Implementarea paralelismului la nivel de instructiune în microprocesoarele superscalare

Prof.dr.ing. Gheorghe DODESCU
Catedra de Informatica Economica, A.S.E. Bucuresti
Lec.ing. Bogdan OANCEA
Universitatea "Artifex" Bucuresti

Tendinta dominanta a acestui sfârsit de deceniu în domeniul microprocesoarelor o constituie introducerea paralelismului la nivel de instructiune. Apar astfel procesoarele superscalare, procesoare care sunt capabile sa executa mai multe instructiuni simultan. În articolul prezent vom prezenta câteva din tehnicile cele mai noi de procesare a instructiunilor utilizate în microprocesoarele superscalare.
Cuvinte cheie: *arhitectura microprocesoarelor,microprocesoare superscalare, procesoare ILP.*

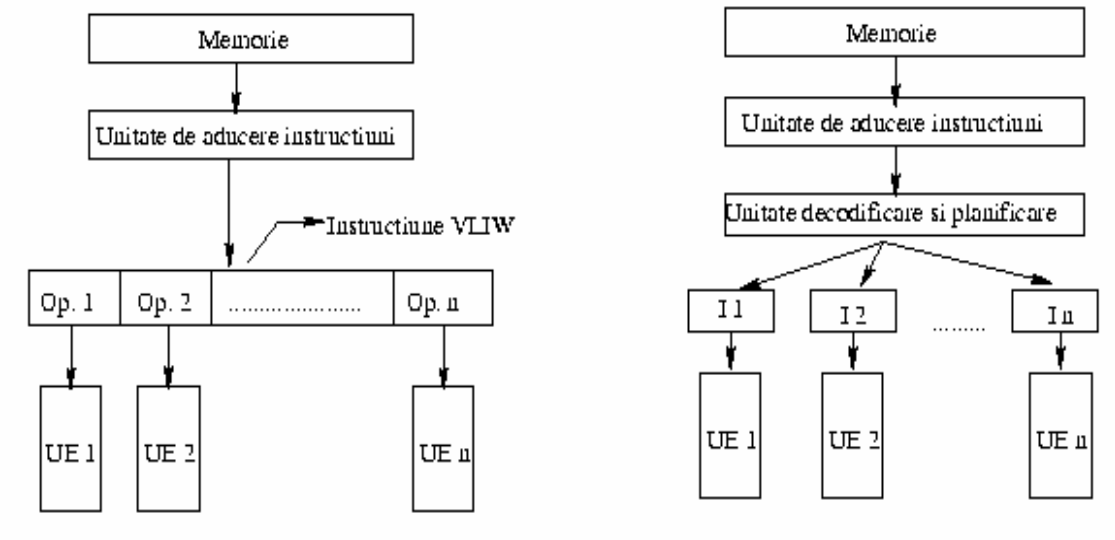
1. Introducere

Ca urmare a aparitiei unor aplicatii din ce în ce mai complexe, cerintele de putere de procesare au cunoscut o evolutie exponen-tiala. Pentru a putea face fata acestor cerinte, arhitectii sistemelor de calcul si-au canalizat eforturile de cercetare pe doua aspecte distincte: îmbunatatirea tehnologiilor în domeniul microelectronicii si introducerea unor noi concepte în structura functionala a procesoarelor. Primul aspect a condus la obtinerea unor dispozitive electronice mai performante, caracterizate prin scara de integrare mai mare, frecvente de ceas mai mari, în timp ce al doilea aspect a condus la aparitia unor noi tipuri arhitecturale de procesoare: procesoarele cu paralelism la nivel de instructiune.

Paralelismul la nivel de instructiune a fost introdus fie prin utilizarea metodei de prelucrare în banda de asamblare (pipeline), fie prin folosirea mai multor unitati func-tionale care lucreaza simultan. Acest al doilea tip de paralelism s-a concretizat prin aparitia procesoarelor VLIW si a proce-soarelor superscalare. Tendinta dominanta a acestui sfârsit de deceniu o constituie dominatia procesoarelor superscalare care se pare ca au câstigat în lupta cu masinile VLIW.

În figura 1 se poate observa modul de functionare al procesoarelor VLIW si superscalare. Masinile VLIW sunt construite având la baza o instructiune foarte lunga (între 256 si 1024 de biti) care este împartita în mai multe câmpuri, fiecare câmp repre-zentând codul unei operatii pentru cele *n* unitati functionale. Aceasta modalitate de abordare presupune un compilator foarte complicat care sa poata detecta mai multe instructiuni independente pentru a le grupa într-o instructiune VLIW.

Modelul procesoarelor superscalare se ba-zeaza pe un flux secvential de instructiuni obisnuite (RISC sau CISC), unitatea de decodificare si planificare având sarcina de a rezolva dependentele între instructiuni si de a trimite spre executie numai acele instructiuni care sunt independente. În acest caz se pot folosi compilatoare obisnuite, dar o anumita optimizare a codului din partea compilatorului va usura sarcina micropro-cesorului.



a) procesoare VLIW b)procesoare superscalare

Fig. 1. Procesoare VLIW si superscalare

Tabelul 1 prezinta cele mai cunoscute procesoare superscalare comerciale.

Tabelul 1. Microprocesoare superscalare

Firma producatoare	Anul aparitiei							
	1989	1990	1991	1992	1993	1994	1995	1996
INTEL	960CA		960MM					
					Pentium		PentiumPro	
IBM		Power1 RS/6000			Power2			
Motorola					MC68060			
DEC				Alfa 21064		Alfa 21064A	Alfa 21164	
Sun				Super Sparc			Ultra Sparc	
MIPS						R8000		R10000
AMD							AMD 29000	
							K5	
HP				PA7100			PA7200	PA8000
PowerPC					PowerPC 603	PowerPC604	PowerPC 620	

Dupa cum se observa din datele de mai sus, primele microprocesoare superscalare care au aparut pe piata au fost cele RISC, microprocesoarele CISC aparând câtiva ani mai târziu datorita complexitatii lor mai mari. Tendinta dominanta este spre dezvol-tarea de microprocesoare RISC datorita performantelor superioare ale acestora. Chiar si microprocesoarele CISC cele mai cunoscute (PentiumPro, K5) sunt construite pe baza unui nucleu RISC.

Caracteristica de baza a acestor microprocesoare este faptul ca ele executa mai multe instructiuni simultan. De aici apar si proble-me specifice care nu erau prezente în microprocesoarele scalare. Cele mai importante aspecte legate de procesoarele superscalare care vor fi tratate în lucrare se refera la: *decodificarea paralela, distributia spre executie, executarea paralela a instructiunilor, mentinerea secventialitatii programelor si a tratarii exceptiilor.*

2. Decodificarea

Decodificarea în microprocesoarele superscalare este mult mai dificila deoarece acestea trebuind sa prezinte spre executie mai multe instructiuni într-un tact de ceas, numarul de operatii necesitat de verificarea dependentelor între instructiuni este mai mare. Astfel, trebuie verificate dependentele între instructiunile candidate pentru noua planificare în urmatorul tact de ceas, dar si între acestea si cele care se afla deja în executie. Solutia acceptata tot mai larg este utilizarea unei faze de predecodificare atunci când instructiunile sunt aduse din cache-ul de nivel 2 în cel de nivel 1, în aceasta etapa adaugându-se fiecarei instructiuni un numar de biti care specifica clasa din care face parte instructiunea, ce resurse necesita pen-tru executie etc. În tabelul 2 sunt prezentate o serie de microprocesoare care folosesc aceasta tehnica.

Tabelul 2. Microprocesoare care folosesc predecodificarea

Denumire	Anul aparitiei	Numarul. de biti adaugati la fiecare instructiune la predecodificare
PA7200 (2)*	1995	5
PA8000 (4)	1996	5
PowerPC 620 (4)	1995	7
UltraSparc (4)	1995	4
R10000	1996	4
AMD K5 (~2)	1995	5**

* între paranteze este trecuta rata de distributie a instructiunilor

** K5 care este un procesor CISC adauga 5 biti la fiecare byte al instructiunii

3. Distributia spre executie

Cea mai importanta problema care trebuie rezolvata la procesoarele superscalare este distribuirea spre executie a mai multor instructiuni într-un singur tact de ceas. Acest aspect impune tratarea *dependentelor false între date*, a *dependentelor de control nerezolvate,* folosirea unor tehnici speciale pre-cum este tehnica *shelving* pentru evitarea blocajelor dar si stabilirea unor *metode de rezolvare a blocajelor* când ele apar.

Dependentele false între date, care sunt dependentele de tipul WAW sau WAR pot fi eliminate prin *redenumirea registrilor*. Bineînteles, prin aceasta metoda se pot elimina dependentele între datele reprezenta-te prin registri, dependente care sunt si cele mai des întâlnite.

Redenumirea registrilor are loc dinamic, în timpul executiei instructiunilor si în prezent aceasta metoda a fost introdusa în aproape toate modelele de microprocesoare. La început, s-a practicat redenumirea doar pentru anumite instructiuni cum ar fi spre exemplu Power1 (RS/6000) care folosea aceasta tehnica numai pentru instructiunile în virgula mobila de tip *load*. Cu timpul aceasta metoda s-a extins pentru toate tipurile de instructiuni. Exista mai multe variante de implementare a acestei tehnici, variante prezentate în continuare.

O prima modalitate de implementare a redenumirii registrilor o reprezinta folosirea unui set comun de registri arhitecturali si registri

pentru redenumire. În aceasta varianta, fiecare registru destinatie care apare într-o instructiune este alocat într-un registru fizic din setul comun de registri, setul de registri arhitecturali aparând în acest fel ca o multime care se schimba în mod dinamic. Alocarea registrilor este urmarita folosind o tabela de mapare dupa cum se poate observa în figura 2.

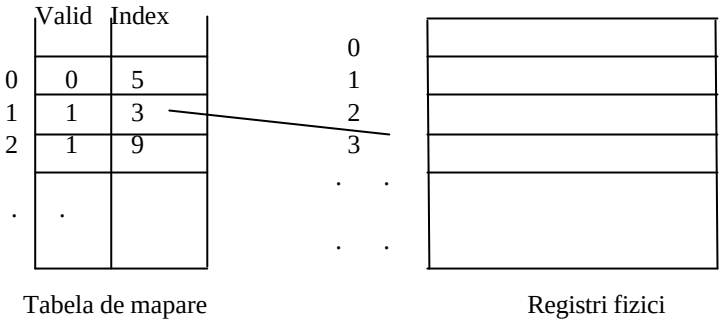


Fig. 2. Folosirea tabelei de mapare a registrilor

Tabela de mapare contine în primul rând un bit care indica daca intrarea respectiva este valida sau nu, apoi indexul registrului arhitectural în setul de registri fizici. În exemplul nostru registrul arhitectural R1 este alocat în registrul fizic RF3.

Alte modalitati de a implementa redenumi-rea registrilor presupune folosirea unor seturi separate de registri arhitecturali si registri de denumire, folosirea buffer-ului de reordonare a instructiunilor (ROB) sau a buffer-ului de redenumire, reordonare si shelving (DRIS). În aceste ultime trei cazuri, accesarea valorilor din buffer-ele de redenumire se poate face folosind o tabela de mapare ca în exemplul de mai sus, sau folosind o cautare asociativa în acest buffer. Tabelul 3 prezinta sintetic informatiile legate de redenumirea registrilor.

Al doilea tip de dependente care trebuie rezolvate sunt *dependentele de control nere-zolvate*. Aceste dependente sunt introduse de instructiunile de salt conditionat pentru care conditia de salt nu a fost evaluata înca la momentul executarii instructiunii de respec-tive. Tratarea eficienta a instructiunilor de salt presupune detectarea lor din timp. Procesoarele din linia DEC Alfa precum si Power PC 601 detecteaza si decodifica salturile în paralel cu decodificarea obisnuita pe când Power1, Power2 si PowerPC 603 folosesc o tehnica denumita *look-ahead branch detection* în care instructiunile de salt sunt detectate si decodificate înaintea instructiunilor obisnuite. Cea mai avansata metoda de detectare a instructiunilor de ramificare presupune ca la aducerea instruc-tiunilor din cache în buffer-ul de instructiuni sa fie detectat daca urmatoarea instructiune este de salt sau nu, iar în caz afirmativ se vor aduce din cache si instructiunea de la adresa de salt si cea care ar urma în ordinea secventiala. Aceasta metoda este utilizata de PowerPC 604, 620, R8000, PA8000.

Tabelul 3. Redenumirea registrilor în procesoare superscalare

Denumire	Metoda utilizata	Numarul de buffere de redenumire	
		Instr. V.F.	Instr. V.M
Power1	Set comun de registri arh. si de redenumire	-	32 arh./8 reden.
Power2	Set comun de registri arh. si de redenumire	-	32 arh./22 reden.
R10000	Set comun de registri arh. si de redenumire	32 arh./32 reden.	32arh./32reden.
PowerPC 604	Seturi separate de registri	12	8

PowerPC 620	Seturi separate de registri	8	8
K5	Redenumire în ROB	16	
PentiumPro	Redenumire în ROB	40	

Datorita faptului ca la momentul executarii unei instructiuni de salt conditionat s-ar putea sa nu fie înca accesibil rezultatul testului, instructiunea de salt ar trebui blocata pâna la aflarea rezultatului ceea ce ar impune o penalitate foarte mare în timpul de executie. Pentru a reduce aceste penalitati, se foloseste o metoda de predictie a saltului, instructiunile fiind executate pe ramura prezisa. În cazul în care s-a dovedit ca predictia a fost eronata toate aceste instructiuni executate speculativ vor fi anulate. Metodele de predictie pot fi împartite în *predictie fixa* si *predictie reala.*

În predictia fixa, instructiunea de salt va fi tratata în acelasi mod întotdeanua: fie va fi efectuat saltul (MC68040) fie nu (i486, SuperSparc, Power1, Power2, DEC Alfa 21064). Aceasta este si cea mai simpla metoda dar si cea mai ineficienta.

Metodele de predictie reala (fie se efectu-eaza saltul fie nu, dar nu întotdeauna decizia este aceeasi) pot fi la rândul lor de mai multe tipuri: *predictie statica* sau *predictie dina-mica*. Predictia statica se bazeaza fie pe codul operatiei (pentru unele instructiuni se efectueaza saltul iar pentru altele nu, ca în cazul PowerPC601,603) fie pe deplasamen-tul adresei de salt (daca este negativ se efectueaza saltul iar în caz contrar nu se efectueaza, ca la DEC Alfa 21064) fie pe un bit care este setat sau nu în codul operatiei de catre compilator (exemple: PA 8000, PowerPC 603).

Predictia dinamica implica memorarea isto-riei celor mai recente salturi si în functie de comportamentul trecut se va prezice si comportamentul actual. În functie de numa-rul de biti utilizati la memorarea compor-tarilor anterioare distingem scheme de pre-dictie cu 1 bit, cu 2 biti sau cu 3 biti. Cea mai simpla metoda, cea cu un singur bit, presupune memorarea valorii 1 sau 0 în cazul când la ultima executie a instructiunii a avut loc sau nu saltul. Daca bitul are valoarea 1 se va efectua si în prezent saltul, urmând ca ulterior, când conditia de salt a fost evaluata sa se actualizeze în mod corespunzator acest bit. Metodele cu 2 respectiv 3 biti sunt într-un fel asema-natoare, dar, putând stoca mai multe infor-matii despre istoria salturilor aceste metode sunt mult mai performante.

Pentru a nu introduce o penalitate mare în cazul unei predictii gresite toate procesoa-rele dispun de mecanisme de restaurare a starii initiale prin pastrarea în buffere speciale si a instructiunilor de pe ramura secventiala si pe cele de pe ramura de salt.

Clasificarea metodelor de predictie a salturilor poate fi urmarita în tabelul 4.

Tabelul 4. Clasificarea metodelor de predictie a salturilor

Predictie fixa	Executa saltul întotdeauna	MC68040	
	Nu executa saltul niciodata	i486, R4000, SuperSparc,Power1, Power2, DECAlfa 21064	
Predictie reala	Predictia statica	Bazata pe codul operatiei	PowerPC 601, PowerPC 603
		Bazata pe deplasamentul adresei de salt	DEC Alfa 21064A
		Bazata pe indicatii date de compilator	PA8000, PowerPC 603
	Predictia	1 bit	R8000
		2 biti	MC68060,PowerPC604, 620, UltraSparc, DEC Alfa 21164, Pentium,R10000,K5,K6,M2

	dinamica	3 biti	PA8000

O tehnica speciala folosita în procesoarele recente pentru a evita aparitia blocajelor atunci când apar dependente între date este folosirea unor buffere de instructiuni plasate imediat în fata unitatilor de executie, denu-mite buffere de *shelving*. Astfel de buffere se pot introduce fie în fata fiecarei unitati de executie, fie un singur buffer poate servi mai multe unitati de executie, fie se poate introduce un buffer care sa serveasca toate unitatile de executie, dupa cum putem observa în figura 3.

Aceste buffere poarta denumirea de statii de rezervare (individuale, de grup, respectiv centrale). O alternativa la folosirea statiilor de rezervare este folosirea unui singur buffer pentru shelving, redenumirea registrilor si reordonarea instructiunilor denumit DRIS (**D**effered scheduling, **R**egister renaming and **I**nstruction **S**helve).

Când este folosita aceasta tehnica, distributia instructiunilor spre statiile de rezervare este mult simplificata deoarece verificarea de-pendentelor între instructiuni este distribuita spre fiecare statie în parte urmând ca ea sa fie efectuata în faza a doua de planificare si trimitere a instructiunilor spre executie.

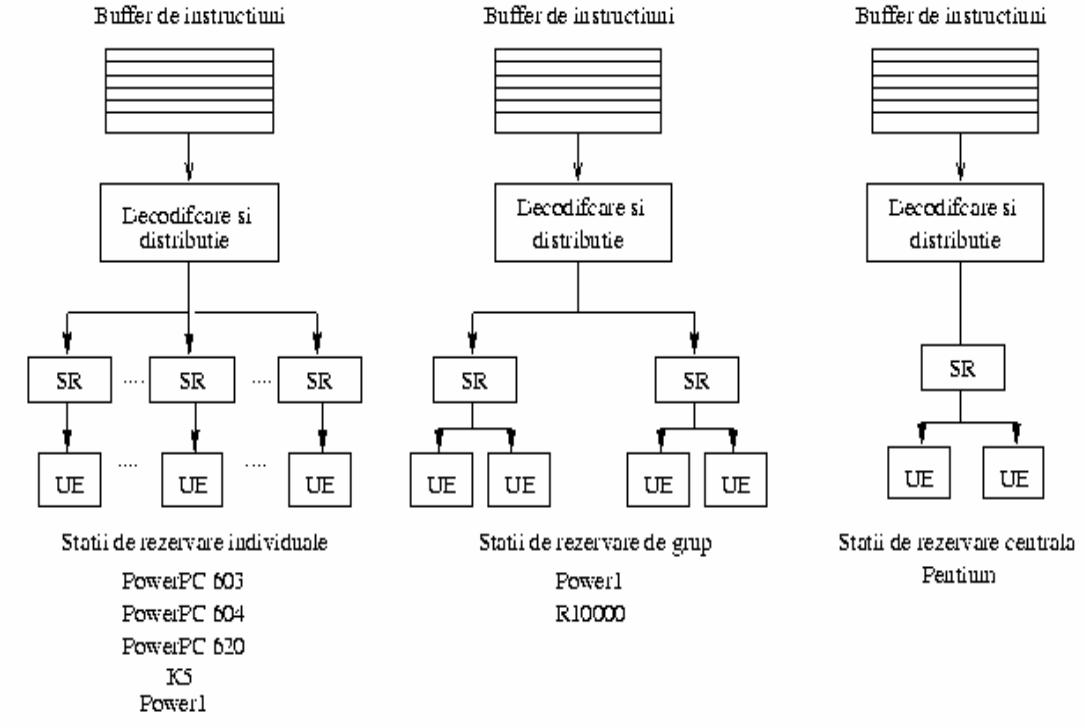


Fig. 3. Statii de rezervare

Numarul de intrari în statiile de rezervare variaza de la procesor la procesor. Cele care folosesc statii de rezervare individuale vor avea un numar de intrari mai mic (spre exemplu PowerPC 603 are 3 intrari, PowerPC 620 are 20 de intrari) pe când cele care folosesc statii de rezervare de grup sau centrale vor trebui sa aiba mai multe intrari (R10000 are 48 de intrari iar Pentium are 20 astfel de intrari). Problema care apare aici este ca numarul mare de intrari impune si un numar mai mare de porturi de citire/scriere în statiile de rezervare.

Instructiunile prezente în statiile de rezerva-re sunt verificate pentru detectarea depen-dentelor, cele independente fiind selectate pentru înaintarea lor catre unitatile de executie. Daca sunt mai multe astfel de instructiuni decât numarul unitatilor de executie disponibile atunci

se aleg de obicei acele instructiuni care au stat perioada cea mai îndelungata în statia de rezervare.

Modul cum este tratata o instructiune dependenta atunci când se face planificarea pentru executie conduce la identificarea a trei scheme de planificare:

-planificare "*in-order*" - o instructiune neexecutabila va duce la blocarea instruc-tiunilor ulterioare de la planificare chiar daca acestea se pot executa;

-planificare "*out-of-order*" - o instructiune care nu se poate executa la momentul actual nu blocheaza de la planificare alte instruc-tiuni ulterioare dar care se pot executa;

-planificare partiala "*out-of-order*" - anumite statii de rezervare practica planificare in-order altele practica planificarea out-of-order.

În tabelul 5 sunt date exemple de procesoare care implementeaza tehnicile de planificare amintite mai sus.

Tabelul 5. Metode de planificare a instructiunilor

Planificare *in-order*	Planificare *out-of-order*	Planificare *partiala out-of-order*
Power1	PentiumPro	Power2
PowerPC 603	R10000	PowerPC 604
	PA 8000	PowerPC 620

Un ultim aspect care trebuie adus în discutie aici este rata instructiunilor planificate spre executie. Procesoarele ce folosesc statii de rezervare individuale vor avea aceasta rata egala cu unitatea dar cele cu statii de rezervare de grup si centrale vor trebuie sa fie capabile sa planifice mai multe instructiuni într-un tact de ceas. În tabelul 6 se pot observa ratele de planificare a instructiunilor din buffer-ul de instructiuni în statiile de rezervare precum si cea de planificare a instructiunilor din statiile de rezervare spre unitatile de executie pentru procesoare care folosesc tehnica de *shelving*.

Tabelul 6. Rate de planificare a instructiunilor pentru diferite procesoare

Procesorul	Rata de planificare spre statiile de rezervare (instructiuni/ciclu)	Rata de planificare spre unitatile de executie (instructiuni/ciclu)
K5	~2(4)*	5
PentiumPro	~2(4)*	5
PA 8000	4	4
PowerPC 603	3	3
PowerPC 604	4	6
PowerPC 620	4	6
Power2	6	10
R10000	4	5

* aproximativ 2 instructiuni CISC ceea ce este echivalent cu 4 instructiuni RISC

Pentru tratarea situatiei când apar blocaje de planificare (care pot apare chiar daca se foloseste tehnica *shelving*) trebuie avute în vedere doua probleme: ordinea de planifica-re, despre care am discutat anterior si alinierea planificarii care va fi prezentata în continuare.

În figura 4 avem reprezentata o situatie în care procesorul poate planifica 5 instructiuni simultan. Ultimele 5 instructiuni din buffer-ul de instructiuni care sunt candidate pentru urmatoarea planificare formeaza fereastra de planificare (figura 4a). În functie de cum este luata în considerare aceasta fereastra de instructiuni atunci când apar dependente între instructiuni distingem doua situatii: planificare cu fereastra fixa sau planificare cu fereastra glisanta. Instructiunile din doua ferestre consecutive sunt reprezentate în figura 4b, cele

încercuite fiind instructiuni dependente. În figurile 4c si 4d se pot vedea cele doua modalitati de planificare: cu fereastra fixa (4c) si cu fereastra mobila (4d). Cu toate ca metoda a doua este mai performanta, majoritatea procesoarelor (Po-werPC, DEC Alfa, PA 8000, R10000) folosesc prima metoda de planificare.

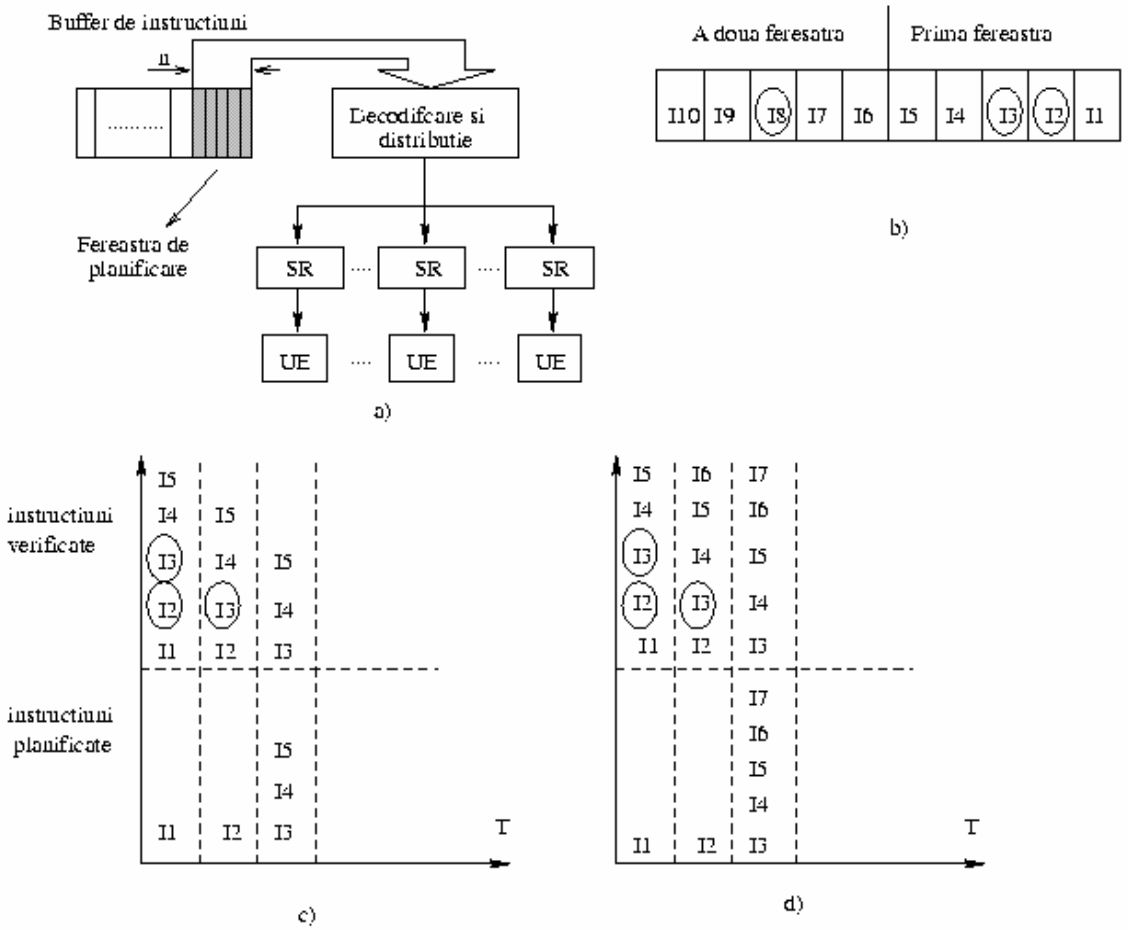


Fig. 4. Alinierea ferestrei de planificare a instructiunilor

Concluzionând discutia despre planificarea instructiunilor, daca luam în considerare doar trei aspecte mai importante, redenumi-rea registrilor, executia speculativa (predic-tia salturilor) si folosirea tehnicii shelving, putem obtine clasificarea din tabelul 7.

Tabelul 7. Tehnici de planificare a instructiunilor

Fara redenumire Executie speculativa Fara shelving	Fara redenumire Executie speculativa Cu shelving	Cu redenumire Executie speculativa Fara shelving	Cu redenumire Executie speculativa Cu shelving
Pentium			PentiumPro, K5
PowerPC 601		PowerPC 602	PowerPC 603,604,620
PA 7200			PA 8000
SuperSparc,UltraSparc			
Dec Alfa 21064, 21064A, 21164			
	R 8000		R10000

4. Executia paralela si mentinerea secven-tialitatii programelor

Pentru a permite executarea în paralel a instructiunilor planificate toate procesoarele suprescalare poseda mai multe unitati functionale. Astfel PentiumPro este dotat cu 10 unitati functionale, MIPS R10000 cu 7 iar PowerPC 620 cu 6 unitati functionale. Deoarece nu toate unitatile functionale au aceeasi viteza de executie (de exemplu o împartire în VM va dura mai mult decât o adunare în VF) instructiunile care la un moment dat sunt în faza de executie se pot termina în alta ordine decât ordinea secven-tiala în care ele apar în program. Pentru a mentine corectitudinea programului, proce-sorul trebuie sa rezolve aceasta problema a mentinerii secventialitatii instructiunilor.

Majoritatea procesoarelor trateaza acest aspect prin folosirea unui buffer de reor-donare a instructiunilor (ROB). Bufferul de reordonare este organizat ca o lista circulara în care sunt introduse toate instructiunile, de la planificarea lor pentru executie pâna la obtinerea rezultatelor finale. Fiecare intrare în lista, intrare ce reprezinta o instructiune are un indicator de stare: instructiunea este planificata pentru executie, este în executie sau a terminat executia. Instructiunile care si-au terminat executia nu sunt lasate sa actualizeze starea masinii pâna când nu le vine rândul sa paraseasca acest buffer. Parasirea bufferului se face în ordinea sec-ventiala a programului, la parasire rezulta-tele instructiunilor fiind stocate în registrii sau în memorie.

Buffer-ul de reordonare se foloseste si pen-tru tratarea întreruperilor. Când în cursul executiei unei instructiuni se genereaza o exceptie, aceasta nu este tratata pâna când instructiunea care a generat-o nu devine ultima din lista circulara, venindu-i astfel rândul sa paraseasca buffer-ul si asigurându-ne în acest fel ca toate instructiunile anterioare au actualizat starea procesorului.

Cu toate ca este prezent în aproape toate procesoarele, buffer-ul de reodonare poarta nume diferite. La PentiumPro, PowerPC 604, 620, K5 el se numeste *reorder buffer* în timp ce la R10000 se numeste *active list,* la UltraSparc *completion unit,* iar la PA 8000 *instruction reorder buffer*.

Bibliografie

1. MIPS Tehnologies Inc., Microprocesor Product overview, 1994.
2. John L. Hennesy, David A. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann Publishers, Inc, 1996.
3. Dezso Sima, Terence Fountain, Peter Kacsuk, Advanced Computer Arhitecture, Addison-Wesley 1997.
4. Kai Hwang, Faye A. Briggs, Computer Arhitecture and Parallel Procesing, McGraw-Hill, 1985.
5. Kai Hwang, Advanced Computer Arhitecture, McGraw-Hill, 1993.
6. Hans Peter Messmer, The Indispensable PC Hardware Book, Addison-Wesley, 1995.
7. L. Gwennap, PA-8000 combines complexity and speed, Microprocesor Report, 8(15), 1996.