Tehnici de prelucrare a instructiunilor de transfer al controlului

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

Instructiunile de transfer al controlului au pus mari probleme proiectantilor de procesoare datorita penalitatilor mari introduse de acestea. Folosirea tehnicii de prelucrare pipeline si aparitia procesoarelor superscalare a impus gasirea unor modalitati de tratare eficienta a acestui tip de instructiuni. Articolul prezinta cele mai des utilizate tehnici de predictie a salturilor întâlnite în procesoarele moderne.
Cuvinte cheie: *procesor, instructiuni de salt, tehnici de predictie.*

Instructiunile de salt reprezinta un caz special de instructiuni. Toate procesoarele moderne dispun de tehnici speciale pentru prelucrarea acestor instructiuni. Vom arata în primul rând de ce se impun astfel de tehnici speciale pentru instructiunile de salt. Pentru aceasta vom pleca de la constatarea ca în prezent toate procesoarele utilizeaza metoda de prelucrare în banda de asamblare (pipeline). Consideram un caz simplificat de procesor pipeline în care avem doar 4 nivele: aducerea instructiunii din cache (Fetch), decodificarea ei (D), executia instructiunii (EX) si în final scrierea rezul-tatului în registrul destinatie (WB). Trebuie mentionat ca acest caz nu este departe de realitate, multe procesoare RISC având imple-mentat un astfel de pipeline la nivelul unitatii de prelucrare a instructiunilor. Presupunem ca avem de executat cu acest procesor fluxul de instructiuni I1, I2, Is, I3, I4,I5...It, unde Ii sunt instructiuni obisnuite, Is este o in-structiune de salt neconditionat iar It este instructiunea unde se efectueaza saltul. Mai facem ipoteza ca la sfârsitul fazei de executie a instructiunii de salt este disponibila adresa la care se va efectua saltul. Putem urmari modul de executie al acestor instructiuni în figura 1.

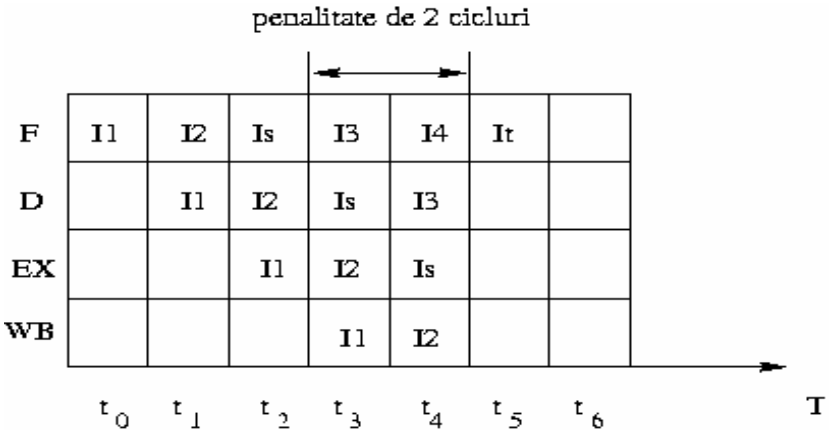

Fig. 1. Executia unei instructiuni de salt

Din momentul în care instructiunea de salt Is este în faza F si pâna când instructiunea de la adresa de salt (It) intra în aceeasi faza (F) trec doua cicluri de ceas care practic sunt irosite. Aceste

cicluri pierdute se numesc cicluri de penalitate la executarea unei instructiuni de salt. Dupa cum se poate usor observa din figura, instructiunile I3 si I4 care au fost deja admise în pipeline devin inutile si deci s-a pierdut timp cu prelucrarea lor.
Situatia se complica si mai mult daca banda de asamblare are mai multe nivele cum este cazul procesoarelor DEC Alfa 21164 si Pentium Pro care au 12 nivele sau MC68060 cu 9 astfel de nivele. O problema în plus apare în cazul instructiunilor de salt conditionat, unde evaluarea conditiei mai cere câtiva cicli procesor ceea ce duce la cresterea penalitatii. Pe de alta parte, în momentul executarii instructiunii de salt conditionat s-ar putea sa nu fie înca evaluata expresia care sta la baza conditiei, asa cum se întâmpla în procesoarele superscalare care poseda mai multe unitati functionale ce lucreaza independent. În acest caz spunem ca avem o instructiune de salt conditionat nerezolvata. Toate aceste aspecte pot conduce la o scadere sensibila a performantelor. Spre exemplu procesorul Intel 80386 are o penalitate de 8 cicli pentru instructiunile la care se executa salturile si 2 cicli pentru cele la care nu se executa saltul. Procesoarele moderne, care dispun de metode speciale de tratare a unor astfel de situatii au reusit sa reduca aceste penalitati pâna la 0 cicli în anumite cazuri sau 2...4 cicli în cel mai rau caz.
Atentia acordata acestui tip de instructiuni este impusa si de faptul ca instructiunile de salt conditionat sau neconditionat apar foarte des în programe. Statisticile demonstreaza ca în programele de uz general ponderea instruc-tiunilor de salt este de aproximativ 20-25% în timp ce în programele stiintifice este de 5-10%. Dintre acestea, aproximativ 75% dupa unii autori sau 83% dupa altii sunt instructiuni în care se efectueaza saltul iar în restul de 25% respectiv 17% din cazuri se continua pe calea secventiala. Procentul mare de instruc-tiuni în care se efectueaza saltul conduce la cautarea unor solutii arhitecturale care sa minimizeze penalitatile în acest caz.
Putem rezuma tehnicile de tratare a in-structiunilor de salt la urmatoarele :

- a) folosirea metodei "delayed branching"
- b) blocarea procesarii instructiunii de salt conditional nerezolvat
- c) executarea speculativa a instructiunilor de salt conditional nerezolvat
- d) executarea ambelor cai a unei instructiuni de salt (multiway branching)

***

a) Metoda "delayed branching" a fost prin-tre primele încercari de a reduce penalitatile care apar în cazul instructiunilor de salt. Ideea de baza este urmatoarea: daca ne amintim situatia prezentata în figura 1, constam ca imediat dupa ce instructiunea de salt a intrat în pipeline se pierd un numar de cicli. Putem reutiliza acesti cicli pierduti daca procesorul dispune de posibilitatea de a lansa în executie mai întâi instructiunea de salt apoi cea care o precede.
Concret, sa consideram secventa de instruc-tiuni:

```
        ...
    add R1,R2,R3;  R1<-R2+R3
     jmp @1
        ...
@1: mul R5,R6,R4;  R5<-R6*R4
```

Facem ipoteza ca adresa de salt va fi disponibila la sfârsitul fazei de decodificare, iar instructiunea de adunare se poate executa într-un singur ciclu. Atunci, daca înaintea instructiunii *add R1,R2,R3* se va executa instructiunea *jmp @1* se va putea recupera ciclul pierdut datorat saltului, dupa cum se poate observa si în figura 2.
Aceasta metoda presupune un ajutor din partea compilatorului care va trebui sa gaseasca o instructiune independenta care sa poata fi planificata pentru executie în modul prezentat în figura 2. Cu toate ca unele arhitecturi precum MIPS-X, MC88100, HP PA folosesc aceasta metoda, în prezent tendinta este de a se renunta la ea si de a folosi metoda de executie speculativa a instructiunilor.

b) Blocarea procesarii instructiunilor de salt nerezolvate este o metoda care s-a utilizat în primele procesoare pipeline (i386, MC68020) si presupune asteptarea rezolvarii conditiei de salt si abia apoi lansarea în executie a instructiunii de transfer a con-trolului. Datorita ineficientei, aceasta metoda a fost abandonata.

c) Executia speculativa este deocamdata cea mai avansata tehnica de procesare a instruc-tiunilor de salt.

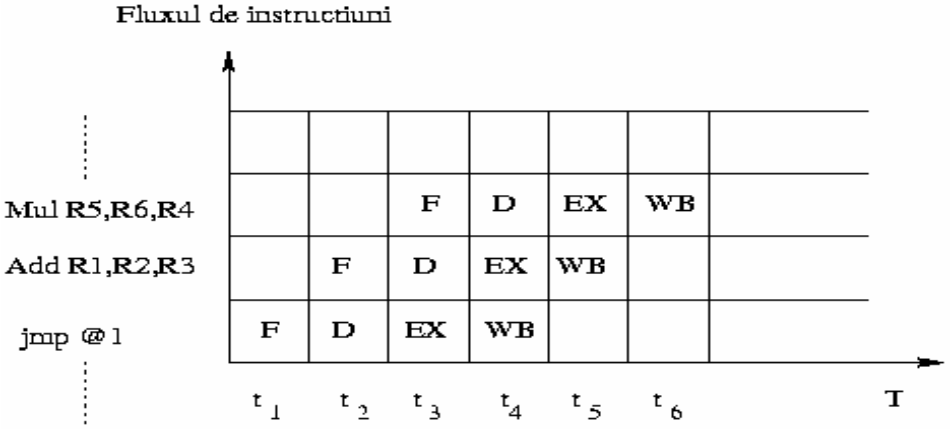

Fig. 2. Metoda "delayed branching"

Pentru a permite tratarea eficienta a instruc-tiunilor de salt trebuie ca ele sa fie detectate 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 instruc-tiunile de salt sunt detectate si decodificate înaintea instructiunilor obisnuite. Astfel, daca pentru decodificare si planificare spre exe-cutie se aduc *n* instructiuni/ciclu din memoria cache, pentru detectarea instructiunilor de salt, vor fi inspectate ultimele *n+k* intrari din memoria cache. Procesorul Power1 aduce 4 instructiuni/ciclu din memoria cache pentru decodificare, dar pentru detectarea salturilor inspecteaza ultimele 5 intrari din cache, în timp ce Power2 decodifica 6 instructi-uni/ciclu si verifica ultimele 8 intrari din cache pentru detectarea salturilor.

Cea mai avansata metoda de detectare a instructiunilor de ramificare presupune ca la aducerea instructiunilor din memoria cache în buffer-ul de instructiuni sa fie detectat daca urmatoarea instructiune este de salt sau nu, iar în caz afirmativ se va aplica o metoda de prezicere a saltului pentru a aduce din cache instructiunea de la adresa de salt sau cea care ar urma în ordinea secventiala. Aceasta metoda este utilizata de PowerPC 604, 620, R8000, PA8000.

Odata detectata o instructiune de salt, proce-sorul va încerca sa ghiceasca pe care ramura se va continua programul si în continuare se vor executa instructiunile de pe ramura prezisa. În momentul în care rezultatul evaluarii conditiei de salt devine disponibil, se va verifica daca prezicerea a fost corecta sau nu. Daca aceasta s-a dovedit a fi corecta atunci programul continua în mod normal, dar daca prezicerea a fost eronata, toate instruc-tiunile executate speculativ vor fi anulate si executia continua pe calea corecta.

Metodele de predictie a instructiunilor de salt se pot clasifica în metode de predictie *fixe* si metode de preditie *reale*.

Metodele fixe se caracterizeaza prin faptul ca instructiunile de salt vor fi tratate întotdeauna în acelasi fel: fie se prezice ca saltul se va efectua mereu, cum este cazul procesorului MC68040, fie se prezice ca nu se efectueaza saltul niciodata ci se continua pe calea secventiala, metoda aplicata la INTEL 486, SuperSparc, Power1,

Power2, DEC Alfa 21064. Aceste metode sunt mai simplu de implementat, dar acuratetea predictiei s-a dovedit a fi necorespunzatoare. De aceea majoritatea procesoarelor din ultima generatie folosesc metode de predictie reala.

Metodele de predictie reale pot fi de doua tipuri: metode *statice* si metode *dinamice*.

Metodele statice le putem separa mai departe dupa informatia care sta la baza prezicerii saltului. Distingem astfel 3 metode statice:

Metode bazate pe codul operatiei. Unele procesoare verifica un anumit bit din codul unei operatii si în functie de acesta va lua decizia. Un exemplu în acest sens este procesorul MC88110 care seteaza bitul 21 din instructiunile de comparare când conditia din instructiune este $\neq 0$, >0 sau ≥ 0. În aceasta situatie se va efectua saltul. Bitul 21 se va reseta atunci când conditia este $=0$, <0 sau ≤ 0 si saltul nu se va efectua.

Metode bazate pe directia în care se efectueaza saltul: daca saltul se efectueaza înainte (adresa de salt mai mare decât a instructiunii curente) saltul nu se efectueaza, iar în cazul când adresa de salt este mai mica, deci saltul are loc înapoi se va efectua saltul. Proce-soarele care implementeaza aceasta metoda sunt PowerPC 601, PowerPC 603, DEC Alfa 21064.

Metode bazate pe un bit din codul unei operatii care este setat sau nu de un compilator. În functie de o anumita constructie de limbaj, compilatorul poate prezice daca se va efectua saltul sau nu. Exemple: PowerPC 601, PowerPC 603, HP PA8000.

Metodele dinamice iau în considerare istoria executiei instructiunilor de salt. În functie de comportarea în trecut a unei instructiuni de salt se prezice si comportarea ei viitoare. Pentru a putea tine cont de istoria executiei instructiunilor de salt, trebuie memorata com-portarea acestor instructiuni. Deosebim în acest fel scheme de predictie cu 1 bit, cu 2 biti respectiv cu 3 biti de memorie.

Cea mai simpla, schema cu 1 bit memoreaza istoria ultimei executii: bitul va avea valoarea 1 daca saltul s-a executat si 0 în caz contrar. În momentul predictiei se verifica valoarea acestui bit: daca este 1 se va prezice saltul, daca este 0 se continua pe calea secventiala. În momentul în care conditia care a stat la baza saltului este evaluata valoarea bitului se va actualiza în mod corespunzator.

Metoda cu 2 biti poate fi asemuita cu functionarea unei masini cu stari finite. Cu ajutorul a 2 biti putem memora 4 stari : 00, 01, 10 si 11. Daca starea curenta în momentul când trebuie sa se prezica saltul este 00 sau 01 decizia este de a nu efectua saltul, iar în starea 10 sau 11 decizia este de a efectua saltul. În momentul când conditia de salt a fost evaluata, cei 2 biti se vor actualiza conform diagramei de tranzitie a starilor
din figura 3.

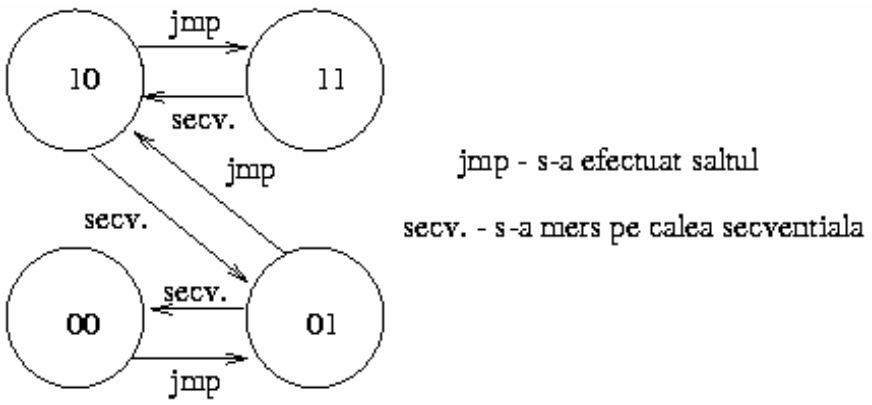

Fig. 3. Diagrama tranzitiilor de stare.

Simularile arata ca aceasta metoda are rezul-tate mult mai bune decât în situatia folosirii unui singur bit. Probabilitatea de prezicere corecta este în jurul valorii de 93% pentru 2 biti fata de numai 89% pentru 1 bit sau 68% în cazul metodelor statice.

Schema de predictie cu 3 biti presupune memorarea ultimelor 3 executii ale unei instructiuni de salt. În momentul când trebuie luata o decizie se numara bitii cu valoarea 1 si 0 iar decizia este luata pe baza majoritatii: mai multi de 1 înseamna salt, mai multi 0 - se continua ramura secventiala. Actualizarea acestor biti se face pe baza disciplinei FIFO.

În tabelul 1 sunt prezentate exemple de procesoare care implementeaza aceste tehnici de predictie a salturilor.

Predictie fixa	Executa saltul întotdeauna	MC68040	
	Nu executa saltul niciodata	i486, R4000, SuperSparc,Power1, Power2, DECAlfa 21064	
Predictie dinamica	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 dinamica	1 bit	R8000
		2 biti	MC68060,PowerPC604, 620, UltraSparc, DEC Alfa 21164, Pentium,R10000
		3 biti	PA8000

Tabelul 1. Clasificarea metodelor de predictie a salturilor

Executia speculativa a instructiunilor poate conduce la urmatoarea situatie: la întâlnirea unei instructiuni de transfer a controlului nerezolvate se prezice o cale de urmat, dar în timpul executarii instructiunilor de pe aceasta cale apare o alta instructiune de salt. Problema este câte astfel de instructiuni vor fi tratate speculativ. Cu cât sunt mai multe, cu atât vor fi mai mari performantele procesorului, dar în cazul unei predictii gresite cu atât va fi mai grea sarcina de refacere a starii initiale. Cele mai multe procesoare accepta tratarea specu-lativa doar a unei singure instructiuni de salt (DEC Alfa 21064, 21064A, PowerPC 601, 603) dar exista si procesoare care permit mai multor instructiuni de salt sa fie tratate speculativ: 2 instructiuni pentru Power2, 4 pentru PowerPC 620 si MIPS R10000 si chiar 6 pentru DEC Alfa 21164.

Pentru a face fata situatiei în care s-a facut o predictie gresita, majoritatea procesoarelor dispun de facilitati pentru a reduce penalitatile în astfel de cazuri. Metoda cea mai simpla este de a salva adresa de continuare secven-tiala în cazul în care s-a prezis efectuarea saltului sau de a începe precalcularea adresei de salt atunci când s-a prezis urmarea caii secventiale (PowerPC 601, 603, 620). O metoda mai avansata include chiar salvarea câtorva dintre instructiunile de pe calea secventiala în prima situatie sau a instruc-tiunilor de la adresa de salt în cea de-a doua situatie. Acest lucru implica existenta a mai multor buffere de instructiuni (Power1, Power2, Pentium).

Dupa cum am aratat mai sus, în aproximativ 75%-83% din instructiunile de salt se efectueaza saltul. De aceea trebuie luate masuri speciale pentru a micsora timpul de acces la instructiunea de la adresa de salt. Modalitatea cea mai simpla este ca la aparitia unei instructiuni de salt, adresa sa fie calculata de un sumator si apoi sa fie aduse

în cache instructiunile de la aceasta adresa. Exista însa si metode mai performante:

- BTAC (**B**ranch **T**arget **A**ddres **C**ache)
- BTIC (**B**ranch **T**arget **I**nstruction **C**ache)
- existenta în memoria cache a unui indice care sa arate instructiunea imediat urmatoare.

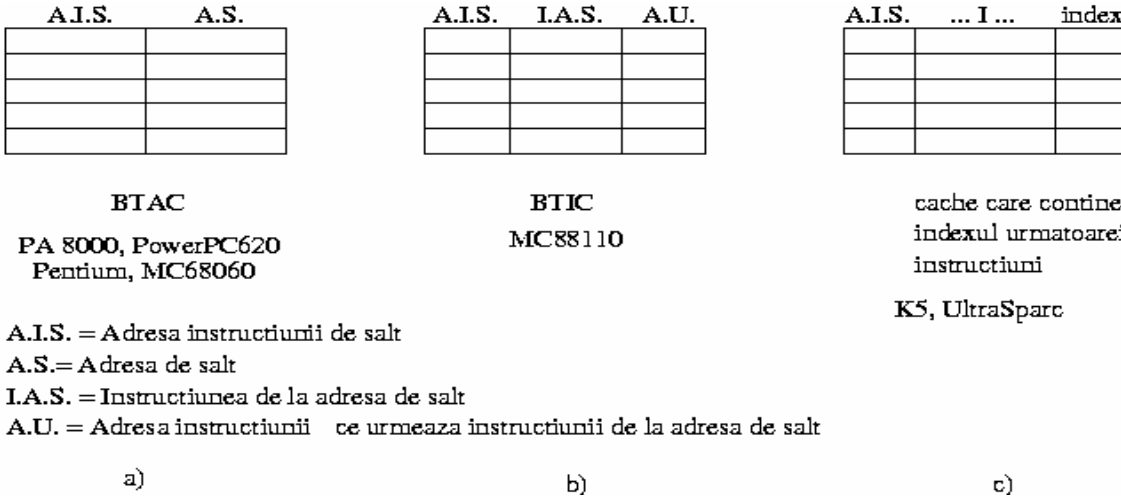

Fig. 4. Metode de accesare a caii de salt

Metoda BTAC consta în existenta unei me-morii cache suplimentare în care se pastreaza adresa unei instructiuni de salt precum si adresa la care se efectueaza saltul (figura 4 a). Aceasta memorie va fi accesata asociativ de fiecare data când urmatoarea instructiune este una de salt, iar daca se gaseste o intrare corespunzatoare în BTAC atunci se salveaza timpul pierdut la calcularea adresei de salt. Marimea BTAC variaza între 32 pâna la 4k intrari. Metoda BTIC este asemanatoare meto-dei precedente cu deosebirea ca în loc de a se mentine în memoria cache speciala adresa saltului, se mentine chiar instructiunea de la adresa de salt (figura 4 b).

Ultima metoda amintita aici, implica existenta unui câmp suplimentar în memoria cache pentru instructiuni , câmp care va puncta catre urmatoarea instructiune care trebuie adusa în buffer-ul de prefetch (figura 4 c).

Aceste tehnici pot ajuta chiar la luarea unei decizii cu privire la tratarea unei instructiuni de salt nerezolvata: o intrare în BTAC sau BTIC înseamna ca respectiva instructiune a mai fost întâlnita în cursul executarii pro-gramului si s-a efectuat saltul (deci seamana cu metoda de predictie cu 1 bit). Pentru a mari performantele, unele procesoare aplica sche-ma de predictie BTAC sau BTIC în combi-natie cu predictia cu 2 biti (PowerPC 604, 620, Pentium).

d) executarea ambelor cai ale unei instructiuni de transfer a controlului este o metoda foarte pretentioasa, mare consumatoare de resurse si deocamdata nu este aplicata decât în unele masini VLIW.

Bibliogafie

1. Dezso Sima, Terence Fountain, Peter Kacsuk, Advanced Computer Arhitecture, Addison-Wesley 1997.

2. Kai Hwang, Faye A. Briggs, Computer Arhitecture and Parallel Procesing, McGraw-Hill, 1985.

3. Kai Hwang, Advanced Computer Arhitec-ture, McGraw-Hill, 1993.

4. Hans Peter Messmer, The Indispensable PC Hardware Book, Addison-Wesley, 1995.

5. Marius Marcu, Microprocesoarele INTEL, PC Report, iunie 1998.

6. Marius Marcu, Procesoarele AMD, PC Report, septembrie 1998.