

Popa Catalin Gabriel.

Algoritmii lui Tomasulo.

Robert Marco Tomasulo s-a născut în anul 1934, 31 Octombrie și a fost un informatician american. Acesta a absolvit universitatea Manhattan, dedicându-și cariera în cercetare.

În anul 1956, s-a alăturat echipei de cercetare IBM. După o perioadă de 10 ani activitate, a trecut la dezvoltarea mainframe-urilor IBM/360 Model 14 și succesorii acestuia. După 25 de ani în cadrul IBM, Robert a lucrat la un important proiect pt. dezvoltarea primei mainframe-uri bazate pe CROS.

Acesta a fost câștigătorul premiului Eckert-Mauchly în anul 1997. Arhitectura lui R. Tomasulo este una de tip supercalar, cu mai multe unități de execuție iar algoritmul de control stabilește relația la o instrucțiune adusă, momentul în care acesta este lansată în execuție și respectiv unitatea de execuție care va procesa instrucțiunea.

Arhitectura permite execuție multiple și Out of Order. a. instrucțiunilor, de asemenea, algoritmul de gestionare referent arhitecturii permite ambarea hazardurilor WAR și RAW. Modul de funcționare al algoritmului este prezentat în felul următor: stătile de rezervare care preiau instrucțiunile din buffer.

Toate unitățile de execuție (Add, Mul) au asociate o stație de rezervare. Operațiile ADD efectuează operații de adunare, scădere, iar MUL operații de înmulțire și împărțire. Modelul LB și SB memorează datele încărcate din memoria de date. Pe neașteptata CAB sunt trimise toate rezultatele provenite de la bufferul LB și de la unitățile de execuție. Algoritmii Tomosului poate fi reprezentat pe 3 etape.

1) Startare:

- Instrucțiunea este adusă din bufferul de prefetch într-o stație de rezervare. Dacă operandii se află în rețea de registre generali atunci vor fi aduși în stațiile de rezervare aferente. Dacă instrucțiunea este de tip Load/Store atunci va fi încărcată într-o stație de rezervare dacă există un buffer de rezervare, iar instrucțiunea va aștepta până când una dintre resurse se va elibera.

2) Execuție:

- Dacă un operand nu este disponibil, se testează existența hazardurilor de tip RAW între instrucțiuni, instrucțiunea se execută în unitatea de execuție corespunzătoare dacă ambii operanzi devin valizi.

Scrisoare rezultat:

Resultatul se va scrie pe CAB și de aici în FPR sau într-o stație de rezervare (FPR - și registre)

La această etapă nu există testare pentru hazarduri de tip WAR sau WAW. Operanșii vor fi prelucrați de către stația de rezervare direct de pe CDB prin "forwarding".

Stația de rezervare conține 6 câmpuri:

- OP - opcode, codul operației instrucțiunii
- Of, Ok - codifică pe un număr de biți, unitate de execuție (Add, Mul) sau numărul bufferului LB. Câmpurile Of, Ok sunt pe Post de TAG, adică când o unitate execuție sau un buffer - LB, precepe rezultatul pe CDB.
- Vj, Vk - conțin valorile operanșilor surse aferente instrucțiunii
- Busy. Indică atunci când este setat ca stația de rezervare și unitatea de execuție aferentă sunt ocupate momentan.

Exemplificarea funcționării algoritmului se poate face printr-o secvență simplă de program maximă

		Start	Execuție	WB
1. LF	F6, 27 (R1)	X	X	X
2. LF	F2, 45 (R2)	X	X	
3. MULT	F0, F2, F4	X		
4. SUB	F8, F6, F2			

5. DIV^{start} F10, F0, F6. X Ex oute XB

6. ADD^{start} F6, F8, F2 X

1. L^{start} F6, 27(R1) X X X

2. L^{start} F2, 45(R2) X X X

3. MULT^{start} F0, F2, F4 X X

4. SUB^{start} F8, F6, F2 X X X

5. DIV^{start} F10, F0, F6 X

6. ADD^{start} F6, F8, F2 X X X

Dupa toate operatiile starea statilor de rezervare si a setului de registre generali va fi

Numar SR	BUSy	OP	Vj'	Vic	Oj'	Ok.
ADD1	Nu.					
ADD2	Nu.					
ADD3	Nu.					
MUL1	Da.	/XIU LOAD		F4.		
MUL2	Da	Da		LOAD1		MUL1

CAMP	F0	F2	F4	F6	F8	F10
Q1	MUL1					
BUSY	DA	NU	NU	NU	NU	DA

Instrucțiunile au fost activate, prima a fost executată
 iar a doua în fază de execuție.
 Stările de rezervare

NUME SR.	BUSY	OP	Vj	Vk	Oj	Qk
Add1	DA	SUB	LOAD			LOAD2
Add2	DA	ADD			Add1	LOAD2
Add3	NU					
MUL1	DA	MUL		F(4)	LOAD2	
MUL2	DA	MUL		LOAD1	MUL1	

Registri Generali.

CÂMP	F0	F2	F4	F6	F8	F10
Q1	MUL1	LOAD		ADD2	ADD1	MUL1
BUSY	DA	DA	NU	DA	DA	DA

Să considerăm de exemplu. că latenta unităților ADD este de 2 implementări impulsuri de tact, latenta unităților MUL este 10 impulsuri de tact. pentru o înmulțire și respectiv 40 de impulsuri de tact. pentru o operație de împărțire "starea" secvenței anterioare în tactul premergător celui în care instrucțiunea MULTF va intra în faza WB. va fi următoarea.

Arhitectura Tamasulo are avantaje în detecția hazardurilor. printr-o logică distribuită și prin redenumirea dinamicii a resurselor. Un dezavantaj ar fi că necesită costuri ridicate, pentru că este o arhitectură complexă. Este necesară o logică de control complexă; să execute căutări/memorări rapide. Având în vedere progrese mari ale tehnologiilor VLSI, variante uscare, îmbunătățite ale acestei arhitecturi se aplică în toate procesoarele superclase actuale.