1}{20}$$

$$b = \frac{0}{20}$$

$$c = \frac{950}{10}$$

$$d = - 301$$

$$e = 20 \qquad (31)$$

$$f = - \frac{3}{2}$$

Pentru determinarea numărului mediu de defecte de pagină cu relaţia (30) se impune calculul funcţiilor Γ incomplete $I[x,\beta]$, x şi β fiind întregi, anume:

$$I[x,\beta] = \frac{1}{\Gamma(\beta)} \int_0^x y^{\beta-1} e^{-y} dy$$

Integrînd prin părţi:

$y^{\beta-1} = u$ $\qquad$ $e^{-y}dy = dv$

$du = (\beta-1)\ y^{\beta-2}\ dy$ $\qquad$ $v = - e^{-y}$

$$I[x;\beta] = \frac{1}{\Gamma(\beta)} \left| -y^{\beta-1} e^{-y} \right|_0^x + \frac{1}{\Gamma(\beta)} (\beta-1) \int_0^x y^{\beta-2} e^{-y} dy =$$

$$= - \frac{1}{\Gamma(\beta)} x^{\beta-1} e^{-x} + \frac{1}{\Gamma(\beta-1)} \int_0^x y^{\beta-2} e^{-y} dy =$$

$$= - \frac{1}{\Gamma(\beta)} x^{\beta-1} e^{-x} + I[x; \beta-1] \qquad (32)$$

./.

Scriind această relaţie de recurenţă pentru β descrescători:

$$I[x;\beta-1] = -\frac{1}{\Gamma(\beta-1)} x^{\beta-2} e^{-x} + I[x;\beta-2] \qquad (83)$$

$$I[x,3] = -\frac{1}{\Gamma(3)} x^2 e^{-x} + I[x,2] \qquad (84)$$

$$I[x,2] = -\frac{1}{\Gamma(2)} xe^{-x} + I[x,1] = -\frac{1}{\Gamma(2)} xe^{-x} + \frac{1}{\Gamma(1)} \int_0^x e^{-y}dy =$$

$$= -\frac{1}{\Gamma(2)} xe^{-x} + \left|-e^{-y}\right|_0^x = -\frac{1}{\Gamma(2)} xe^{-x} - e^{-x} + 1 \qquad (85)$$

Insumînd relaţiile (82-85):

$$I[x;\beta] = -\frac{1}{\Gamma(\beta)} x^{\beta-1}e^{-x} - \frac{1}{\Gamma(\beta-1)} x^{\beta-2}e^{-x} - \ldots - \frac{1}{\Gamma(3)} x^2e^{-x} -$$

$$- \frac{1}{\Gamma(2)} xe^{-x} - e^{-x} + 1 = 1 - e^{-x}\left[\frac{x^{\beta-1}}{\Gamma(\beta)} + \frac{x^{\beta-2}}{\Gamma(\beta-1)} + \ldots + \frac{x^2}{\Gamma(3)} + \frac{x}{\Gamma(2)} + 1\right]$$

$$I[x;\beta] = 1 - e^{-x}\left[1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^{\beta-1}}{(\beta-1)!}\right] \qquad (86)$$

Valorile numerice ale expresiei (86) pot fi foarte comod calculate cu ajutorul unui calculator. In cazul programului cercetat s-au determinat cu ajutorul programului GAMINC următoarele valori pentru funcţiile Γ incomplete $I[x;\beta]$, valori trecute în tabelul 4.

Tabelul 4

$x \backslash \beta$	18	19	20
12	0,0630	0,0374	0,0213
15	0,251	0,181	0,125
30	0,993	0,987	0,978

Expresia (50) devine, ţinînd seama de (80) şi (81):

./.

$$DPA = \frac{1}{20} \cdot \frac{18 \cdot 19}{9} I\left[12;20\right] + \frac{9}{20} \cdot \frac{18}{3} I\left[12;19\right] +$$

$$+ \frac{959}{10} \cdot \frac{18}{3} \left\{ I\left[15;19\right] - I\left[12;19\right] \right\} -$$

$$- 381 \left\{ I\left[15;18\right] - I\left[12;18\right] \right\} +$$

$$+ 20 \cdot \frac{18}{3} \left\{ I\left[30;19\right] - I\left[15;19\right] \right\} -$$

$$- \frac{3}{2} \left\{ I\left[30;18\right] - I\left[15;18\right] \right\} = 108$$

Numărul de defecte de pagină raportat în (72) pentru acest program este 105, deci se poate constata o bună concordanţă între rezultatul obţinut cu ajutorul modelului şi cel experimentat.

Tabelul 5 cuprinde rezultate relative la numărul de defecte de pagină generate în cadrul execuţiei unui program: coloana (e) se referă la rezultate experimentale comunicate în lucrările amintite, coloana (m) se referă la rezultatele obţinute prin utilizarea modelului propus de comportare al programului.

Tabelul 5

Dimensiunea paginii	Dimensiunea programului	Alocarea	e	m	Datele experimentale din lucrarea
4 K	517 p	76 p	1.807	1.821	
		197 p	820	812	(93)
		512 p	517	523	
32 K	166 p	32 p	598	605	
		64 p	380	368	(70)

Pe baza rezultatelor din tabelul 5 se poate trage concluzia că modelul propus dă rezultate corecte.

CAP.5. STUDIUL ANOMALIEI DIMENSIUNII DE PAGINA

1. Cercetarea anomaliei dimensiunii de pagină cu ajutorul modelului elaborat

Fie expresia ce determină numărul mediu de defecte de pagină, care sînt generate în cadrul execuţiei unui program.

$$DPA = a\frac{\beta(\beta+1)}{\varsigma^2} I\left[\varsigma(s-p);\ \beta+2\right] + b\cdot\frac{\beta}{\varsigma} I\left[\varsigma(s-p);\ \beta+1\right] +$$
$$+ c\cdot\frac{\beta}{\varsigma}\left\{I\left[\varsigma s;\ \beta+1\right] - I\left[\varsigma(s-p);\ \beta+1\right]\right\} +$$
$$+ d\left\{I\left[\varsigma s;\beta\right] - I\left[\varsigma(s-p);\beta\right]\right\} +$$
$$+ e\frac{\beta}{\varsigma}\left\{I\left[\varsigma p;\beta+1\right] - I\left[\varsigma s;\ \beta+1\right]\right\} +$$
$$+ f\left\{I\left[\varsigma p;\beta\right] - I\left[\varsigma s;\beta\right]\right\} \qquad (87)$$

Această expresie este funcţie de dimensiunea de pagină p utilizată, prin intermediul coeficienţilor a-f:

$$a = \frac{1}{2Pp};$$

$$b = \frac{1}{p} - \frac{s}{Pp} - \frac{1}{2P};$$

$$c = \frac{s}{p^2}(\sigma-1) + \frac{1}{p} - \frac{1}{P};$$

$$d = -(\sigma-1)\frac{s^2}{p^2} + \frac{s}{p}\left(\sigma-1-\frac{s}{2P}\right) + \frac{s}{2P};$$

$$e = \frac{\sigma}{p};$$

$$f = -\frac{s^2}{2Pp} - \frac{s}{2P} \qquad (88)$$

In cadrul acestor expresii, ca şi în capitolul precedent, secţiunea 2.6.1., s-a presupus - fără a afecta generalitatea rezultatelor - că dimensiunea programului P este multiplu întreg de dimensiunea paginii p. Notaţiile folosite în cadrul expresiilor (87) şi (88) sînt cele din cap.4.

Cercetarea cu ajutorul modelului elaborat al comportării de acces a unui program a anomaliei dimensiunii de pagină constă în determinarea cu ajutorul expresiei (87) a numărului de defecte de pagină care apar în cadrul execuţiei acestui program şi uti-

./.

lizării succesive a două dimensiuni de pagină, p şi respectiv $\frac{p}{k}$ (de k ori mai mică) şi apoi analiza raportului numerelor acestor defecte:

$$\frac{DPA\ (\frac{p}{k})}{k\ DPA(p)}$$

sau, ceea ce este similar, analiza semnului diferenţei:

$$DIF = DPA\ (\frac{p}{k}) - k\ DPA\ (p) \qquad (89)$$

Situaţia de anomalic a dimensiunii de pagină va apare în cazul în care numărul defectelor de pagină generate la execuţia programului şi utilizarea unei dimensiuni de pagină $\frac{p}{k}$ va fi de peste k ori mai mare decît numărul defectelor de pagină generate la execuţia aceluiaşi program şi utilizarea unei dimensiuni de pagină p, adică în cazul în care:

$$DIF > 0$$

Practic, pentru a determina valorile DPA $(\frac{p}{k})$ şi DPA (p) ale expresiei (87), avem nevoie de valorile coeficienţilor a-f (88) în cazul utilizării celor două dimensiuni de pagină.

De altfel, (88) reprezintă valorile a-f pentru dimensiunea de pagină p. In cazul dimensiunii de pagină $\frac{p}{k}$ valorile acestor coeficienţi devin:

$$a^x = k \cdot \frac{1}{2\ Pp} \ ;$$

$$b^x = k\ (\frac{1}{p} - \frac{s}{Pp}) - \frac{1}{2P} \ ;$$

$$c^x = k^2 \cdot \frac{s(\sigma-1)}{p^2} + k \cdot \frac{1}{p} - \frac{1}{P} \ ; \qquad (90)$$

$$d^x = -k^2(\sigma-1)\frac{s^2}{p^2} + k \cdot \frac{s}{p}\ (\sigma - 1 - \frac{s}{2P}) + \frac{s}{2P} \ ;$$

$$e^x = k \cdot \frac{\sigma}{p} \ ;$$

$$f^x = -k \cdot \frac{s^2}{2\ Pp} - \frac{s}{2P}$$

Pe baza valorilor (88) şi (90) se poate trece la determinarea diferenţei (89):

./.

$$DIF = \frac{\beta}{\varsigma} I\left[\varsigma(s-p);\beta+1\right]\left(k\frac{1}{p} - k\frac{s}{Pp} - \frac{1}{2P} - k\frac{1}{p} + k\frac{s}{Pp} + k\frac{1}{2P}\right) +$$

$$+ \frac{\beta}{\varsigma}\left\{I\left[\varsigma s;\beta+1\right] - I\left[\varsigma(s-p);\beta+1\right]\right\}\left(k^2 s\frac{\sigma-1}{p^2} + k\frac{1}{p} - \frac{1}{P} - \right.$$

$$\left. - ks\frac{\sigma-1}{p^2} - k\frac{1}{p} + k\frac{1}{P}\right) + \left\{I\left[\varsigma s;\beta\right] - I\left[\varsigma(s-p);\beta\right]\right.\cdot$$

$$\cdot\left[-k^2s^2\frac{\sigma-1}{p^2} + ks\frac{1}{p}\left(\sigma-1-\frac{s}{2P}\right) + \frac{s}{2P}\right. +$$

$$\left. + ks^2\frac{\sigma-1}{p^2} - ks\frac{1}{p}\left(\sigma-1-\frac{s}{2P}\right) - k\frac{s}{2P}\right] +$$

$$+\left\{I\left[\varsigma P;\beta\right] - I\left[\varsigma s;\beta\right]\right\}\left(-k\frac{s^2}{2Pp} - \frac{s}{2P} + k\frac{s^2}{2Pp} + k\frac{s}{2P}\right)$$

Efectuînd calculul parantezelor se obţine:

$$DIF = \frac{\beta}{\varsigma} I\left[\varsigma(s-p);\beta+1\right]\cdot\frac{1}{2P}(k-1) + \frac{\beta}{\varsigma}\left\{I\left[\varsigma s;\beta+1\right] -\right.$$

$$\left. - I\left[\varsigma(s-p);\beta+1\right]\right\}(k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{P}\right] - \left\{I\left[\varsigma s;\beta\right] - I\left[\varsigma(s-p);\beta\right]\right\}\cdot$$

$$\cdot\, s(k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{2P}\right] + \left\{I\left[\varsigma P;\beta\right] - I\left[\varsigma s;\beta\right]\right\}\frac{s}{2P}(k-1)$$

sau o formă identică:

$$DIF = -\frac{\beta}{\varsigma} I\left[\varsigma(s-p);\beta+1\right](k-1)\left[ks\frac{-1}{p^2} + \frac{1}{2P}\right] +$$

$$+ I\left[\varsigma(s-p);\beta\right]s(k-1)\,ks\left[\frac{\sigma-1}{p^2} + \frac{1}{2P}\right] + \frac{\beta}{\varsigma} I\left[\varsigma s;\beta+1\right](k-1)\cdot$$

$$\cdot\left[ks\frac{\sigma-1}{p^2} + \frac{1}{P}\right] - I\left[\varsigma s;\beta\right]s(k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{P}\right] +$$

$$+ I\left[\varsigma P;\beta\right]\frac{s}{2P}(k-1) \qquad (91)$$

Funcţia Γ incompletă $I\left[\varsigma(s-p);\beta+1\right]$

$$I\left[\varsigma(s-p);\beta+1\right] = \frac{1}{\Gamma(\beta+1)}\int_{0}^{\varsigma(s-p)} x^{(\beta+1)-1}\,e^{-x}\,dx$$

poate fi transformată, prin integrare prin părţi, astfel:

$$u = x^{\beta} \qquad dv = e^{-x}dx$$

$$du = \beta x^{\beta-1}dx \qquad v = -e^{-x}$$

v/.

$$I\left[\varsigma(s-p;\beta+1\right] = \frac{1}{\Gamma(\beta+1)}\left|- x^{\beta} e^{-x}\right|_{0}^{(s-p)} + \frac{\beta}{\Gamma(\beta+1)}\int_{0}^{\varsigma(s-p)} x^{\beta-1} e^{-x} dx =$$

$$= -\frac{1}{\Gamma(\beta+1)}\left[\varsigma(s-p)\right]^{\beta} e^{-\varsigma(s-p)} + I\left[\varsigma(s-p);\beta\right] \quad (92)$$

In mod asemănător, rezultă:

$$I\left[\varsigma s;\beta+1\right] = -\frac{1}{\Gamma(\beta+1)}\left[\varsigma s\right]^{\beta} e^{-\varsigma s} + I\left[\varsigma s;\beta\right] \quad (93)$$

Inlocuind (92) şi (93) în expresia (91) se obţine:

$$DIF = \frac{\beta}{\varsigma}\frac{1}{\Gamma(\beta+1)}\left[\varsigma(s-p)\right]^{\beta} e^{-\varsigma(s-p)}(k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{2P}\right] -$$

$$- \frac{\beta}{\varsigma}\frac{1}{\Gamma(\beta+1)}\left[\varsigma s\right]^{\beta} e^{-\varsigma s}(k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{P}\right] +$$

$$+ I\left[\varsigma(s-p);\beta\right](k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{2P}\right](s - \frac{\beta}{\varsigma}) -$$

$$- I\left[\varsigma s;\beta\right](k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{P}\right](s - \frac{\beta}{\varsigma}) + I\left[\varsigma P;\beta\right]\frac{s}{2P}(k-1) \quad (94)$$

Notăm cu:

$$A = \frac{1}{\Gamma(\beta)}\ \frac{1}{\varsigma}\left[\varsigma(s-p)\right]^{\beta} e^{-\varsigma(s-p)}$$

$$B = \frac{1}{\Gamma(\beta)}\ \frac{1}{\varsigma}\left[\varsigma s\right]^{\beta} e^{-\varsigma s}$$

$$C = I\left[\varsigma(s-p);\beta\right] \quad (95)$$

$$D = I\left[\varsigma s;\beta\right]$$

$$E = I\left[\varsigma P;\beta\right]$$

Cu notaţiile (95) expresia (94) devine:

$$DIF = A(k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{2P}\right] - B(k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{P}\right] +$$

$$+ C(k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{2P}\right](s - \frac{\beta}{\varsigma}) - D(k-1)\left[ks\frac{\sigma-1}{p^2} + \frac{1}{P}\right](s - \frac{\beta}{\varsigma}) +$$

$$+ E\frac{s}{2P}(k-1)$$

Sau, organizînd în partea dreaptă:

./.

$$DIF = \frac{k-1}{2P}\left[A-2B+(s-\frac{\beta}{\varsigma})(C-2D)+E.s\right]-(k-1)ks\frac{\varsigma-1}{p^2}\left[-A+B+(s-\frac{\beta}{\varsigma})(-C+D)\right] \quad (96)$$

Expresia relativ simplă (96) obţinută pentru valoarea DIF va fi cercetată în continuare pentru precizarea posibilităţilor de apariţie a anomaliei dimensiunii de pagină.

In cele ce urmează ne vom limita la cazul în care $s > \frac{\beta}{\varsigma}$ adică dimensiunea memoriei reale alocate pentru rularea programului este mai mare decît lungimea medie a ciclurilor din program. Este o limitare firească, ce decurge din practica curentă a sistemelor cu memorii virtuale, în care, pentru a menţine activitatea de paginare în limite moderate, un program este eligibil de a fi rulat numai în momentul în care există posibilitatea de a i se aloca suficient spaţiu de memorie reală. In cazul nostru această "suficienţă" este caracterizată de lungimea medie a ciclurilor programului în discuţie. O discuţie mai în detaliu a acestor probleme a fost făcută în cap.2 secţiunea 6.1.1. relativă la sistemele cu memorie virtuală paginată şi factorii ce afectează performanţa unui asemenea sistem.

Pentru stabilirea valorilor mărimilor A şi B cercetăm variaţia funcţiei:

$$f(x) = \frac{1}{\Gamma(\beta)}\ \frac{1}{\varsigma}\ x^{\beta}e^{-x}$$

Prima derivată a acestei funcţii:

$$f'(x) = \frac{1}{\Gamma(\beta)}\ \frac{1}{\varsigma}\ x^{\beta-1}\ e^{-x}(\beta-x)$$

se anulează în punctul $x = \beta$, punct de maxim pentru f(x).

Intrucît:

$$A = f\left[\varsigma(s-p)\right] ;$$

$$B = f\left[\varsigma s\right]$$

iar ambele argumente fiind mai mari ca β ($\varsigma s > \beta$ sau $s > \frac{\beta}{\varsigma}$, condiţie îndeplinită în situaţia în care ne aflăm), rezultă că

$$1 > A > B$$

De asemenea, conform celor arătate în cap.4,

$$C < D < E < 1$$

Inegalităţile rezultă din relaţia de definiţie a funcţiei Γ incomplete şi a valorii β

$$\beta > 1$$

./.

Expresia (96) se mai poate scrie:

$$DIF = \frac{k-1}{2P}\left[A - 2B + \frac{\beta}{\varsigma}(2D-C) + s(C - 2D + E)\right] - \\ - (k-1)\,ks\,\frac{\sigma - 1}{p^2}\left[-A + B + (s - \frac{\beta}{\varsigma})(-C + D)\right] \qquad (97)$$

Notînd cu:

$$\lambda_1 = \frac{k-1}{2P}\left[A - 2B + \frac{\beta}{\varsigma}(2D - C) + s(C - 2D + E)\right]$$
$$\lambda_2 = (k-1)\,ks\,\frac{1}{p^2}\left[-A+B+(s - \frac{\beta}{\varsigma})(-C+D)\right] \qquad (98)$$

şi ţinînd seama că, conform celor arătate

$$\lambda_1 > 0$$
$$\lambda_2 > 0$$

expresia DIF devine:

$$DIF = \lambda_1 - (\sigma - 1)\lambda_2 \qquad (99)$$

Ea atinge valori pozitive (deci are loc situaţia de anomalie a dimensiunii de pagină) în cazul în care:

$$\sigma < \frac{\lambda_1}{\lambda_2} + 1 \qquad (100)$$

Valoarea maximă pozitivă a DIF este atinsă în situaţia în care $\sigma = 1$.

Rezultatul obţinut arată că apariţia situaţiei de anomalie a dimensiunii de pagină este posibilă în cazul acelor programe sau secţiuni de program cu valori foarte mici ale parametrului σ, ce satisfac (100).

Pentru a interpreta sensul rezultatului analitic obţinut vom cerceta mai în detaliu comportarea localizată a unui program.

2. Comportarea localizată a unui program.

2.1. Definiţie şi justificare.

In decursul întregii lucrări am presupus că localizarea este o proprietate fundamentală a comportării de acces a unui program. Localizarea este proprietatea ca, în decursul oricărui interval de execuţie, programul va favoriza unele din paginile

./.

lui într-o măsură mai mare decît pe altele; în decursul unor intervale distincte de execuţie grupul de pagini favorizate poate fi diferit. Cu alte cuvinte, observînd mostrele de referire la adresele logice ale unui program într-un interval oarecare de timp se poate constata că programul nu-şi distribuie referirile în mod uniform în întreg spaţiu al adreselor. Există cel puţin cinci factori ce motivează această presupunere:

1. fluxuri de instrucţii secvenţiale.

Atît programatorii, cît şi compilatoarele manifestă tendinţa să organizeze în mod secvenţial instrucţiile ce direcţionează activitatea unui program; acest lucru este în special valabil în cazul calculatoarelor cu instrucţii cu o singură adresă. Dacă un program aduce o instrucţie dintr-o pagină dată, este foarte probabil ca el să aducă în continuare instrucţia imediat următoare, din aceeaşi pagină.

2. modularitatea funcţională.

Modulele de program sînt frecvent organizate şi executate printr-o funcţie.

3. organizarea datelor înrudite dupa conţinut.

Informaţiile sînt în mod obişnuit grupate după conţinut în segmente şi sînt în mod normal referite pe această cale. Astfel, referirile programului în spaţiul adreselor vor apărea sub formă de pachete, un pachet de referiri corespunzînd domeniului din spaţiul adreselor ce conţine informaţii înrudite după conţinut.

4. ciclurile.

Programele prezintă frecvent cicluri, ce se pot extinde asupra unui grup de pagini.

5. modul de programare.

Dîndu-şi seama că programele pe care le scriu vor rula într-un mediu virtual şi că transferurile de pagini sînt costisitoare, programatorii îşi organizează algoritmii lor astfel încît activitatea programului să fie localizată în cadrul unor submulţimi ale informaţiilor. Acest aspect a fost tratat mai extins la cap.2, în secţiunea 6.1.1.2. relativă la căile de îmbunătăţire a performanţei sistemului într-un mediu cu memorie virtuală.

Observaţiile experimentale confirmă că presupunerea privind comportarea localizată a mostrei de referiri a unui program este corectă.

Presupunem, în cazul contrar, că, în timpul fiecărui interval de timp virtual, programul îşi distribuie referirile lui la spaţiul adreselor în mod uniform. Presupunem că o fracţiune $s(0 \leq s \leq 1)$ a paginilor lui a fost dispusă în memoria principală. Fie $\mu(s)$ fracţiunea din referirile programului, ce se fac asupra grupului de pagini ce nu sînt plasate în memorie; deoarece referirile sînt distribuite uniform, rezultă că:

$$\mu(s) = 1 - s$$

Funcţia $\mu(s)$ determinată experimental (18,59) şi ilustrată în fig.35 contrazice ipoteza făcută. S-a stabilit că în realitate $\mu(s)$ se poate reprezenta printr-o curbă, situată sub graficul $\mu(s) = 1 - s$.

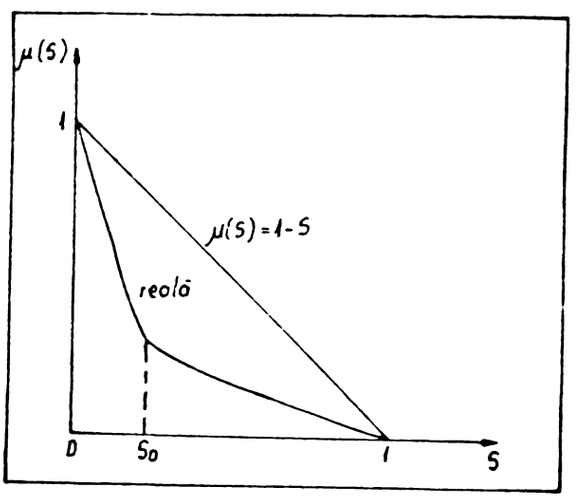


Fig.35. Funcţia $\mu(s)$

A fost observat că există un număr s_0 şi o constantă $k > 1$, astfel că, dacă $s \leq s_0$, atunci:

$$\mu(s) = 1 - ks$$

adică programul îşi distribuie referirile lui în mod uniform numai asupra unui subgrup al informaţiilor lui. Numerele s_0 şi k depind de programul particular şi de algoritmul concret de decizie utilizat pentru a stabili ce informaţii trebuie să stea în memoria principală.

Pe baza celor arătate mai sus se poate aprecia că localizarea este o proprietate a comportării de acces a programului.

./.

2.2. Tipuri de localizare a comportării de acces a programului.

Se pot evidenţia două forme extreme ale comportării localizate a unui program în privinţa modului cum face referiri la spaţiul adreselor: localizare temporală şi localizare spaţială.

(a) localizare temporală.

Dacă adresele logice (a_1,a_2,...) sînt referite în decursul intervalului de timp (t-T,t) atunci există o probabilitate mare ca aceleaşi adrese logice să fie referite în decursul intervalului (t, t+T). Această comportare poate fi explicată prin construcţii de program precum cicluri, variabile frecvent utilizate şi subrutine frecvent utilizate;

(b) localizare spaţială.

Dacă adresa logică a este referită la momentul t, atunci există o probabilitate mare ca o adresă logică în domeniul (a-A, a+A) să fie referită la momentul t+1. Această comportare poate fi explicată prin construcţii de program, precum o ordonare secvenţială a instrucţiilor şi structuri lineare de date (ex.matrici).

Aceste două tipuri de comportare localizată a programelor - temporală şi spaţială - sînt aspectele de bază, de extremă, ce se pot remarca în modul cum îşi distribuie un program referirile la spaţiul adreselor. Aceste concepte se pot îmbina pentru a defini localizarea unui program la modul general, spaţio-temporal, în modul următor:

Dacă adresele logice (a_1,a_2,...) sînt referite în decursul intervalului de timp (t-T,t), atunci există o probabilitate mare ca adresele logice din domeniile [(a_1-A, a_1+A), (a_2-A, a_2+A)...] să fie referite în decursul intervalului de timp (t, t+T).

2.3. Localizarea şi algoritmii de înlocuire.

Putem începe a înţelege factorii ce cauzează anomalia dimensiunii de pagină prin studierea modului cum diferiţi algoritmi de înlocuire convenţionali (vezi secţiunea 3.4 a cap.2) tratează localizarea temporală şi spaţială.

Astfel putem constata că, în timp ce măsurilor ce vizează localizarea temporală a programelor li se acordă o atenţie explicită, măsurile ce se referă la localizarea spaţială sînt de obicei

./.

tratate implicit. De exemplu, algoritmul LRU (cel mai puţin recent utilizat) se preocupă foarte mult de aspectele temporale ale mostrei de referiri a programului la spaţiul adreselor. Aspectele spaţiale sînt tratate ca un produs secundar, prin faptul că algoritmul de înlocuire aduce în memoria principală, la un moment dat, o pagină întreagă (adică o regiune spaţială), iar deciziile de înlocuire LRU sînt bazate pe utilizarea acestor pagini.

Ţinînd seama de acest mod în care sînt luate în consideraţie localizarea temporală şi spaţială a programelor de către algoritmii de înlocuire uzuali, se poate face următoarea observaţie.

Micşorarea dimensiunii de pagină conduce la creşterea performanţei (mărirea valorii proporţiei succeselor) algoritmilor de înlocuire în cazul unui program caracterizat printr-o comportare localizată pronunţat temporală şi la scăderea performanţei (micşorarea valorii proporţiei succeselor) în cazul unui program caracterizat printr-o comportare localizata pronunţat spaţială. Acest lucru este o urmare a faptului că, prin micşorarea dimensiunii de pagină (în cadrul unei alocări date în M_1), în nivelul M_1 al ILVM se va realiza o colecţie mai variată de porţiuni de program, deşi fiecare din ele mai mică ca dimensiune. Acest tip de colecţie poate favoriza un program cu referiri dispersate şi grupate la spaţiul adreselor, ca urmare a unui caracter temporal al comportării sale de acces. Pe de altă parte însă, micşorarea dimensiunii de pagină defavorizează un program ce are o comportare de localizare spaţială în modul de a-şi referi adresele, căci prin micşorarea dimensiunii paginii se micşorează şi numărul de adrese ce pot fi referite succesiv în cadrul aceleiaşi pagini.

Este de dorit ca algoritmii de înlocuire utilizaţi să ia în consideraţie ambele localizari - temporală şi spaţială.

Modul în care un program sau o secţiune a unui program îşi distribuie referirile la spaţiul de adrese poate fi caracterizat prin următoarea matrice a localizărilor 2 x 2:

./.

		Localizare temporală	
		slabă	pronunţată
Localizare spaţială	slabă	1	2
	pronunţată	3	4

Domeniul 1 caracterizat printr-o slabă localizare temporală şi slabă localizare spaţială este în mod evident nedorit ca situaţie într-un mediu cu memorii virtuale. După cum s-a arătat în secţiunea 6.1.1.2. a cap.2 există numeroşi algoritmi şi tehnici pentru a îmbunătăţi slaba comportare de localizare a programelor. Domeniul 4 caracterizat printr-o pronunţată localizare spaţială şi pronunţată localizare temporală este domeniul cu performanţele cele mai bune şi reprezintă scopul unei bune proiectări de program. Nu este însă totdeauna posibil sau convenabil a proiecta programe care ating caracteristici pronunţate atît de localizare temporală cît şi de localizare spaţială. Astfel, există multe programe operînd în domeniile 2 şi 3.

In consecinţă, tehnicile de gestiune a ierarhiilor de memorii necesită algoritmi ce să prevadă o mai mare flexibilitate şi robusteţe pentru a ţine seama de sensibilitatea sistemului în privinţa localizării temporale şi spaţiale. Aceşti algoritmi trebuie să ia în considerare în mod explicit localizarea spaţială a unui program.

3. Interpretare cu caracter general a anomaliei dimensiunii de pagină.

In secţiunea 1 a acestui capitol s-a ajuns la rezultatul, conform căruia apariţia situaţiei de anomalie a dimensiunii de pagină este posibilă în cazul acelor programe sau secţiuni de program cu valori mici ale parametrului σ, ce satisfac (100). Parametrul σ reprezintă în cadrul modelului propus al comportării de acces a programului, numărul de ori de care este executat un ciclu cu lungimea α. O valoare mică a acestui parametru (deci apropiată de 1) este indicele unei slabe localizări temporale a programului sau secţiunii de program respective, conform determinării localizării temporale date în secţiunea 2.2 a acestui capitol.

./.

In cazul limită, în care anomalia de pagină apare în modul cel mai pronunţat (analitic vorbind, DIF din expresia (99) atinge o valoare maximă) valoarea lui σ este $\sigma = 1$, deci nu există cicluri în cadrul acelei secţiuni de program, instrucţiile executîndu-se strict secvenţial. Este evident cazul unei localizări spaţiale pronunţate a modului în care programul, în cadrul secţiunii lui respective, îşi distribuie referirile la spaţiul adreselor.

In consecinţă, anomalia dimensiunii de pagină poate apărea în cadrul unui program operînd într-un domeniu caracterizat printr-o pronunţată localizare spaţială şi o slabă localizare temporală (domeniul 3 în matricea localizărilor). Ea nu este o funcţie strictă de dimensiunea de pagină, ci este o consecinţă a modului în care programul îşi distribuie referirile lui la spaţiul adreselor şi anume a proprietăţilor de localizare spaţială şi temporală ale programului.

CAP.6. PREVENIREA APARIŢIEI ANOMALIEI DIMENSIUNII DE PAGINĂ

Cercetarea anomaliei dimensiunii de pagină, efectuată în cap.5., a evidenţiat faptul că apariţia acestei situaţii în cadrul unui sistem ierarhic de memorii este o consecinţă a comportării de acces localizate, pronunţat spaţială şi slab temporală, a programului. Această comportare localizată este însă o caracteristică globală, la nivel macroscopic, a modului cum programul îşi distribuie referirile la spaţiul adreselor.

Pentru a putea elabora măsuri care să conducă la posibilitatea prevenirii apariţiei acestei anomalii este necesară o cercetare mai detaliată a mecanismului prin care modul în care sînt făcute referirile programului în spaţiul adreselor conduce la apariţia anomaliei. Această analiză la nivel microscopic a comportării de acces a programului se referă la cercetarea şirului de referiri de pagină.

1. Cercetarea şirului de referiri de pagină.

Fie un sistem ierarhic de memorii cu două nivele, în cadrul căruia utilizăm succesiv pagini cu dimensiunea de p şi respectiv $\frac{p}{2}$ octeţi. Comportamentul de acces al programului la spaţiul adreselor este caracterizat prin cele două şiruri de referiri de pagină X_p şi $X_{\frac{p}{2}}$.

Fie un moment oarecare t_i la care are loc referirea la adresa logică a, ce corespunde referirii la pagina A (în cadrul şirului de referiri de pagină X_p) şi - respectiv - referirii la pagina A_1 (în cadrul şirului de referiri de pagină $X_{\frac{p}{2}}$). A_1 şi A_2 sînt cele două pagini de dimensiune $\frac{p}{2}$ octeţi ce alcătuiesc pagina A cu dimensiunea p octeţi.

Ca şi în cadrul cap.3, secţiunea 2, e_p şi $e_{\frac{p}{2}}$ reprezintă numărul defectelor de pagină obţinute în cadrul utilizării dimensiunii de pagină p şi respectiv $\frac{p}{2}$, iar ς este raportul acestor două numere:

$$\varsigma = \frac{e_{\frac{p}{2}}}{e_p}$$

In tratarea referirii la adresa logică a şi a utilizării

./.

celor două dimensiuni de pagină pot apărea următoarele 4 cazuri, indicate în tabelul 6 şi care sînt dependente de faptul dacă pagina referită A (în cazul utilizării dimensiunii p) sau A_1 (în cazul utilizării dimensiunii $\frac{p}{2}$) se găsesc sau nu în nivelul L_1 al ierarhiei de memorii.

Tabelul 6

Cazul	Pagina A se găseşte în L_1	Pagina A_1 se găseşte în L_1	Δe_p	$\Delta e_{\frac{p}{2}}$	$\Delta e_{\frac{p}{2}} - \Delta e_p$	Efect
1	da	da	0	0	0	$\varrho \to 1$
2	da	nu	0	1	1	$\varrho \to > 1$
3	nu	da	1	0	-1	$\varrho \to < 1$
4	nu	nu	1	1	0	$\varrho \to 1$

In cazurile 1 şi 4 se constată situaţii asemănătoare în cadrul utilizării celor două dimensiuni de pagină (în cazul 1 nu este necesară aducerea unei pagini, iar în cazul 4 este necesară aducerea unei pagini noi, deci are loc un defect de pagină); apariţia acestor situaţii determină pe ϱ să tindă spre 1.

In cazul 3, în cadrul utilizării dimensiunii de pagină p are loc un defect de pagină, iar în cadrul utilizării dimensiunii de pagină $\frac{p}{2}$ nu este cerută o pagină nouă; această situaţie, dacă apare frecvent, va cauza ca ϱ să fie mai mic ca 1. Acesta este rezultatul dorit prin reducerea dimensiunii de pagină.

Numai cazul 2, în care prin utilizarea dimensiunii de pagină $\frac{p}{2}$ are loc un defect de pagină, în timp ce prin utilizarea dimensiunii de pagină p nu este cerută o pagină nouă, contribuie la creşterea raportului ϱ .

Astfel, în scopul urmărit de noi, vom analiza mai amănunţit această ultimă situaţie.

Situaţia în discuţie implică faptul că pagina A să existe în memorie la momentul t_i. De aici rezultă că această

./.

pagină a fost adusă în M_1 printr-o referire anterioară. Prin cercetarea în sens invers a şirului de referiri de pagină X_p se poate identifica momentul t_j în care a avut loc referirea ce a condus la aducerea paginii A în M_1.

La momentul t_j a avut loc o referire şi în cazul şirului $X_{\frac{p}{2}}$, fie la A_1, fie la A_2. Vom considera separat aceste două cazuri.

Cazul 1: $t = \ldots, t_j, \ldots, t_i, \ldots$
$X_p = \ldots, A, \ldots, A, \ldots$
$X_{\frac{p}{2}} = \ldots, A_2, \ldots, A_1, \ldots$

In acest caz, la momentul t_j, în cazul utilizării dimensiunii de pagină $\frac{p}{2}$, a fost referită pagina A_2. Astfel, pentru a aduce în M_1 acelaşi volum de informaţii (p octeţi) în cazul utilizării dimensiunii de pagină de $\frac{p}{2}$ pot fi necesare două aduceri de pagină, faţă de una singură necesară în cazul utilizării dimensiunii de pagină p. Prin situaţii ce se caracterizează prin aceste şiruri de referiri ϱ nu ar putea depăşi valoarea 2.

Cazul 2: $t = \ldots, t_j, \ldots, t_i, \ldots$
$X_p = \ldots, A, \ldots, A, \ldots$
$X_{\frac{p}{2}} = \ldots, A_1, \ldots, A_1, \ldots$

In acest caz, urmare a referirii din momentul t_j, pagina A (în cadrul utilizării dimensiunii p) şi pagina A_1 (în cadrul utilizării dimensiunii $\frac{p}{2}$) se găsesc în M_1 din acest moment. Totuşi în momentul t_i pagina A se mai găseşte în M_1, în timp ce pagina A_1 nu mai este. In aceste circumstanţe, ϱ poate depăşi valoarea 2. De exemplu imaginînd că următoarea referire de pagină ar fi A_2, vom avea o situaţie în care au loc 3 aduceri de pagină în cazul utilizării dimensiunii $\frac{p}{2}$, faţă de o singură aducere de pagină în cazul utilizării dimensiunii p. Mai mult chiar, este posibil ca referirile situate între t_j şi t_i să fie repetate: acest lucru ar conduce la aducerea continuă a paginii A_1 în cazul unei dimensiuni de pagină $\frac{p}{2}$, fără nici o aducere corespunzătoare în cazul celeilalte dimensiuni de pagina.

Acest caz apare numai în situaţia în care între momentele t_j şi t_i, pagina A_1 (în cazul utilizării dimensiunii de pagina $\frac{p}{2}$), a fost înlocuită din M_1, în timp ce pagina A (în cazul utilizării

./.

dimensiunii de pagină p) rămîne în M_1. Cu alte cuvinte, această situaţie poate apărea numai dacă la un anumit moment t, pagina A_1 sau A_2 (în cazul dimensiunii de pagină $\frac{p}{2}$) este selectată pentru a fi înlocuită din M_1, în timp ce pagina corespunzătoare A (în cazul dimensiunii de pagină p) nu este înlocuită din M_1.

2. Modificarea algoritmilor de înlocuire uzuali pentru prevenirea apariţiei anomaliei dimensiunii de pagină.

Pe baza identificării situaţiilor ce conduc la apariţia anomaliei dimensiunii de pagină, identificare realizată în secţiunea precedentă, se poate propune o procedură ce conduce la posibilitatea prevenirii apariţiei acestei anomalii. Această procedură simplă constă în următoarele:

In primul rînd - paginile programului în cauză se organizează în grupe, mărimea cărora corespunde unei dimensiuni mărite de pagină (după numărul de pagini din grupă). O asemenea grupă o vom numi pagină-martor . Intr-un caz frecvent întîlnit în cele ce urmează, două pagini consecutive ale programului iniţial sînt considerate ca alcătuind o grupă. Dimensiunea acestei pagini-martor corespunde utilizării unei dimensiuni duble de pagină.

In al doilea rînd - algoritmii de înlocuire uzuali vor fi aplicaţi la ansamblul paginilor unei grupe (cu alte cuvinte aceste pagini sînt cuplate în privinţa deciziilor de ordonare), astfel ca o pagină oarecare din cadrul unei grupe nu poate să fie niciodată înlocuită din M_1, dacă pagina-martor corespunzătoare nu ar fi fost de asemenea înlocuită din M_1.

In termenii secţiunii precedente este împiedicată o situaţie în care o pagină a programului A_1 sau A_2 (ce formează împreună o grupă) să fie scoasă din M_1, în timp ce pagina-martor corespunzătoare A nu este scoasă din M_1.

In acest mod se evită posibilitatea apariţiei unei situaţii, caracterizată ca reprezentînd cazul 2 în tabelul 6 şi, conform celor expuse în secţiunea precedentă, se evită astfel posibilitatea ca raportul ξ să poată depăşi va-

./.

loarea 2. Astfel, se previne posibilitatea apariţiei anomaliei dimensiunii de pagină.

Modul de aplicare al acestei proceduri este exemplificat în continuare în cazul utilizării unor algoritmi de înlocuire cunoscuţi, dar modificaţi conform procedurii descrise.

3. Exemple de utilizare a unor algoritmi de înlocuire modificaţi.

Vom arăta în continuare modul în care se aplică procedura descrisă, în cadrul algoritmilor LRU şi FIFO modificaţi.

3.1. Algoritm LRU modificat.

In fig.36 este cercetat un şir de referiri de pagină, în cazul utilizării unor pagini de dimensiunea p şi respectiv $\frac{p}{2}$ octeţi şi folosirea ca algoritm de înlocuire atît a LRU, cît şi LRU modificat.

Parametrii sistemului, în cazul utilizării paginii p

- X_p = A,B,A,B,C,C,B,A,A,C,C
- λ = 11
- Γ = {A,B,C}
- η = 3
- m_1 = 2
- algoritm de înlocuire LRU

Parametrii sistemului, în cazul utilizării paginii $\frac{p}{2}$

- $X_{\frac{p}{2}}$ = $A_1,B_1,A_2,B_2,C_1,C_2,B_1,A_1,A_2,C_1,C_2$
- λ = 11
- Γ = $\{A_1,A_2,B_1,B_2,C_1,C_2\}$
- η = 6
- m_1 = 4
- algoritm de înlocuire - LRU
  - LRU modificat

./.

Succesiunea referirilor		1	2	3	4	5	6	7	8	9	10	11
Pagina p	Sirul de referiri	A	B	A	B	C	C	B	A	A	C	C
	Defecte de pagină	*	*			*			*		*	
	Conţinutul lui M_1 1	A	B	A	B	C	C	B	A	A	C	C
	2		A	B	A	B	B	C	B	B	A	A
Algoritm LRU	Sirul de referiri	A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1	C_2
	Defecte de pagină	*	*	*	*	*	*	*	*	*	*	*
	Conţinutul lui M_1 1	A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1	C_2
	2		A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1
	3			A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2
	4				A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1
Pagina $\frac{p}{2}$	Sirul de referiri	A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1	C_2
	Defecte de pagină	*	*	*	*	*	*		*	*	*	*
	Conţinutul lui M_1 1	A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1	C_2
Algoritm LRU modificat	2		A_1	A_1	B_1	B_2	C_1	B_2	B_1	A_1	A_2	C_1
	3			B_1	A_2	B_1	B_2	C_2	B_2	B_1	A_1	A_2
	4				A_1	A_2	B_1	C_1	C_2	B_2	B_1	A_1

Rezultate:

- algoritm LRU - $e_p = 5$
  - $e_{\frac{p}{2}} = 11$
  - $\varsigma = \frac{11}{5} = 2,2$
- algoritm LRU modificat: - $e_p = 5$
  - $e_{\frac{p}{2}} = 10$
  - $\varsigma = \frac{10}{5} = 2$

Fig.36. Exemplu de utilizare a algoritmului LRU modificat.

Din fig.36 rezultă că prin aplicarea algoritmului LRU modificat, raportul ς a fost limitat la valoarea 2, deşi el atingea valoarea 2,2 în cazul utilizării unui algoritm de înlocuire LRU normal. Din figură rezultă de asemenea, modul cum procedura descrisă în secţiunea 2 precedentă afectează algoritmul de înlocuire. Astfel, se pot observa diferenţe

./.

între conţinuturile lui M_1 în cazul aplicării algoritmului LRU şi în cazul aplicării algoritmului LRU modificat.

Pentru scopuri de referire, în fig.36 fiecare stare a conţinutului M_1 este identificată cu numere de la 1 la 11.

In implementarea descrisă a algoritmului LRU modificat ori de cîte ori ambele pagini ce alcătuiesc o grupă se găsesc în M_1 ele sînt întotdeauna învecinate în ordonarea realizată în M_1.

Astfel, la pasul 3 al şirului de referiri de pagină se poate constata prima deosebire între modul de aplicare al algoritmului LRU şi al celui LRU modificat. Pagina A_2 este referită în acest pas şi este adusă în M_1. In cazul utilizării algoritmului LRU, A_2 este plasată în fruntea ordonării paginilor din M_1, ordonare care devine A_2, B_1, A_1. Pe de altă parte, în cazul utilizării algoritmului LRU modificat, la pasul 3 este observat că pagina A_1 a fost deja adusă în M_1. Astfel, în momentul în care pagina A_2 este plasată în fruntea ordonării paginilor din M_1, A_1 este cuplat cu A_2, rezultînd ordonarea A_2, A_1, B_1.

La pasul 7 al şirului de referiri de pagină se constată un alt aspect de interes al modelului cum acţionează algoritmul LRU modificat. La pasul precedent ordonarea paginilor în M_1 a fost C_2 C_1 B_2 B_1.

Cînd se face referirea la B_1, în pasul 7, nu există necesitatea de a iniţia o aducere de pagină, deoarece B_1 este deja în M_1. Ordonarea paginilor din M_1 devine: B_1 B_2 C_2 C_1, căci algoritmul LRU impune ca referirea cea mai recentă B_1 să fie plasată în fruntea ordonării. Insă modul cum a fost modificat acest algoritm a determinat ca pagina B_2 să fie de asemenea deplasată spre fruntea ordonării, pentru a continua să fie învecinată, în cadrul ordonării din M_1, cu B_1.

Există diferite moduri în care poate fi realizat deziderатul ca decizia de înlocuire a paginilor să fie luată pe baza considerării ansamblului paginilor unei grupe. In algoritmul LRU modificat descris mai sus, ori de cîte ori toate paginile ce formează o grupă se găsesc în M_1, ele sînt rearanjate, în ordonarea paginilor din M_1, astfel ca să fie învecinate.

Condiţia ca o pagină a unei grupe să nu poată fi înlocuit din M_1, dacă pagina martor corespunzătoare nu a fost şi ea înlo-

./.

cuită din M_1, poate fi realizată în diferite moduri.Astfel, şirul de înlocuire LRU poate fi lăsat în ordonarea lui normală, conformă aplicării unui algoritm LRU obişnuit, însă la apariţia necesităţii de a înlocui o pagină din M_1 trebuie construit un algoritm care să cerceteze şirul LRU şi să selecteze pagina cerută pentru înlocuire, pe baza considerării ansamblului paginilor din grupe.

3.2. Algoritm FIFO modificat.

In fig. 37 este cercetat un şir de referiri de pagină, în cazul utilizării unor pagini de dimensiunea de p şi respectiv $\frac{p}{2}$ octeţi şi folosirea ca algoritm de înlocuire atît a FIFO, cît şi FIFO modificat.

Parametrii sistemului, în cazul utilizării paginii p:

- X_p = A,B,A,B,C,C,B,A,A,C,C
- λ = 11
- γ = {A,B,C}
- η = 3
- m_1 = 2
- algoritm de înlocuire FIFO

Parametrii sistemului, în cazul utilizării paginii $\frac{p}{2}$

- $X_{\frac{p}{2}}$ = $A_1,B_1,A_2,B_2,C_1,C_2,B_1,A_1,A_2,C_1,C_2$
- λ = 11
- γ = $\{A_1,A_2,B_1,B_2,C_1,C_2\}$
- η = 6
- m_1 = 4
- algoritm de înlocuire - FIFO
  - FIFO modificat

./.

Succesiunea referirilor			1	2	3	4	5	6	7	8	9	10	11
Pagina p	Sirul de referiri		A	B	A	B	C	C	B	A	A	C	C
	Defecte de pagină		*	*			*			*			
	Conținutul lui M_1	1	A	B	B	B	C	C	C	A	A	A	A
		2		A	A	A	B	B	B	C	C	C	C
Algoritm FIFO	Sirul de referiri		A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1	C_2
	Defecte de pagină		*	*	*	*	*	*	*	*	*	*	*
	Conținutul lui M_1	1	A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1	C_2
		2		A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1
		3			A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2
		4				A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1
Pagina $\frac{p}{2}$ Algoritm FIFO modificat	Sirul de referiri		A_1	B_1	A_2	B_2	C_1	C_2	B_1	A_1	A_2	C_1	C_2
	Defecte de pagină		*	*	*	*	*	*		*	*		
	Conținutul lui M_1	1	A_1	B_1	A_2	B_2	C_1	C_2	C_2	A_1	A_2	A_2	A_2
		2		A_1	A_1	B_1	B_2	C_1	C_1	C_2	A_1	A_1	A_1
		3			B_1	A_2	B_1	B_2	B_2	C_1	C_2	C_2	C_2
		4				A_1	A_2	B_1	B_1	B_2	C_1	C_1	C_1

Rezultate:

. algoritm FIFO - $e_p = 4$

- $e_{\frac{p}{2}} = 11$

- $\varrho = \frac{11}{4} = 2,75$

. algoritm FIFO modificat

- $e_p = 4$

- $e_{\frac{p}{2}} = 8$

- $\varrho = \frac{8}{4} = 2$

Fig.37. Exemplu de utilizare a algoritmului FIFO modificat.

Din fig.37 se observă că prin utilizarea algoritmului FIFO modificat, raportul ϱ a fost limitat la valoarea 2, deși el atingea valoarea 2,75 în cazul utilizării unui algoritm de înlocuire FIFO normal.

Exemplul din fig.37 ilustrează complet toate aspectele importante implicate de utilizarea algoritmului FIFO modificat asupra ordonării paginilor. In particular, presupunem de exemplu că pagina A_1 este referită în cadrul unui şir de referiri de pagină şi, întrucît nu se găseşte în M_1, trebuie adusă. Ordinea paginilor conţinute în M_1 se modifică după cum urmează:

1. Dacă pagina A_2 nu este conţinută în M_1, atunci pagina A_1 este plasată în fruntea ordonării FIFO.

2. Dacă pagina A_2 este conţinută în M_1, atunci pagina A_1 este plasată imediat înaintea paginii A_2 în ordonarea FIFO. Ordonarea relativă a paginii A_2 rămîne nemodificată.

Motivul ce stă la baza părţii a doua a acestei proceduri decurge din regula de ordonare FIFO normală, care plasează o pagină A în fruntea ordonării numai dacă ea nu era deja conţinută în M_1. Dacă ea se afla deja în M_1, ea rămîne la poziţia ei anterioară de ordonare. Pe de altă parte în cadrul unui algoritm modificat pe baza procedurii descrise în secţiunea 2 decizia de înlocuire a paginilor se ia pe baza considerării ambelor pagini (A_1, A_2) ale unei grupe.

4. Eficienţa algoritmilor modificaţi.

Procedura propusă de modificare a algoritmilor de înlocuire uzuali are în mod evident o influenţă asupra eficienţei generale a algoritmului de bază de înlocuire ce este utilizat.

Putem considera că comportarea de acces a unui program în spaţiul adreselor, în timpul unei activităţi de scurtă durată, poate fi caracterizată prin trei regiuni, delimitate de valoarea raportului ϱ al numerelor defectelor de pagină obţinute în cadrul utilizării dimensiunilor de pagină de p şi $\frac{p}{2}$ şi în presupunerea că nu este utilizată procedura propusă de modificare a algoritmului de înlocuire.

1. referire rară - ϱ mic ($\varrho < 1$)
2. referire moderată- ϱ moderat ($1 < \varrho < 2$)
3. referire densă - ϱ mare ($\varrho > 2$)

In regiunea de referire rară, este puţin probabil ca ambele jumătăţi A_1 şi A_2 ale unei pagini martor A să se găsească în M_1 simultan: ca atare procedura propusă de modi-

./.

ficare a algoritmului de înlocuire va avea un efect minim asupra performanţei.

In regiunea de referire moderată, algoritmul de înlocuire modificat acţionează aproape la fel de eficient ca şi algoritmul de înlocuire nemodificat. Astfel, pe baza considerării fig.38, unde este analizat modul de comportare al algoritmului de înlocuire LRU şi al celui LRU modificat asupra unui şir de referiri de pagină, se poate constata că utilizarea algoritmului de înlocuire LRU modificat nu numai că menţine raportul ϱ al numerelor defectelor de pagină mic, dar, în cazul considerat, conduce chiar la micşorarea lui.

Parametrii sistemului, în cazul utilizării dimensiunii p de pagină:

- X_p = A,B,A,C,B,C,A,B,C,B,C
- λ = 11
- γ = {A,B,C}
- η = 3
- m_1 = 2
- algoritm de înlocuire LRU

Parametrii sistemului, în cazul utilizării dimensiunii $\frac{p}{2}$ de pagină:

- $X_{\frac{p}{2}}$ = $A_1,B_1,A_2,C_1,B_2,C_2,A_2,B_1,C_1,B_2,C_2$
- λ = 11
- γ = $\{A_1,A_2,B_1,B_2,C_1,C_2\}$
- η = 6
- m_1 = 4
- algoritm de înlocuire - LRU
  - LRU modificat

./.

Succesiunea referirilor			1	2	3	4	5	6	7	8	9	10	11
Pagina p	Şirul de referiri		A	B	A	C	D	D	A	D	C	B	C
	Defecte de pagină		*	*		*	*		*	*	*		
	Conţinutul lui M_1	1	A	B	A	C	D	D	A	D	C	B	C
		2		A	B	A	C	D	C	A	D	C	B
Algoritm LRU	Şirul de referiri		A_1	B_1	A_2	C_1	D_2	D_1	A_2	B_1	C_1	B_2	C_2
	Defecte de pagină		*	*	*	*	*		*	*	*	*	*
	Conţinutul lui M_1	1	A_1	B_1	A_2	C_1	D_2	D_1	A_2	B_1	C_1	B_2	C_2
		2		A_1	B_1	A_2	C_1	D_2	C_2	A_2	B_1	C_1	B_2
		3			A_1	B_1	A_2	C_1	D_2	C_2	A_2	B_1	C_1
		4				A_1	B_1	A_2	C_1	D_2	C_2	A_2	B_1
Pagina $\frac{p}{2}$	Şirul de referiri		A_1	B_1	A_2	C_1	D_2	D_1	A_2	B_1	C_1	B_2	C_2
	Defecte de pagină		*	*	*	*	*		*	*	*		
	Conţinutul lui M_1	1	A_1	B_1	A_2	C_1	D_2	D_1	A_2	B_1	C_1	B_2	C_2
		2		A_1	A_1	A_2	B_1	C_1	C_2	B_2	C_2	B_1	C_1
Algoritm LRU modificat		3			B_1	A_1	C_1	B_2	C_1	A_2	B_1	C_1	B_2
		4				B_1	A_2	B_1	B_2	C_2	B_2	C_2	B_1

Rezultate:

- algoritm LRU
  - $e_p = 7$
  - $e_{\frac{p}{2}} = 10$
  - $\varrho = \frac{10}{7} = 1,43$

- algoritm LRU modificat:
  - $e_p = 7$
  - $e_{\frac{p}{2}} = 9$
  - $\varrho = \frac{9}{7} = 1,29$

Fig.38. Utilizarea algoritmului LRU modificat în regiunea cu ϱ moderat.

./.

In regiunea de referire densă, am văzut deja în secţiunea 3 a acestui capitol că utilizarea procedurii propuse de modificare a algoritmilor de înlocuire previne valori extreme ale lui ς .

In concluzie, procedura propusă de modificare a algoritmilor de înlocuire fixează o limită superioară pentru raportul ς al numerelor defectelor de pagină pentru regiunea cu ς mari, fără a micşora valoarea proporţiei succeselor în regiunile cu ς originali mici.

Astfel, prin utilizarea acestei proceduri se creează posibilitatea ca să se obţină îmbunătăţiri suplimentare în gestiunea unui sistem ierarhic de memorii prin utilizarea unor dimensiuni reduse de pagină, fără riscul unui consum mare de timp neproductiv datorat activităţii exagerate de paginare, procedura propusă evitînd apariţia anomaliei dimensiunii de pagină.

5. Perspective.

Unele probleme tratate în lucrare deschid noi direcţii de cercetare. Cea mai importantă ni se pare studiul şi elaborarea unor noi algoritmi de înlocuire, spaţiali, care să ia în consideraţie conceptul de localizare spaţială. Este posibil să existe şi alte soluţii în această direcţie, pe lîngă procedura de modificare a algoritmilor uzuali, propusă în lucrare.

In legătură cu algoritmii modificaţi, descrişi în secţiunea 3, mai rămîn probleme nerezolvate. De exemplu, care este relaţia algoritmilor modificaţi descrişi cu clasa algoritmilor de stivă, studiaţi de Mattson ş.a. (121) (vezi cap.2, secţiunea 4.1.2.) în particular, în ce condiţii, dacă există, un algoritm modificat este un algoritm de stivă. Asemănător, care este relaţia algoritmilor modificaţi cu algoritmul de înlocuire teoretic optim, numit OPT (121) sau MIN (18,20).

Din punct de vedere practic, rămîne de arătat cît de eficientă se poate dovedi implementarea unui algoritm de înlocuire modificat conform procedurii propuse sau a unui alt algoritm de înlocuire spaţial. In acest scop sînt necesare măsurători ale performanţei unui sistem ierarhic de memorii, utilizînd un astfel de algoritm de înlocuire sau cel puţin este necesară o analiză de simulare mai extinsă.

CAP.7. CONCLUZII

In lucrarea de faţă s-a realizat cercetarea unei probleme neclarificate pînă în prezent apărute în gestiunea automată a ierarhiilor de memorii şi anume situaţia de apariţie a anomaliei dimensiunii de pagină. Astfel, s-a arătat că, deşi prin utilizarea unor dimensiuni mici de pagină în cadrul unei ierarhii de memorii se aduc îmbunătăţiri în utilizarea sistemului de memorii, apariţia anomaliei dimensiunii de pagină în cazul utilizării unor dimensiuni reduse de pagină conduce la scăderi în utilizarea sistemului de memorii, cauzate de activitatea exagerată de paginare ce are loc în cadrul acestei anomalii.

Cercetarea acestei probleme, realizată cu ajutorul unui model original al comportării de acces a unui program operînd într-un mediu cu memorie virtuală, stabileşte cauzele acestei anomalii şi, pe accastă bază, propune o procedură de modificare a algoritmilor de înlocuire a paginilor utilizaţi în cadrul gestiunii ierarhiei de memorii, procedură ce evită apariţia anomaliei dimensiunii de pagină.

Procedura propusă de modificare a algoritmilor de înlocuire are un interes practic imediat. Utilizarea ei creează posibilitatea obţinerii unor îmbunătăţiri suplimentare în gestiunea unui sistem ierarhic de memorii prin utilizarea unor dimensiuni reduse de pagină, fără riscul unui consum mare de timp neproductiv datorat activităţii exagerate de paginare, procedura propusă evitînd apariţia anomaliei dimensiunii de pagină.

Contribuţiile originale aduse în lucrare de autor, în ordinea din teză, sînt:

1. Prezentarea unitară a problemelor actuale majore ale studiului legat de sistemele ierarhice de memorii (cap.2.
2. Analiza factorilor implicaţi în alegerea dimensiunii de pagină utilizate în cadrul unei ierarhii de memorii, prezentarea situaţiei de anomalie a dimensiunii de pagină şi stabilirea necesităţii unor studii dedicate acestei probleme (cap.3.);
3. Analiza critică a modelelor analitice descrise în literatură ale comportării de acces a unui program operînd

./.

într-un mediu cu memorie virtuală şi precizarea inaplicabilităţii lor în cercetarea problemei anomaliei dimensiunii de pagină (cap.4.);

4. Realizarea unui instrument care să poată fi utilizat în cercetarea anomaliei dimensiunii de pagină, respectiv elaborarea unui model analitic al comportării de acces a unui program. Modelul, ce ţine seama de variaţia dimensiunii de pagină utilizate, stabileşte o expresie analitică pentru funcţia DPA ce determină valoarea medie a numărului de defecte de pagină, ce apar în decursul execuţiei unui program într-un mediu cu memorie virtuală (cap.4.);
5. Studiul proprietăţilor funcţiei DPA definite prin modelul elaborat al comportării de acces a unui program (cap.4.);
6. Regăsirea pe cale analitică, cu ajutorul modelului elaborat al comportării de acces a unui program operînd într-un mediu cu memorie virtuală, a unor rezultate numerice experimentale recunoscute ca semnificative (cap.4.);
7. Studiul analitic, cu ajutorul modelului elaborat al comportării de acces a programului, a situaţiilor de apariţie a anomaliei dimensiunii de pagină (cap.5.);
8. Studiul comportării localizate, temporale şi spaţiale, a unui program şi a modului în care algoritmii de înlocuire uzuali iau în consideraţie această comportare localizată (cap.5.);
9. Introducerea conceptului de comportare localizată spaţio-temporală şi interpretarea cu caracter general, pe baza acestui concept, a apariţiei anomaliei dimensiunii de pagină (cap.5.);
10. Identificarea mecanismului de apariţie a anomaliei dimensiunii de pagină pe baza cercetării şirului de referiri de pagină a programului (cap.6.);
11. Elaborarea unei proceduri de prevenire a apariţiei anomaliei dimensiunii de pagină şi descrierea modului ei de aplicare (cap.6.);

./.

12. Modificarea unor algoritmi uzuali de înlocuire pe baza procedurii propuse de prevenire a apariţiei anomaliei dimensiunii de pagină şi precizarea modului lor de operare (cap.6.);
13. Studiul eficienţei algoritmilor modificaţi prin procedura propusă de prevenire a apariţiei anomaliei dimensiunii de pagină (cap.6.);
14. Precizarea unor perspective de interes în domeniu prin indicarea unor noi direcţii posibile de cercetare, ca şi a unor probleme rămase nerezolvate (cap.6.).

B I B L I O G R A F I E

ABREVIERI

CACM - Communications of the ACM
JACM - Journal of the ACM
SJCC - Spring Joint Computer Conference
FJCC - Fall Joint Computer Conference
NCC - National Computer Conference
IEEE-TC - IEEE Transactions on Computers
IEEE-TEC- IEEE Transactions on Electronic Computers

xxx - Programul Partidului Comunist Român de făurire a societăţii socialiste multilateral dezvoltate şi înaintare a României spre comunism. Congresul XI al PCR, nov.1974.

xxx - Orientările generale cu privire la întocmirea planului cincinal de dezvoltare economico-socială a României în perioada 1981-1985. Conferinţa Naţională a PCR din 7-9 dec.1977

1. Aho,A.V.,P.J.Denning şi J.Ullman - Principles of Optimal Page Replacement, JACM 18, 1(January 1971), 80-93.
2. Amdahl, G.M. şi L.D.Amdahl - Fourth-Generation Hardware, Datamation (January 1967).
3. Amelio, G.F. - Charge Coupled Devices for Memory Applications, Proceedings of the 1975 N.C.C., 512-522.
4. Anacker,W. - Superconducting Memories Employing Josephson Devices Proceedings of the 1975 NCC, 529-534.
5. Anacker,W. şi C.P.Wong - Performance Evaluation of Computer Systems with Memory Hierarchies, IEEE-TEC EC-16, 6 (December 1967), 765-773.
6. Arden,B.W.,B.A.Geller,T.C.O'Brien şi F.H.Westervelt - Programming and Addressing Structure in a Time-Sharing Environment, JACM,13, (1966), 1-16.
7. Arora,S.R. şi A.Gallo - Optimal Sizing, Loading and Re-Loading in a Multilevel Memory Hierarchy System, SJCC, 38(1971), 337-344.

./.

8. Auerbach Computer Technology Reports Series, Auerbach Publishers Inc.
Segments A General Purpose Computers
B Standard Peripherals
J System Software

9. Auerbach Reporter,Fine-Tuning Virtual Storage for Efficiency, Auerbach Publishers Inc.,(Dec.1974).

10. Austin,B.J. - A Dinamic Disc Allocation Algorithm Designed to Reduce Fragmentation During File Re-Loading, The Computer Journal, 14,4 (1971) 378-381.

11. Bard,Y. - Experimental Evaluation of System Performance, IBM Syst.J., 12 (1973) 302-318.

12. Bard,Y. - Characterization of Program Paging in a Time Sharing Environment, IBM J.Res.Develop 17, 5 (sept.1973), 387-393.

13. Bard,Y. - Application of the Page Survival Index (PSI) to Virtual Memory System Performance, IBM J.Res.Develop., 19,3 (May 1975), 212-220.

14. Batson,A.,Shy-Ming In şi D.C.Wood - Measurements of Segment Size, CACM 13, 3 (March 1970), 155-159.

15. Baylis,M.H.J.,D.G.Fletcher şi D.J.Howarth - Paging Studies Made on the ICT Atlas Computer, Proceedings IFIP Congress 1968,2,835-836.

16. Belady,L.A.,R.A.Nelson şi G.S.Shedler - An Anomaly in Space-Time Characteristics of Certain Programs Running in a Paging Machine, CACM 12, 6(June 1969), 349-353.

17. Belady,L.A. şi C.J. Kuehner - Dinamic Space Sharing in Computer Systems,CACM 12, 5(May 1969), 282-288.

18. Belady,L.A. - A Study of Replacement Algorithms for a Virtual Storage Computer, IBM Syst.J., 5, 2(1966), 78-101.

19. Belady, L.A. - Biased Replacement Algorithms for Multiprogramming, IBM Thomas Watson Research Center, Research Note NC 697 (March 1967).

20. Belady, L.A. şi F.P. Palermo - On-Line Measurement of Paging Behavior by the Multivalued MIN Algorithm, IBM J.Res.Develop., 18, 1(Jan.1974), 2-19.

21. Bell,G.C. şi D.Casasent - Implementation of a Buffer Memory in Minicomputers, Computer Design (nov.1971), 83-89.

22. Bell,G.C. şi A. Newell - A Panel Session-Computer Structure - Past, Present and Future, FJCC, 1971.

23. Bensonssan,A., C.T.Clingen şi R.C.Daley - The Multics Virtual Memory, Proceedings of the ACM Second Symposium on Operating System Principles, Princeton University,(Oct.20-22,1969), 30-42.

./.

24. Bobeck,A.H., P.I.Boneyard,J.E.Gensic - Magnetic Bubbles - An Emerging New Memory Technology, Proceedings of the IEEE, Aug.1975, 1176-1195.

25. Bogdan R.C., C. Bilciu, A.Davidoviciu şi D.Schiopulescu - Memoriile Calculatoarelor Electronice Funcţionare şi Utilizare, Editura Tehnică, Bucureşti, 1975.

26. Brawn,B.S. şi F.G. Gustavson - Program Behavior in a Paging Environment, Afips Conference Proceedings, FJCC 33 (1968) 1019-1032.

27. Brewer,J.E. şi D.R. Hadden - Block-Oriented Random Access LNOS Memory, Proceedings of the 1974 NCC, Afips, Vol.43, 837-840.

28. Bryant,P. - Predicting Working Set Sizes, IBM J.Res.Develop. 19, 3 (May 1975), 221-229.

29. Buzen, J. - Optimizing the Degree of Multiprogramming in Demand Paging Systems, Proceedings of the 1971 International Computer Society Conference.

30. Carnes, J.E., W.F.Kosonocky, J.K.Chambers şi D.J. Sauer - Charge-Coupled Devices for Computer Memories, Proceedings of the 1974 NCC, 827-835.

31. Chamberlain,D.D., S.H.Fuller şi L.Y.Liu - An Analysis of Page Allocation Strategies for Multiprogramming Systems with Virtual Memory, IBM J.Res.Develop, 17, 5 (sept.1973), 404-412.

32. Chandy,K.M. - The Analysis and Solution of General Queueing Networks, Proceedings of the Sixth Annual Princeton Conference on Information Sciences and Systems, Princeton University, Princeton, NJ, March 1972.

33. Chandy, K.M., T.W.Keller şi J.C. Browne - Design Automation and Queueing Networks: An Interactive System for the Evaluation of Computer Queueing Models, Ninth Annual Design Automation Workshop, Dallas, Texas, June 1972.

34. Chang Hsu - Capabilities of the Bubble Technology, Proceedings of the 1974 NCC, Afips, Vol.43, 847-855.

35. Chow, C.K. - On Optimization of Storage Hierarchies - IBM J.Res. Develop. 18, 3 (May 1974) 194-203.

36. Chu, W.W. şi H. Opderbeck - The Page Fault Frequency Replacement Algorithm, Proceedings 1972 FJCC 41, 597-609.

37. Chu,W.W., N.Oliver şi H. Opderbeck - Measurement Data on the Working Set Replacement Algorithm and their Applications - Proceedings 22-ND International Symposium on Computer Communications, Networks and Teletraffic, Polytechnic Institute of Brooklyn Publications, April 1972.

38. Coffman,E.G. - Analysis of a Drum Input/Output Queue Under Scheduled Operation in a Paged Computer System, JACM, 16 (1969), 73.
xxx ERRATA, JACM 16 (1969), 646.

./.

39. Coffman,E.G. şi T.A.Ryan - A Study of Storage Partitioning Using a Mathematical Model of Locality CACM, 15, 3, 1972.

40. Coffman,E.G. şi L.C.Varian - Further Experimental Data on the Behavior of Programs in a Paging Environment CACM, 11, 5 (July 1968), 471-474.

41. Coffman,E.G. şi R.C. Wood - Interarrival Statistics for Time-Sharing Systems, CACM 9, 7 (July 1966), 500-503.

42. Cojanu M, C.Ionescu - Evoluţie şi Perspective în Domeniul Memoriilor pentru Sistemele de Calcul, ALC, Vol.23, 285-298, Editura Tehnică, Bucureşti, 1976.

43. Considine,J.P. şi A.H.Weis - Establishment and Maintenance of a Storage Hierarchy for an On-Line Data Base under TSS/360, FJCC 35 (1969), 433-440.

44. Conti,C.J., D.H.Gibson şi S.H.Pitkowsky - Structural Aspects of the System/360 Model 85: I.General Organization, IBM Sys J, 7, 1 (1968), 2-14.

45. Conti,C.J. - Concepts for Buffer Storage, IEEE Computer Group News (March 1969), 6-13.

46. Cook, R.W. şi M.J. Flynn - System Design of a Dynamic Microprocessor, IEEE-TC, C-19, 3 (March 1970), 213-222.

47. Corbato, F.J. şi V.A. Vyssotsky - Introduction and Overview of the Multics System, Afips, FJCC, Afips Press, Spartan, Washington D.C.,1965, 185-196.

48. Daley,R.C. şi J.B. Dennis - Virtual Memory, Processes and Sharing in Multics, CACM 11, 5, (May 1968), 306-312.

49. Datapro 70 - The EDP Buyers Bible, Datapro Research Corp., MCGRAW - HILL CO.
Sections: 70 C Computers
70 D Peripherals
70 E Software

50. Dell,H.R. - Design of a High-Density Optical Mass Memory System, Computer Design, August 1971, 49-53.

51. De Meis,W.M. şi N. Weizer - Measurement and Analysis of a Demand Pagind Time-Sharing System, Proceedings 24-TH National Conference of ACM, ACM Publications, Thompson, Washington,D.C., 1969, 201-216.

52. Denning,P.J. - The Working Set Model for Program Behavior, CACM 11, 5 (May 1968), 323-333.

53. Denning,P.J. - Virtual Memory, Computing Surveys, 2, 3 (sept.1970), 153-183.

54. Denning,P.J. - Third Generation Computer Systems, Computing Surveys, 3, 4 (Dec.1971), 175-216.

55. Denning,P.J. - Thrashing - Its Causes and Prevention, FJCC 33 (1968), 915-922.

./.

56. Denning,P.J. şi S.C.Schwartz - Properties of the Working Set Model, CACM 15 (1972), 191.

57. Denning,P.J. - Memory Allocation in Multiprogrammed Computers, MIT Project MAC, Computation Structures Group, Memo No 24 (March 1966).

58. Denning, P.J. - Effects of Scheduling on File Memory Operations, Afips Conf.Proc.30, 1967, SJCC, 9-21.

59. Denning, P.J. şi J.R. Spirn - Dynamic Storage Partitioning, 4th ACM Operating System Symposium, IBM Thomas J. Watson Research Center, Yorktown Heights, New-York, Oct.1973.

60. Denning,P.J. - On Modelling Program Behavior,Proc. SJCC, Afips, Spartan Books, New-York, 1972.

61. Dennis, J.B. - Program Structure in a Multiacces Computer, MIT Project MAC, Report MAC-TR-11.

62. Dennis,J.B. - Segmentation and the Design of Multiprogrammed Computer System, JACM, 12, 4 (Oct.1964), 589-602.

63. Dodescu G., D.Ionescu - Memoria Virtuală a Sistemelor de Calcul, ALC, Vol.23, 263-284, Editura Tehnică, Bucureşti, 1976.

64. Donovan,J.J. - Systems Programming, MCGRAW-HILL, New York, 1972.

65. Farr,W.W. şi W.E.Peisel - An Optimum Disc Organization for a Virtual Memory System, Computer Design (June 1971),49-54.

66. Femling,D. - Rubber-Band Memory, Electronic Design 13 (June 1971), 64-68.

67. Ferrari,D. - Improving Locality by Critical Working Sets, CACM 17 (1974), 614.

68. Ferrari,D. - Improving Program Locality by Strategy-Oriented Restructuring, Information Processing 74, North-Holland Publishing Co, Amsterdam, 1974, 266-270.

69. Ferrari,D. - Tailoring Programs to Models of Program Behavior, IBM J. Res.Develop. 19, 3 (May 1975), 244-251.

70. Fetch,G.C. - Memory Organization and Hierarchies of Storage Workshop, Computer Group News (Jan.1969), 24-25.

71. Fikes,R.E., H.C.Lauer şi A.L.Varcha-Jr. - Steps towards a General Purpose Time-Sharing System Using Large Capacity Core Storage and TSS/360, Proceedings of the 23-th ACM National Conference (1968), 7-15.

72. Fine,G.H., C.W.Jackson şi P.V.Mcisaac - Dynamic Program Behavior under Paging, Proceedings 21-th National Conference of ACM, ACM Publications, Thompson, Washington,D.C., (1966), 223-228.

73. Fotheringham,J. - Dynamic Storage Allocation in the Atlas Computer Including an Automatic Use of a Backing Store, CACM-4, 10 (Oct. 1961), 435-436.

74. Freiberger,W.F., U.Grenander şi P.D. Sampson - Patterns in Program References, IBM J.Res.Develop., 19, 3 (May 1975), 230-243.

./.

75. Freibergs,I.F. - The Dynamic Behavior of Programs, Afips, FJCC, Afips Press, Thompson, Washington, DC, 1968, 1163-1168.

76. Gaver,D.P., P.A.W.Lewis şi G.S.Shedler - Analysis of Exception Data in a Staging Hierarchy, IBM J. Res. Develop., 18, 5 (sept.1974), 423-435.

77. Gecsei,J. - Determining Hit Ratios for Multilevel Hierarchies, IBM J.Res.Develop., 18, 4 (July 1974), 316-327.

78. Gentile,R.B. şi R.W.Grove - Mass Storage Utility: Considerations for Shared Storage Applications, IEEE Transactions on Magnetics MAG-7, 4 (Dec.1971) 848-852.

79. Gentile,R.B. şi J.R. Lucas Jr. - The Tablon Mass Storage Network, SJCC 38 (1971), 345-356.

80. Gerz,J.L. - Hierarchical Associative Memories for Parallel Computation, MIT Project MAC, Report MAC-TR-69, MIT, Cambridge, Mass.(June 1970).

81. Ghanem,M.Z. - Dynamic Partitioning of the Main Memory Using the Working Set Concept, IBM J.Res.Develop., 19, 5 (sept.1975), 445-450.

82. Ghanem,M.Z. - Study of Memory Partitioning for Multiprogramming Systems with Virtual Memory, IBM J.Res. Develop. 19, 5 (sept.1975), 451-457.

83. Gibson,C.T. - Time-Sharing in the IBM-System/360 Model 67, Proc. Afips, 1966 SJCC,vol.28, Spartan Books, New-York, 61-78.

84. Gibson,D.H. - Considerations in a Block-Oriented System Design, SJCC, 1967.

85. Gillis,A.C., G.E.Hoffman şi R.H. Nelson - Holographic Memories - Fantasy or Reality, Proceedings of the 1975 NCC, 535-539.

86. Goldberg, R.P. - Virtual Machine Systems, MIT Lincoln Laboratory, Technical Memorandum No 282-0036 (Aug.1969).

87. Guertin,R.L. - Programming in a Paging Environment, Datamation 18, 2 (Feb.1972), 48-55.

88. Hatfield, D.J. - Experiments on Page Size,Program Access Pattern and Virtual Memory Performance,IBM J. Res. Develop., 16, 1 (Jan.1972), 58-66.

89. Hatfield,D.J. şi J.Gerald - Program Restructuring for Virtual Memory, IBM, Syst.J., 10, 3 (1971), 168-192.

90. Hoagland,A.S. - Mass Storage - Past, Present and Future, Computers, 6, 8 (sept.1973), 29-33.

91. Hoagland,A.S. - Magnetic Recording Storage, IEEE Trans on Computers, vol.C-25, No 12, (Dec.1976), 1283-1288.

./.

92. Hodges,D.A. - Alternative Component Technologies for Advanced Memory Systems, Computers 6, 8 (sept.1973), 35-37.

93. Hodges, D.A. - A Review and Projection of Semiconductor Component for Digital Storage, Proceedings of the IEEE, Aug. 1975, 1136-1147.

94. Hodges, D.A. - Trends in Computer Hardware Technology, Computer Design, vol.15, No.2, (Feb.1976), 77-85.

95. Hughes,W.C., C.Q. Lemmond, H.G. Parks, G.J.Ellis, G.E.Possin şi R.H.Wilson - Beamos - A New Electronic Digital Memory, Proceedings of the 1975 NCC, 541-549.

96. IBM - A Guide to the IBM System/370 Model 158,GC 20-1754-2 File No.S/370-01.

97. IBM - A Guide to the System/370 Model 165,Form MR GC 20 - 1730 (June 1970), 19-25.

98. IBM - System/370 - System Summary,GA 22-7001-4 File No S/370-0

99. IBM - Virtual Machine Facility/370 Introduction,Form No GC 20-1800.,IBM Data Processing Division, White Plains, N.Y.,1972.

100. ICL - Documentaţie tehnică privind seria de calculatoare ICL-2

101. Irani,K.B. şi V.L.Wallace - On Network Linguistics and the Conversational Design of Queueing Networks,JACM, Oct.1971.

102. Joseph,M. - An Analysis of Paging and Program Behavior - The Computer Journal, 13, 1 (Feb.1970), 48-54.

103. Kaneko,T. - Optimal Task Switching Policy for Multilevel Storage System, IBM J.Res.Develop. 18, 4 (July 1974), 310-315.

104. Katzan, H.Jr. - Storage Hierarchy Systems, SJCC, Afips Press, 38 (1971), 325-336.

105. Kilburn,T.,D.B.G.Edwards, M.J.Lanigan şi F.H. Summer - One-Level Storage Systems, IEEE-TEC, EC-11, 2 (April 1962), 223-235.

106. Kotak,A. - Lecture Notes CS 252, Digital Equipment Co., Marlborough, Mass., 1976.

107. Laliotis, T.A. - Main Memory Technology, Computers, 6, 8 (sept. 1973), 21-27.

108. Lauer,H.C. - Bulk Core in a 360/67 Time-Sharing System, Computer Design 7, 4 (April 1968), 94-101.

109. Lee,P.F. - Study of a Look-Aside Memory, IEEE-TEC, C-18, 11 (Nov.1969).

110. Lew, A. - On Optimal Pagination of Programs, University of Hawaii Information Sciences Report, Honolulu, Hawaii, (May 1970).

111. Lewis, P.A.W. şi G.S.Shedler - Empirically Derived Micromodels for Sequences of Page Exceptions, IBM J.Res.Develop., 17, 2 (March 1973), 86-100.

112. Lewis,P.A.W.şi G.S.Shedler - A Cyclic Queue Model of System Overhead in Multiprogrammed Computer System, JACM 18, 2, (April 1971), 199-220.

./.

113. Lewis,P.A.W. şi P.C.Yue - Statistical Analysis of Program Reference Patterns in a Paging Environment,Proceedings of the 1971 International Computer Society Conference (sept.1971) 133-134.

114. Lin,Y.S. şi R.L. Mattson - Cost-Performance Evaluation of Memory Hierarchy, IEEE Trans.Magnetics, Mag.-8 (1972), 390.

115. Liptay,J.S. - Structural Aspects of the System/360 Model 85. II The Cache, IBM Syst.J.7, 1 (1968), 15-21.

116. Lorin,H. - Parallelism in Hardware and Software, Prentice Hall, Inc., Englewood Cliffs, New-Jersey, 1972.

117. Mackenzie,F.B. - Automated Secondary Storage Management, Datamation 11, 11 (1965), 24-28.

118. Mahl, R. - An Analytical Approach to Computer Systems Scheduling, University of Utah (June 1970).

119. Martinson,J.R. - Utilization of Virtual Memory in Time-Sharing System/360, IBM Tr.53.0001, IBM Systems Development Division, Yorktown Heights, N.Y. (Oct.1968).

120. Mattson,R.L. - Evaluation of Multilevel Memories, Memorandum, IBM Research Laboratory, San-Jose, Calif., 1971.

121. Mattson,R.L.,J.Gecsei,D.R.Slutz şi J.L. Traiger - Evaluation Techniques for Storage Hierarchies, IBM Syst.J. 9, 2 (1970), 78-117.

122. Mazda,F. - Large Scale Integrated Memories - Electronic Components, 53, 13 (Sept.1972), 870-884.

123. Meade,R.M. - On Memory System Design,FJCC 3 (1970),33-43.

124. Meade,R.M. - How a Cache Memory Enhances a Computer's Performance, Electronics, (Jan.1972), 58-63.

125. Meyer,R.A. şi L.H.Seawright - A Virtual Machine Time-Sharing System, IBM Syst.J.,9, 3 (1970),199-218.

126. Morenoff,E.şi J.B.Mclean - Application of Level-Changing to a Multilevel Storage Organization, CACM 10, 3 (March 1967), 149-154.

127. Mosuda,T.,H.Shiota,M.Noguchi şi T.Ohki - Optimization of Program Organization by Cluster Analysis, Information Processing 74, North Holland Publishing Co, Amsterdam 1974, 261.

128. Oden,P.H. şi G.S.Shedler - A Model of Memory Contention in a Paging Machine, CACM, 15, 8 (Aug.1972),761-771.

129. O'Neil,R.W. - Experience Using a Time-Sharing Multi-Programming System with Dynamic Address Relocation Hardware, Afips Conf.Proc.30, SJCC (1967), 611-621.

130. O'Neil,R.W.şi B.H.Sams - Preplanned Vs.Dynamic Storage Allocation Techniques,CACM 4, 10 (Oct.1961)416-418.

./.

131. Oppenheimer,G. şi N.Weizer - Resource Management for a Medium Scale Time-Sharing Operating System, CACM, 11, 5 (May 1968), 313-322.

132. Opran,M. - Memoriile Calculatoarelor Electronice, Sinteză Documentară, I.D.T., 1972.

133. Organick,E.I. - The Multics System - MIT Press, Cambridge, Mass.(1972).

134. Penny,B.J., R. Fink şi M. Alston - Garnjost - Design of a Very Large Storage System, FJCC 37 (1970), 45-51.

135. Ramamoorthy,C.V. - The Analytic Design of a Dynamic Look-Ahead and Program Segmenting System for Multiprogrammed Computers, Proc.21 Nat.Conf. ACM (1966).

136. Ramamoorthy,C.V. şi K.M.Chandy - Optimization of Memory Hierarchies in Multiprogrammed Systems, JACM 17, 3 (July 1970), 426-445.

137. Randell,B. - A Note on Storage Fragmentation and Program Segmentation, CACM, 12, 7 (July 1969), 365-372.

138. Randell,B. şi C.J. Kuehner - Dynamic Storage Allocation Systems, CACM 11, 5 (May 1968), 297-306.

139. Rege, S.L. - Performance and Power Dissipation Analysis for CCD Memory Systems, Proceedings of the 1976 NCC, 381-391.

140. Rodriguez,R.J. - Experimental Data on how Program Behavior Affects the Choice of Schedular Parameters, 3-rd ACM Symposium on Operating System Principles, ACM Publications, Stanford Univ., Stanford, Calif., (oct.1971), 156-163.

141. Rodriguez,R.J. şi J.P. Dupuy - The Design, Implementation and Evaluation of a Working Set Dispatcher, CACM 16 (1973),247.

142. Rodriguez,R.J. - Empirical Working Set Behavior, CACM, 16 (1973), 556.

143. Rogojan,Al. - Curs de Calculatoare Numerice, vol.1 şi 2, Ed. I.P.T., Timişoara, 1973.

144. Ryder, K.D. - Optimizing Program Placement in Virtual Systems, IBM Syst,J. 13 (1974), 292.

145. Sayre,D. - Is Automatic Folding of Program Efficient Enough to Displace Manual?, CACM 12, 12 (Dec.1969), 656-660.

146. Scarrot,G. - Optical Memory,The Radio and Electronic Engineering, 40, 2 (Aug.1970), 89-95.

147. Schwetman,H.D.- A Study of Resource Utilization and Performance Evaluation of Large Scale Computer Systems, TR-12, Computation Center, the University of Texas at Austin,July 1970.

148. Seligman,L. - Experimental Data for the Working Set Model, MIT Project MAC, Computation Structures Group, Memo No 39, MIT, Cambridge, Mass. (March 1968).

149. Shedler,G.S.şi C.Tung - Locality in Page-Reference Strings, SIAM J.On Computing 1, 3 (1972) 218-241.

./.

150. Shedler,G.S. şi S.C.Yang - Simulation of a Model of Paging System Performance, IBM Syst.J. 2 (1971).

151. Shemer,J.E. şi G.A. Shippey - Statistical Analysis of Paged and Segmented Computer Systems, IEEE T-EC, EC-15, 6 (Dec.1966).

152. Sherman,S., F.Basket şi J.C.Browne - Trace Driven Modelling and Analysis of CPU Scheduling in a Multiprogramming System, Proceedings ACM-Sigops Workshop on System Performance Evaluation, Cambridge, Mass., (April 1971).

153. Smith,J.L. - Multiprogramming under a Page on Demand Strategy, CACM 10, 10 (Oct.1967), 636-646.

154. Spain,K.J., H.I.Jauvtis şi P.T.Durben - Dot Memory Systems, Proceedings of the 1974 NCC, Afips, vol.43, 841-846.

155. Speliotis,D.D. - Bridging the Memory Access GAP, Proceedings of the 1975 NCC, 501-508.

156. Spirn,J.R. - A Model for Dynamic Allocation in a Paging Machine, 8th Annual Princeton Conference on Information Sciences and Systems, Princeton University (March 1974).

157. Thompson,S., J.A. Morton şi A. Bobeck - Memories-Future Storage Techniques, The Electronic Engineer (Aug. 1971), 33-39.

158. Traiger,I.L. şi R.L.Mattson - The Evaluation and Selection of Technologies for Computer Storage Systems, Magnetism and Magnetic Materials-1971, AIP Conference Proceedings nr.5, part 1, (1972), 1-9.

159. Varian,L.C. şi E.G.Coffman - An Empirical Study of the Behavior of Programs in a Paging Environment, ACM Symposium on Operating System Principles, Gatlinburg, Tennessee, (Oct.1967).

160. Vysotsky,V.A. - Structure of the Multics Supervisor, Afips Conf.Proc.27, FJCC (1965).

161. Waas,G.J. - Electronic Memories. A Review of Semiconductor Devices, Control Engineering, 19, 1 (Jan.1972), 57-63.

162. Wallace,V.L. şi D.L.Mason - Degree of Multiprogramming in Page on Demand Systems, CACM 12, 6 (June 1969), 305-310.

163. Walter,C.J., A.B.Walter şi L.J.Bohl - Impact of Fourth Generation Software on Hardware Design, IEEE Computer Group News (July 1968), 1-10.

164. Weizer,N. şi G.Oppenheimer - Virtual Memory Management in a Paging Environment, Afips, SJCC, Afips Press., Montvale,N.Y., 1969, 234-249.

165. Wilkes,M.V. - Slave Memories and Dynamic Storage Allocation, IEEE-TEC 14, 2 (April 1965), 270-271.

./.

166. Williams,J.G. - Large Core Storage in Perspective, Computer Design 11, 1 (Jan.1972), 45-49.

167. Winograd,J., S.J.Morgenstein şi R.Herman - Simulation Studies of a Virtual Memory, Time-Shared, Demand Paging Operating Systems, 3-rd ACM Symposium on Operating Systems Principles, ACM Publications, Stanford University, Stanford, Calif. (Oct.1971).

168. Woolf, A.M. - Analysis and Optimization of Multiprogrammed Computer Systems Using Storage Hierarchies, University of Michigan, Ann Arbor, Michigan, Systems Engineering Laboratory, Sel Technical Report nr.53 (April 1971).

169. Ypma,J.E. - Bubble Domain Memory Systems, Proceedings of the 1975 NCC, 523-528.

170. xxx - Single-Chip Microcomputers.Focus of Intense Price Battle,'Electronic News, vol.22, no.1197, Londay, nov.7, 1977.