Scoreboarding
&
Tomasulo mechanizmus

Osnova

- Předcházení datovým hazardům dynamickým plánováním
- Scoreboarding
- Tomasulo algoritmus
- Závěr

Výhody dynamického plánování

- Dynamické plánování - hardware změní pořadí instrukcí se zachováním toku dat a chování při výjimkách
- Ošetřuje případy, kdy není možno závislosti určit v době překladu
  - to dovoluje procesoru tolerovat nepredikovatelná zpoždění, jako například výpadek cache tak, že provádí jiné instrukce, dokud cache výpadek nezpracuje
- Zároveň umožňuje efektivní běh programu, přeloženého pro jednu pipeline strukturu na jiné
- Zjednodušuje kompilátor
- Hardwarové spekulativní techniky – přinášejí podstatné výkonové zlepšení. Jsou založeny na dynamickém plánování

HW schémata: Instrukční paralelizmus

- Základní myšlenka: Provádět nezávislé instrukce za místem pozastavení

```
DIVD   F0,F2,F4
ADDD   F10,F0,F8
SUBD   F12,F8,F14
```

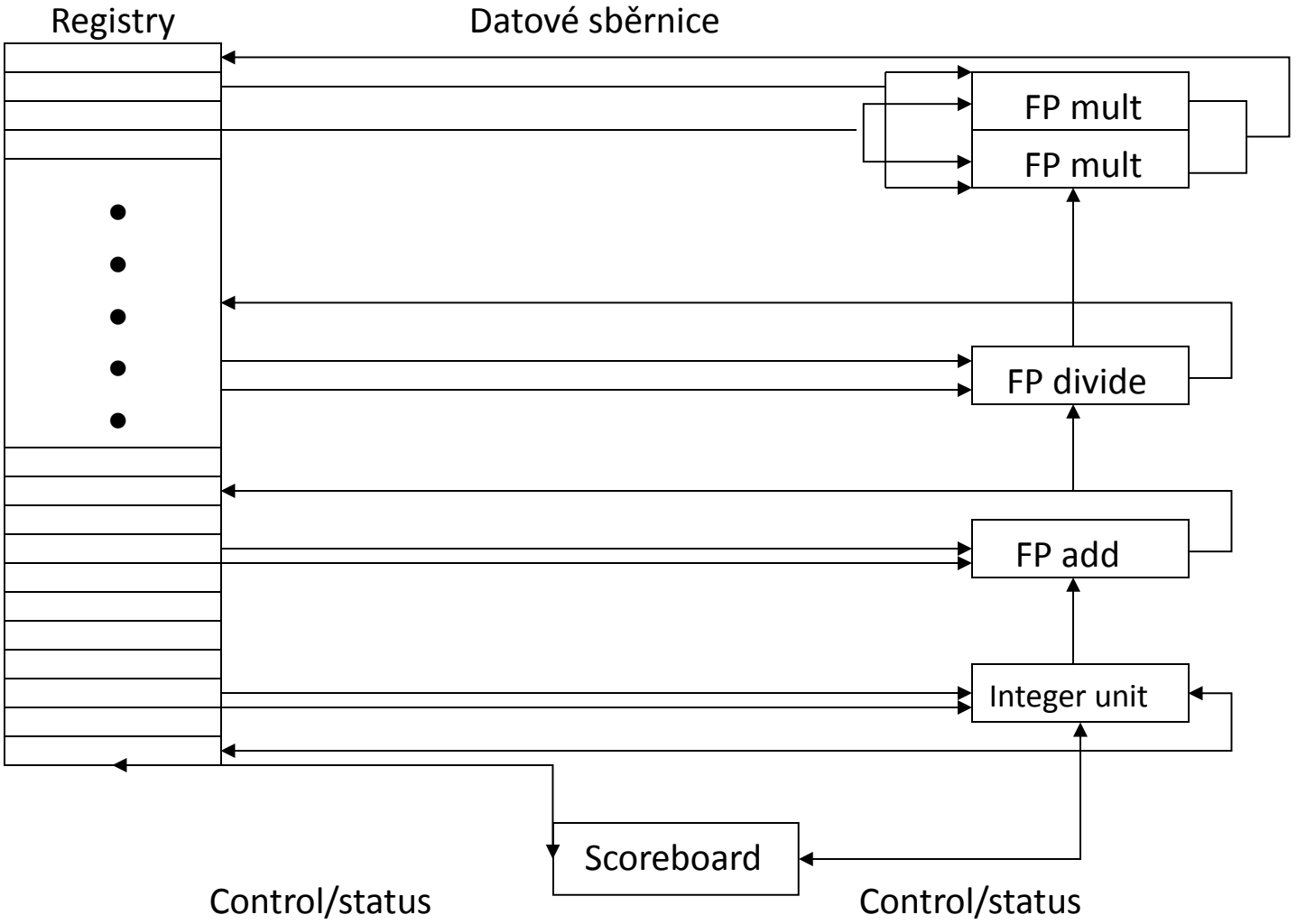
Pozastavení kvůli datovému hazardu

- Umožňuje „out-of-order“ provádění a dovoluje „out-of-order“ dokončení (zde např., **SUBD**)
  - V dynamicky plánované pipeline struktuře všechny instrukce přicházejí do vkládacího stupně (*issue stage*) v původním pořadí
- Rozlišujeme kdy se instrukce *začne vykonávat* a kdy se *dokončí*; mezi těmito dvěma časy se instrukce *provádí*

Poznámka: Dynamické provádění způsobuje hazardy WAR a WAW a ošetření výjimek je složitější

Scoreboarding

Scoreboard



Základní struktura DLX procesoru se scoreboardem

Fáze zpracování Instrukcí

(Instr.Execution cycle)

1. Načtení (IF = instr.fetch) – načtení instr.z paměti, zvýšení ukazatele
2. Dekódování (ID = instr.decode) – dekódování instrukce, načtení hodnot/parametrů z registrů
3. Provedení (EX = execute) – provedení operace nad daty
4. Přístup do paměti (MEM = memory access) – jen pro load/store
5. Zápis výsleku (WB = write back) – zápis výsledky do registru

Čtyři stupně řízení Scoreboardu

1. **Dekódování instrukcí & kontrola strukturních hazardů (ID1)**

   Je-li funkční jednotka pro zpracování instrukce volná a žádná jiná aktivní instrukce nemá stejný cílový registr (WAW) dodá Scoreboard instrukci do funkční jednotky a aktualizuje její interní datovou strukturu. Existuje-li strukturní nebo WAW hazard, potom je vložení instrukce pozastaveno a žádné další instrukce nejsou vkládány dokud nejsou hazardy odstraněny.

Algoritmus:

- Zajistí vkládání „In-Order“
- Umožňuje provedení většího počtu operací během jednoho cyklu
- Kontroluje, zda cílový registr je již rezervován pro zápis (WAW)
- Kontroluje, zda je volný vstup funkční jednotky – „Read-Operand stage“ (možný strukturní hazard)

Čtyři stupně řízení Scoreboardu

2. **Čtení operandů:** čekání dokud nezmizí datové hazardy, potom přečte operandy (ID2) – Prvý funkční stupeň pipeline.

   Zdrojový operand je k dispozici, jestliže žádná dříve vložená instrukce se jej nechystá zapsat, ani aktivní funkční jednotka právě nezapisuje do registru, který operand obsahuje. Když jsou zdrojové operandy k dispozici, Scoreboard přikáže funkční jednotce provést čtení operandů z registrů a spustit provádění. Scoreboard vyhodnotí dynamicky v tomto kroku RAW hazardy a instrukce mohou být poslány do prováděcí jednotky bez dodržení pořadí (out of order).

Algoritmus:

- ☐ Čekání na operandy dokud nejsou k dispozici, stav registru výsledku (RAW)
- ☐ Ukládání operandů do cache je povoleno
- ☐ Forwarding z jiného stupně WB (write-back fáze) je povolen

Čtyři stupně řízení Scoreboardu

3. **Execution:** operace s operandy (**EX**)
   - Funkční jednotka začne po obdržení operandů provádět operaci. Když je výsledek hotov, oznámí scoreboardu, že operaci dokončila. Tento stupeň lze dále podrobit (sub-)pipelinningu.
4. **Write result:** dokončení prováděcí fáze (**WB**)
   - Jakmile Scoreboard zjistí, že funkční jednotka dokončila operaci, Scoreboard testuje výskyt WAR hazardů.
     Neexistují-li, zapíše výsledky. Existují-li WAR, pozastaví instrukci.

Algoritmus:

- Opozdí zápis dokud všechna pole Rj a Rk tohoto registru nejsou označena jako „cached“ nebo „read“.
  - Jsou-li operandy v cache, jsou odpovědi (získané operandy) „forwardovány“ dál.
  - V opačném případě se zápisem čeká, dokud nejsou všechny operandy přečteny.
- Forwarduje odpovědi jednotkám čekajícím na tento zápis jejich operandu.

Tři části Scoreboardu

1. Stav instrukce
   - Indikuje, ve kterém ze čtyř kroků zpracování se instrukce nachází.
2. Stav funkční jednotky
   - Indikuje stav funkční jednotky (FU). 9 polí pro každou funkční jednotku
     - Busy – Indikuje, zda je jednotka v činnosti či nikoliv
     - Op – Operace, která se má v jednotce vykonat (např, + nebo -)
     - Fi – Cílový registr
     - Fj, Fk – Čísla zdrojových registrů
     - Qj, Qk – Funkční jednotky generující zdrojové registry Fj, Fk
     - Rj, Rk – Flagy indikující, že Fj, Fk jsou připraveny a dosud nebyly přečteny. (Alternativně: přečteny a uloženy do cache)
3. Stav registru výsledku:
   - Indikuje, která funkční jednotka bude zapisovat každý registr, pokud existuje. Zůstává prázdný, pokud žádná probíhající instrukce nebude provádět do tohoto registru zápis.

Scoreboard: příklad - cykl 1

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1			
LD	F2	45	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Clock 1

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	Yes	Load	F6		R2				Yes
Mult1	No								
Mult2	No								
Add	No								
Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Int					

R2 nebyl dosud přečten (a umístěn do cache) až do cyklu 2!!!

Scoreboard: příklad - cykl 2

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2		
LD	F2	45	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Clock 2

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	Yes	Load	F6		R2				Yes
Mult1	No								
Mult2	No								
Add	No								
Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Int					

Druhé LD je blokováno strukt.hazardem (jen jedna integer jednotka)

Scoreboard: příklad - cykl 4

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Clock 4

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	Yes	Load	F6		R2				No Yes
Mult1	No								
Mult2	No								
Add	No								
Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Int					

Scoreboard: příklad - cykl 5

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3	5			
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Clock 5

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	Yes	Load	F2		R3				Yes
Mult1	No								
Mult2	No								
Add	No								
Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU		Int							

SUPERSCALAR: Vložit MULTD?

Scoreboard: příklad - cykl 6

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3	5	6		
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Clock 6

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	Yes	Load	F2		R3				Yes
Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
Mult2	No								
Add	No								
Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Int							

MULTD nemůže pokračovat (RAW hazard F2 od druhého LD)

Scoreboard: příklad - cykl 7

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3	5	6	7	
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Clock 7

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	Yes	Load	F2		R3				No
Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
Mult2	No								
Add	Yes	Sub	F8	F6	F2		Int	Yes	No
Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Int			Add				

Číst více operandů? DIVD by mohla být vložena v tomto cyklu.

MUL I SUB čeká na F2

Scoreboard: příklad - cykl 8a

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3	5	6	7	
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Clock 8

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	Yes	Load	F2		R3				Yes
Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
Mult2	No								
Add	Yes	Sub	F8	F6	F2		Int	Yes	No
Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Int			Add	Div			

...Vložíme DIV....

Scoreboard: příklad - cykl 8b

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3	5	6	7	8
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Clock 8

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
Mult2	No								
Add	Yes	Sub	F8	F6	F2			Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1				Add	Div			

...a zároveň dokončujeme druhé LD (zápis F2)

Scoreboard: příklad - cykl 9

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9		
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Clock 9

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
Mult2	No								
Add	Yes	Sub	F8	F6	F2			Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1				Add	Div			

Vložit ADDD?

(nelze - stejná jednotka jako SUB. MUL by případně šlo, tam máme dvě jednotky)

Scoreboard: příklad - cykl 11

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Clock 11

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
Mult2	No								
Add	Yes	Sub	F8	F6	F2			Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1				Add	Div			

Sub je provedeno (2T), MUL se stále provádí (4T)

Scoreboard: příklad - cykl 12

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Clock 12

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
Mult2	No								
Add	No								
Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1					Div			

Dokončení SUB (uvolnění sčítací jednotky)

Scoreboard: příklad - cykl 13

Clock 13

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13			

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	No								
Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
Mult2	No								
Add	Yes	Add	F6	F8	F2			Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1			Add		Div			

Scoreboarding -souhrn

- Omezení scoreboardu u CDC 6600
  - Žádný forwarding hardware (=operandy jen v registrech)
  - Omezeno jen na instrukce základního bloku (malé okénko)
  - Malý počet funkčních jednotek (strukturní hazardy), zvláště jednotek integer/load/store
  - Vkládání se nekoná v případě strukturních nebo WAW hazardů
  - Čekání na zápis u WAR hazardů
  - Nepreciznı́ výjimky
- **Klíčová idea:** Necháme provést instrukce ležící za pozastavenou instrukcí
  - Dekódování $\Rightarrow$ vložení instrukcí a čtení operandů
  - Umožňuje provádění operací out-of-order $\Rightarrow$ out-of-order dokončení

Scoreboarding -souhrn

- Současná vylepšení:
  - Všechny operandy se umísťují do cache, jakmile jsou k dispozici
  - Forwarding
  - Pipelining funkčních jednotek
  - Mikrokódování, např. IA32 (rozšiřuje prováděcí okno)
  - Preciznější výjimky
  - Ukončování v pořadí
  - Pracuje nejlépe s velkým množstvím registrů
- Tomasulo metoda:
  - Rezervační stanice vs. Forwarding a Caching
  - „Dočasné“ registry jsou využívány jako velké množství virtuálních registrů

Metoda Tomasulo

Hardwarová schémata pro ILP

- Klíčová idea: Dovolit zpracování instrukcí za pozastavenou instrukcí
  - Dekódování => vložení instrukcí a čtení operandů
  - Dovoluje out-of-order provádění => out-of-order dokončení (mimo pořadí)
- Proč v hardware v době výpočtu? (stat. vs dyn. analýza)
  - Funguje když závislost není známa v době výpočtu
  - Zjednodušuje kompilátory
  - Dovoluje dobré zpracování kódu, určeného pro jiný stroj (optimalizovaného pro jiný stroj)
- Out-of-order provádění dělí ID stupeň:
  - Vkládání — dekódování instrukcí, kontrola vzniku strukturních hazardů
  - Čtení operandů — čekání dokud neodezní datové hazardy, pak čtení operandů

Tomasulo algoritmus

- Realizován u IBM 360/91 asi 3 roky po CDC 6600
- Cíl: Vysoký výkon bez použití speciálních kompilátorů
- Rozdíly mezi IBM 360 & CDC 6600 ISA
  - IBM má jen 2 registry specifikátor/instrukce oproti 3 u CDC 6600
  - IBM má 4 FP registry oproti 8 u CDC 6600
- Rozdíly mezi Tomasulo algoritmem & technikou Scoreboard
  - Řízení & buffery (nazýváno “rezervační stanice”) jsou distribuovány uvnitř funkčních jednotek oproti centralizovanému uspořádání u Scoreboardingu
  - Registry v instrukcích jsou nahrazeny pointery na buffer rezervační stanice
  - HW přejmenování registrů, aby se zabránilo WAR a WAW hazardům
  - Společná datová sběrnice (CDB) vysílá výsledky funkčním jednotkám (broadcast)
  - S operacemi Load a Store je zacházeno jako s funkčními jednotkami

Dynamický algoritmus: Tomasulo

- Navržen pro IBM 360/91 (ještě předtím, než se objevily cache!)
  - $\Rightarrow$ Dlouhá latence paměti
- Přínos: Vysoký výkon bez speciálních kompilátorů
- Malý počet FP registrů (4 u řady 360) nedovoloval progresivní plánování operací kompilátorem
  - To přivedlo návrháře IBM Tomasula k tomu, jak efektivněji využít registry - změna jména v hardware!
- Proč se zabývat počítačem z roku 1966 ?
- Všichni následníci byli úspěšní!
  - Alpha 21264, HP 8000, MIPS 10000, Pentium III, Pentium 4, PowerPC 604, AMD Opteron, Power 5 (IBM), …

Tři stupně Tomasulova algoritmu

1. Vkládání: Přesun instrukce z fronty FP operací

   Je-li rezervační stanice volná, vloží instrukci & vyšle operandy (přejmenuje registry).

2. Provedení: Operace prováděná s operandy (EX)

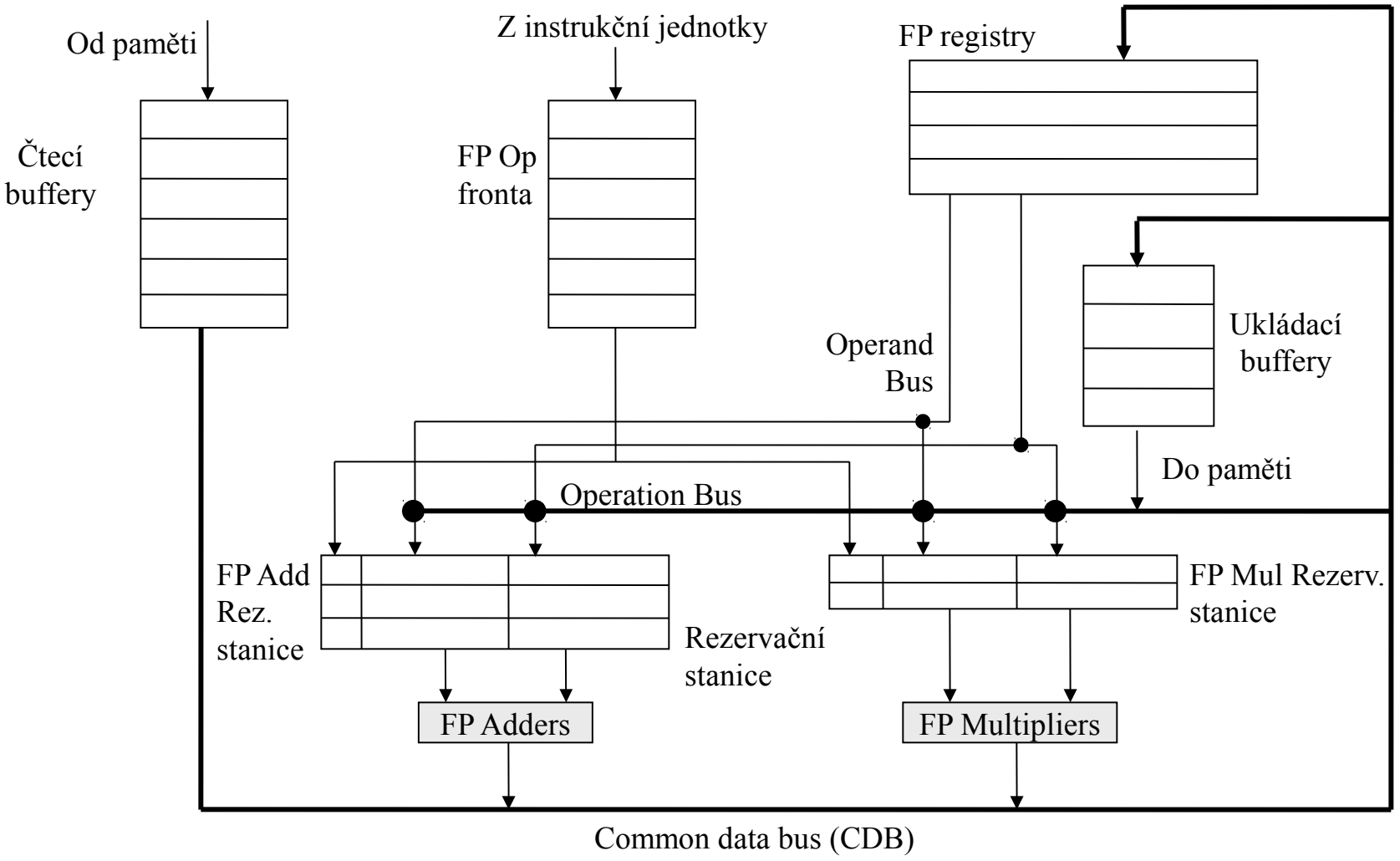
   Jsou-li operandy připraveny, pak provedení; v opačném případě je sledována společná sběrnice, zda nepřišel výsledek.

3. Zápis výsledku: Dokončení prováděcí fáze (WB)

   Zápis po společné sběrnici všem čekajícím jednotkám; označí rezervační stanici jako připravenou.

Common data bus: data + zdroj (přichází po sběrnici)

Tomasulo organizace



Komponenty rezervační stanice

- Op – Operace k provedení v jednotce (např. + nebo –)
- Qj, Qk – Rezervační stanice produkující zdrojové registry
- Vj, Vk – Hodnoty zdrojových operandů
- Rj, Rk – Flagy indikující zda Vj a Vk jsou „ready“
- Busy – Indikující, že rezervační stanice a FU je „busy“

- Stavový registr výsledku
  - Indikuje, která funkční jednotka bude zapisovat každý registr, pokud nějaký existuje. Prázdný, pokud žádné zpracovávané instrukce (pending) nebudou psát do tohoto registru.

Tomasulo: příklad - cykl 1

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34	R2	1		
LD	F2	45	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	No	
Load3	No	

Reservation Stations

Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
Add1	No					
Add2	No					
Add3	No					
Mult1	No					
Mult2	No					

Clock 1

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Load1					

Tomasulo: příklad - cykl 2

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34	R2	1		
LD	F2	45	R3	2		
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	Yes	45+R3
Load3	No	

Reservation Stations

Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
Add1	No					
Add2	No					
Add3	No					
Mult1	No					
Mult2	No					

Clock 2

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU		Load2		Load1					

Tomasulo: příklad - cykl 3

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34	R2	1	3	
LD	F2	45	R3	2		
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	Yes	45+R3
Load3	No	

Reservation Stations

Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
Add1	No					
Add2	No					
Add3	No					
Mult1	Yes	MULTD		R(F4)	Load2	
Mult2	No					

Clock 3

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Load2		Load1					

Jména registrů jsou „přejmenována“ v rezervačních stanicích

Dokončení Load1 — kdo čeká na Load1?

Tomasulo: příklad - cykl 4

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34	R2	1	3	4
LD	F2	45	R3	2	4	
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

Reservation Stations

Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
Add1	Yes	SUBD	M(34+R2)			Load2
Add2	No					
Add3	No					
Mult1	Yes	MULTD		R(F4)	Load2	
Mult2	No					

Clock 4

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Load2		M(34+R2)	Add1				

Dokončení Load2 — kdo na to čeká?

Tomasulo: příklad - cykl 5

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34	R2	1	3	4
LD	F2	45	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
Add1	Yes	SUBD	M(34+R2)	M(45+R3)		
Add2	No					
Add3	No					
Mult1	Yes	MULTD	M(45+R3)	R(F4)		
Mult2	Yes	DIVD		M(34+R2)	Mult1	

Clock 5

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	M(45+R3)			Add1	Mult2			

Tomasulo: příklad - cykl 6

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34	R2	1	3	4
LD	F2	45	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
Add1	Yes	SUBD	M(34+R2)	M(45+R3)		
Add2	Yes	ADDD		M(45+R3)	Add1	
Add3	No					
Mult1	Yes	MULTD	M(45+R3)	R(F4)		
Mult2	Yes	DIVD		M(34+R2)	Mult1	

Clock 6

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1			Add2	Add1	Mult2			

Tomasulo souhrn

- Rezervační stanice: přejmenovává do většího souboru registrů + dočasně pamatuje zdrojové operandy (buffering)
  - Registry přestávají být úzkým místem
  - Zabraňuje WAR a WAW hazardům, které se vyskytovaly u Scoreboardu
  - Dovoluje rozvíjení smyček (loop unrolling) v HW
- Neomezuje se jen na základní blok

  (integer jednotka pracuje dál až za skoky)
- Příspěvek
  - Dynamické plánování
  - Přejmenování registrů
  - Load/store zjednodušení
- 360/91 následníky jsou Pentium III; PowerPC 604; MIPS R10000; HP-PA 8000; Alpha 21264

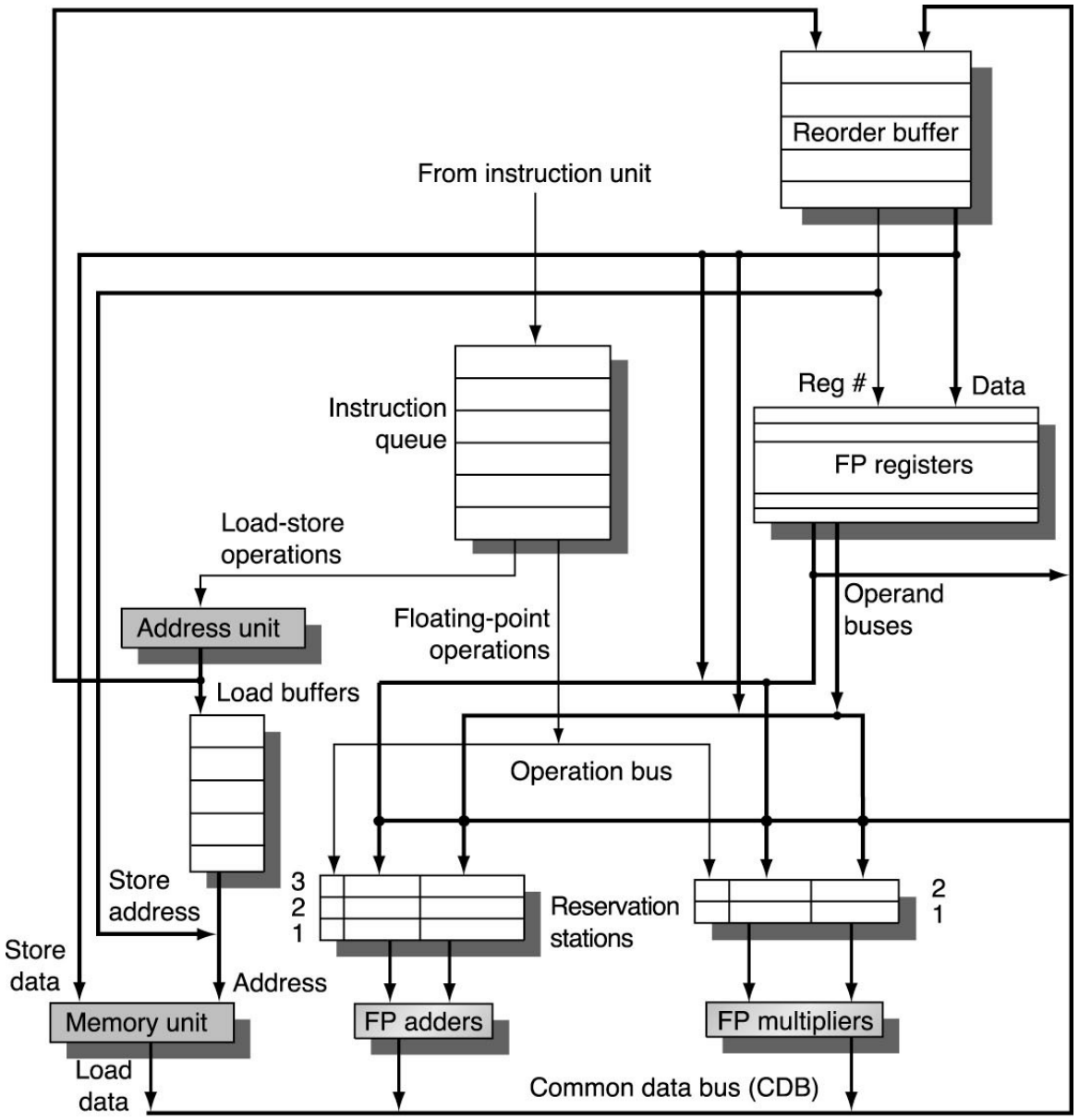
Tomasulo se spekulací

1. *Issue* – Prázdná rezervační stanice a prázdný ROB slot. Operandy jsou zaslány do rezervační stanice z registrového souboru a nebo z ROB (re-order buffer). Tento stupeň se označuje: *dispatch*
2. *Execute* – Monitoruje operandy v CDB, hlídá RAW hazardy. Jsou-li oba operandy k dispozici, provede operaci.
3. *Write Result* – Je-li hotový, je výsledek zapsán do CDB prostřednictvím ROB a do každé další čekající rezervační stanice. Stores write to value field in ROB.
4. *Commit* – tři případy:
   - Normální *Commit*: zápis registrů, „in order“ *Commit*
   - Store: aktualizace paměťi
   - Nekorektní větvení: zahodí ROB, obsah rezervační stanice a restartuje výpočet na korektní hodnotě PC

ROB … ReOrder Buffer

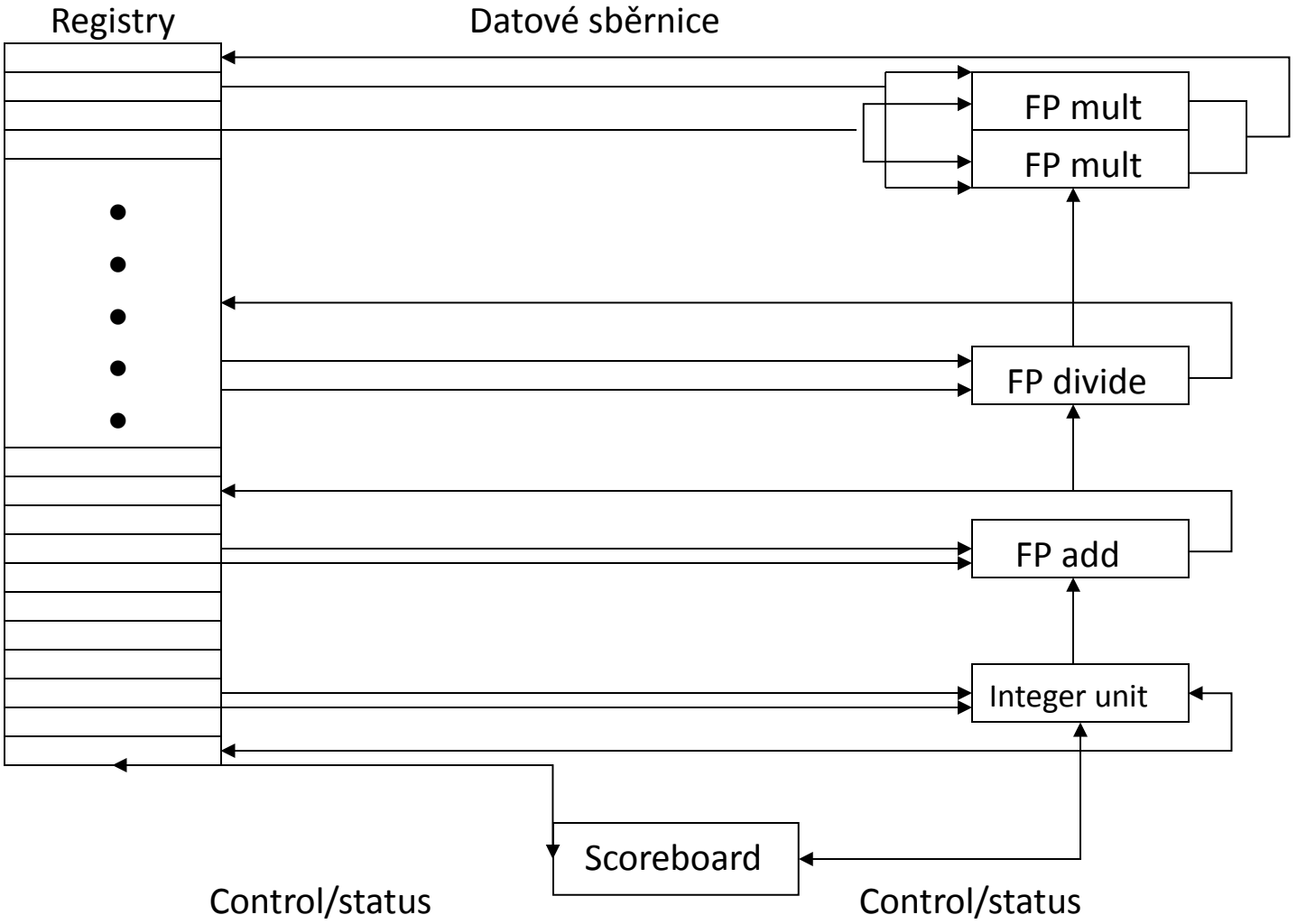
Nevýhody Tomasulova mechanizmu

- Složitost
  - opoždění 360/91, MIPS 10000, Alpha 21264, IBM PPC 620 !
- Mnoho rychlých asociativních pamětí (CDB)
- Výkon omezený sběrnicí „Common Data Bus“
  - Každý CDB musí procházet k většímu počtu funkčních jednotek ⇒velké kapacity, velká hustota propojení
  - Počet funkčních jednotek, které mohou dokončit v jednom cyklu je omezen na jednu!
    - Násobné CDB ⇒ více FU logiky pro paralelní asociativní paměti
- Neprecizní interupty/vyjímky!



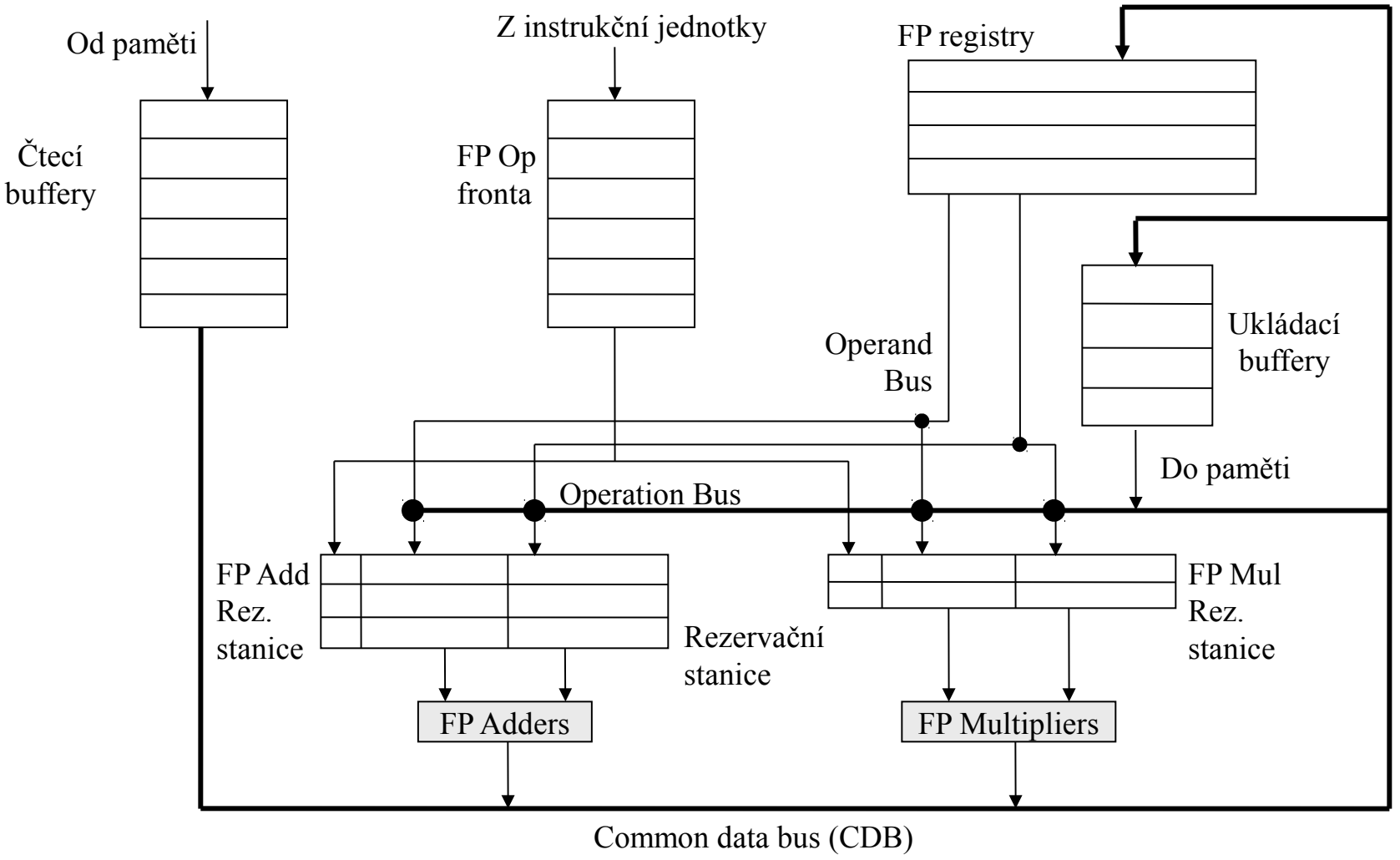
Porovnání

Scoreboard



Základní struktura DLX procesoru se scoreboardem

Tomasulo organizace



Scoreboard: příklad - cykl 6

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34	R2	1	2	3	4
LD	F2	45	R3	5	6		
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Clock 6

Functional unit status

Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
Integer	Yes	Load	F2		R3				Yes
Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
Mult2	No								
Add	No								
Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Int							

Tomasulo: příklad – cykl 6

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34	R2	1	3	4
LD	F2	45	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
Add1	Yes	SUBD	M(34+R2)	M(45+R3)		
Add2	Yes	ADDD		M(45+R3)	Add1	
Add3	No					
Mult1	Yes	MULTD	M(45+R3)	R(F4)		
Mult2	Yes	DIVD		M(34+R2)	Mult1	

Clock 6

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1			Add2	Add1	Mult2			

Rozdíly mezi Tomasulovým algoritmem & Scoreboardingem

- Řízení & buffery (“rezervační stanice”) distribuovány mezi funkční jednotky
- Registry v instrukcích nahrazeny pointery na buffery rezervační stanice
- HW přejmenování registrů kvůli zamezení WAR a WAW hazardům
- Common data bus (CDB) vysílá výsledky funkčním jednotkám
- S Load a Store je zacházeno jako s funkčními jednotkami
- Stupně: Issue, Execution, Write Result

- Řízení & buffery jsou centralizované
- Používá aktuální registry
- Nevkládá, pokud nastává strukturní nebo WAW hazard
- Čekání na WAR hazardy
- Forwarding?
- Stupně: Issue, Read operands, Execution, Write Result

Doing nothing is very hard to do, you never know when you're finished.

----- STOP --- rozpracováno ---

Dynamické plánování - krok 1

- Jednoduchá pipeline struktura obsahuje pouze 1 stupeň pro kontrolu strukturních a datových hazardů: Dekódování instrukce (ID)

Uspořádání:

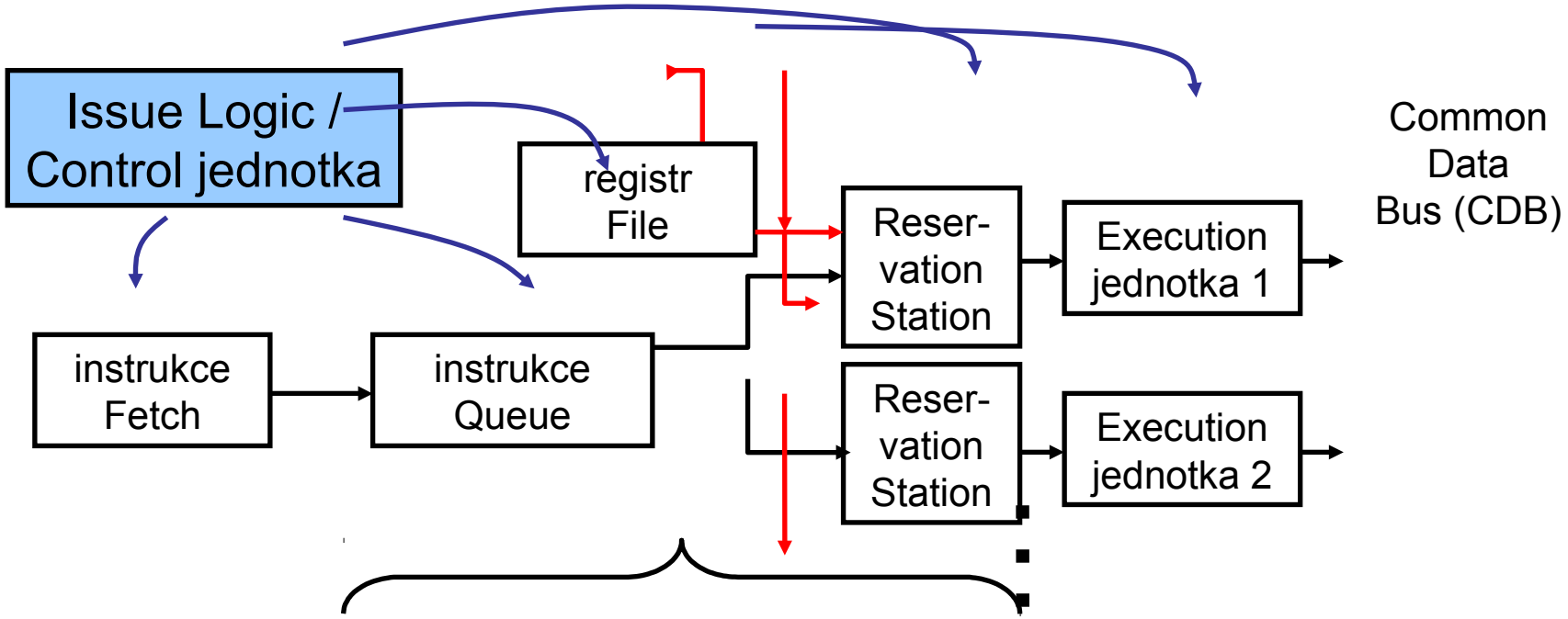
- Rozdělíme ID stupeň jednoduché 5-stupňové pipeline do 2 stupňů:
- *Vkládání (Issue)*- dekódování instrukcí, kontrola strukturních hazardů
- *Čtení operandů* - čekání dokud neodezní hazardy, pak se provede načtení operandů

Dynamický algoritmus: Tomasulo

- Navržen pro IBM 360/91 (ještě předtím, než se objevily cache!)
  - $\Rightarrow$ Dlouhá latence paměti
- Přínos: Vysoký výkon bez speciálních kompilátorů
- Malý počet FP registrů (4 u řady 360) nedovoloval progresivní plánování operací kompilátorem
  - To přivedlo návrháře IBM Tomasula k tomu, jak efektivněji využít registry - změna jména v hardware!
- Proč se zabývat počítačem z roku 1966 ?
- Všichni následníci byli úspěšní!
  - Alpha 21264, Pentium 4, AMD Opteron, Power 5, …

Dynamický algoritmus: Tomasulo

- Klíčové rozdíly (oproti technice Scoreboarding) :
  - Detekce hazardů & vkládání instrukcí se děje pro každou prováděcí jednotku
  - Výsledky jdou přímo tam, kde jsou zapotřebí, použití CDB (Common Data Bus)
  - Operace Load/Store mají vlastní prováděcí jednotky
  - Použití *Rezervační Stanice* pro přejmenování registrů



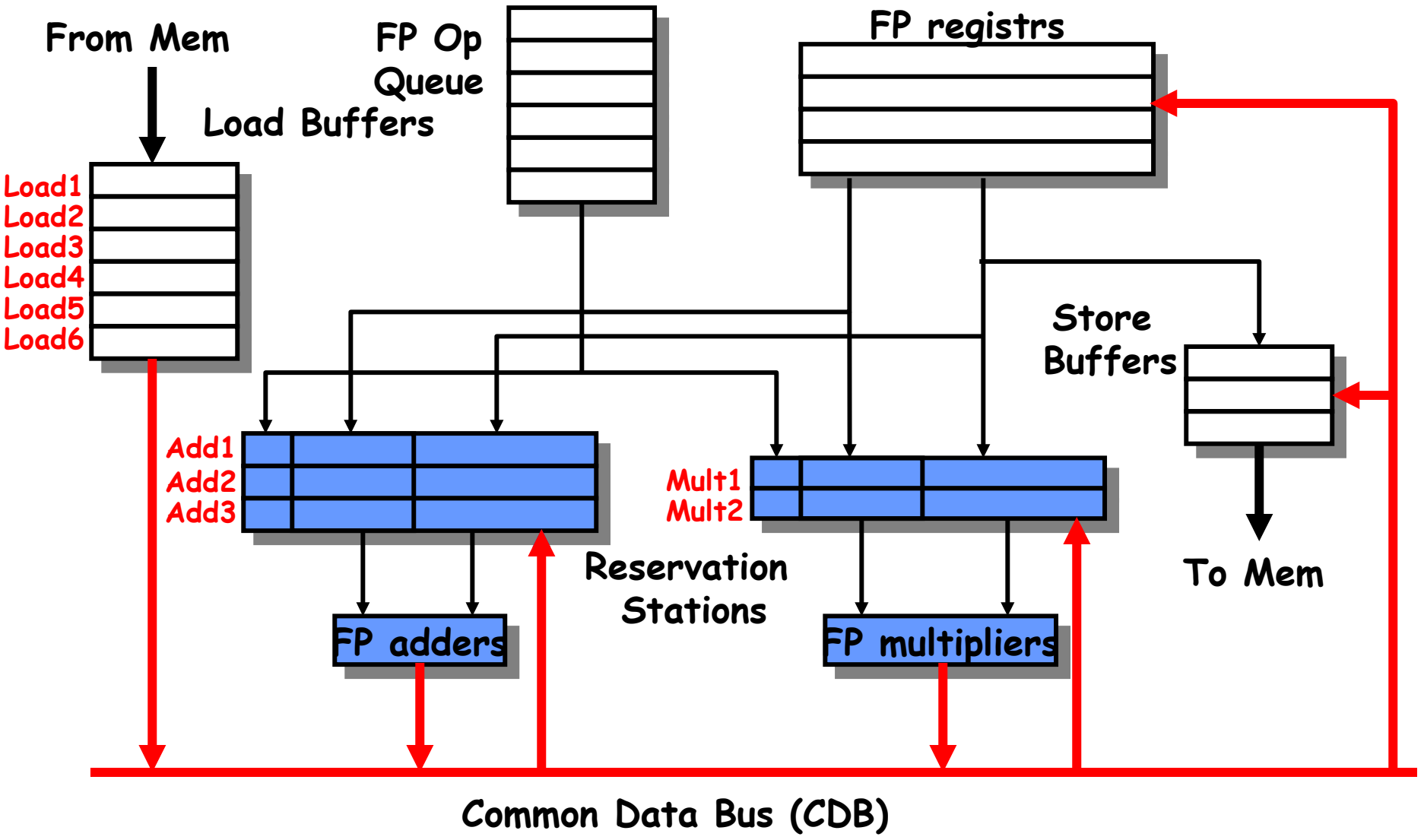
Komponenty Tomasulovy jednotky

- Rezervační stanice (RS)
  - Ukládá operandy probíhajících instrukcí, když čekají na operandy, aby mohly vstoupit do prováděcích jednotek.
- Logika vkládání (Issue logic)
  - Přesměrovává (přejmenovává) výstupní registry instrukcí k uložení do pozic v rezervační stanici.
  - Výsledky jdou přímo do RS, neprocházejí registrovou sadou.
- Distribuovaná detekce hazardů
  - Ošetřováno separátně každou funkční jednotkou
- Load & store buffery (mohou být kombinovány s RS)
  - Vytvářejí frontu požadavků na přístup k paměti

Algoritmus Tomasulo

- Řízení & buffery jsou distribuovány ve funkční jednotce (FU)
  - FU buffery nazývané "rezervační stanice"; obsahují připravené operandy
- Registry v instrukci jsou nahrazeny hodnotami nebo pointery na rezervační stanice (RS); proces se nazývá přejmenování registrů ;
  - Přejmenování zamezuje hazardy typu WAR, WAW
  - Více rezervačních stanic než registrů, lze provádět optimalizaci, které překladače nejsou schopny
- Výsledky do FU z RS neprochází registry, ale sběrnicí Common Data Bus se vysílají do všech FU (broadcast)
  - Zabraňuje se hazardům RAW, protože instrukce se provádí jen když má k dispozici své operandy
- S Load a Store je zacházeno stejně jako s FU s RS
- Integer instrukce mohou přesahovat větvení (predikován skok) a mohou uvolnit FP operace v FP frontě za základním blokem

Tomasulova organizace



Komponenty rezervační stanice

Op: Operace, kterou má jednotka vykonat (např., + nebo –)

Vj, Vk: Hodnota zdrojových operandů

- Buffery pro uložení výsledků mají pole V, výsledek se má uložit

Qj, Qk: Rezervační stanice produkující zdrojové registry (hodnoty, které se mají zapsat)

- Poznámka: Qj,Qk=0 => ready
- Pouze buffery pro uložení výsledků mají Qi pro RS produkující výsledek

Busy: Indikuje, že rezervační stanice nebo FU je v činnosti

Stavový registr výsledku (Register result status) - Indikuje, která funkční jednotka bude zapisovat do registru, pokud existuje. Prázdný, jestliže není žádná probíhající instrukce, která bude do tohoto registru zapisovat.

Tři stupně algoritmu Tomasulo

1. Issue (vkládání) – obdrží instrukci z fronty FP operací

   Je-li rezervační stanice volná (žádný strukturní hazard),
   vloží řadič instrukci & pošle operandy (přejmenuje registry).
2. Execute (provádění) – zpracování operandů (EX)

   Jsou-li oba operandy připraveny, následuje provedení operace;
   v opačném případě se čeká na výsledek ze sběrnice „Common Data Bus“
3. Write result (zápis výsledku) – dokončení provádění instrukce (WB)

   Zápis pomocí sběrnice „Common Data Bus“ do všech čekajících jednotek;
   rezervační stanice se označí jako připravená

- Normální datová sběrnice: data + cíl (“go to” bus)
- Common data bus: data + zdroj (“come from” bus)
  - data 64 bitů + 4 bity zdrojové adresy FU (Functional Unit)
  - Zápis se koná, souhlasí-li předpokládaná FU (produkující výsledek)
  - Provede vyslání (broadcast)
- Příklady rychlosti:
  2 takty hodin pro FP **+**,-; 10 pro * ; 40 taktů hodin pro */*

Příklad (Tomasulo alg.)

instrukce stream

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2			
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

3 Load/Buffers

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

FU count down

3 FP Adder R.S.
2 FP Mult R.S.

Register result status:

Clock	*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
0	*FU*								

Clock cycle counter

Příklad (Tomasulo alg.) - cykl 1

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1		
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
1	*FU*				Load1					

Příklad (Tomasulo alg.) - cykl 2

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1		
LD	F2	45+	R3	2		
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
2	*FU*		Load2		Load1					

Poznámka: Může probíhat větší počet nedokončených operací „Load“

Příklad (Tomasulo alg.) - cykl 3

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	
LD	F2	45+	R3	2		
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		R(F4)	Load2	
	Mult2	No					

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
3	*FU*	Mult1	Load2		Load1					

- **Poznámka: jména registrů v RS jsou odstraněna (“renamed”); vložena MULT**
- **Load1 se dokončuje; kdo čeká na Load1?**

Příklad (Tomasulo alg.) - cykl 4

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	Yes	SUBD	M(A1)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		R(F4)	Load2	
	Mult2	No					

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
4	*FU*	Mult1	Load2		M(A1)	Add1				

- **Load2 se dokončuje; co čeká na Load2?**

Příklad (Tomasulo alg.) - cykl 5

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
2	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
10	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
5	*FU*	Mult1	M(A2)		M(A1)	Add1	Mult2			

- **Timer startuje odpočítávání pro Add1, Mult1**

Příklad (Tomasulo alg.) - cykl 6

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
1	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
9	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
6	*FU*	Mult1	M(A2)		Add2	Add1	Mult2			

- **Vložení ADDD i přes závislost jména F6?**

Příklad (Tomasulo alg.) - cykl 7

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
0	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
7	*FU*	Mult1	M(A2)		Add2	Add1	Mult2			

- **Add1 (SUBD) dokončuje; na co čeká?**

Příklad (Tomasulo alg.) - cykl 8

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
2	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
7	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
8	*FU*	Mult1	M(A2)		Add2	(M-M)	Mult2			

Příklad (Tomasulo alg.) - cykl 9

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
1	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
6	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
9	*FU*	Mult1	M(A2)		Add2	(M-M)	Mult2			

Příklad (Tomasulo alg.) - cykl 10

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
0	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
10	*FU*	Mult1	M(A2)		Add2	(M-M)	Mult2			

- **Add2 (ADDD) dokončuje; co na ní čeká?**

Příklad (Tomasulo alg.) - cykl 11

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
11	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

- **Lze nyní zapsat výsledek od ADDD ?**
- **Všechny rychlé instrukce se dokončují v tomto cyklu!**

Příklad (Tomasulo alg.) - cykl 12

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
12	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Příklad (Tomasulo alg.) - cykl 13

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
13	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Příklad (Tomasulo alg.) - cykl 14

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
14	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Příklad (Tomasulo alg.) - cykl 15

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
15	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

- **Mult1 (MULTD) se dokončuje; co na ní čeká?**

Příklad (Tomasulo alg.) - cykl 16

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
16	*FU*	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

- **Čeká se pouze na dokončení Mult2 (DIVD)**

Pro urychlení některé cykly přeskočíme

Příklad (Tomasulo alg.) - cykl 55

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
1	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
55	*FU*	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

Příklad (Tomasulo alg.) - cykl 56

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5	56	
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
56	*FU*	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

- **Mult2 (DIVD) se dokončuje; co na ni čeká?**

Příklad (Tomasulo alg.) - cykl 57

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5	56	57
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
56	*FU*	M*F4	M(A2)		(M-M+M	(M-M)	Result			

- **Ještě jednou: In-order vkládání, out-of-order provádění a out-of-order dokončení.**

Proč může algoritmus (Tomasulo) překrývat iterace smyček?

- Přejmenování registrů
  - Následné iterace mají rozdílné fyzické cílové registry (dynamické rozbalení smyček).
- Rezervační stanice
  - Dovolují vkládání instrukcí pro urychlení operací za Permit instrukce issue to advance past integer control flow operations
  - Také ukládají starší hodnoty registrů – úplně se vylučuje zastavení kvůli hazardu typu WAR
- Jiná perspektiva: Tomasulo vytváří graf závislosti toku dat on-line (on the fly)

Příklad smyčky - Tomasulo

```
Loop: LD     F0 0  R1
      MULTD  F4 F0 F2
      SD     F4 0  R1
      SUBI      R1 R1 #8
      BNEZ      R1 Loop
```

Load=8,4 Add=3 Mult=4 SUBI/BNEZ=1 Store=3

- Předpokládejme, že násobení trvá 4 takty
- Předpokládejme, že prvé načtení trvá 8 taktů (cache miss?), druhé načtení trvá 4 takty (hit)
- Pro jednoduchost, vypsány takty pro SUBI, BNEZ
- Realita, integer instrukce napřed

Příklad smyčky - cykl 0

Instruction status

Instruction		j	k	iteration	Issue	Execution complete	Write Result
LD	F0	0	R1	1			
MULT	F4	F0	F2	1			
SD	F4	0	R1	1			
LD	F0	0	R1	2			
MULT	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
Load1	No		
Load2	No		
Load3	No		
Store1	No		
Store2	No		
Store3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	No					
0	Mult2	No					

Code:

```
LD    F0   0  R1
MULT  F4   F0 F2
SD    F4   0  R1
SUBI  R1   R1 #8
BNEZ  R1   Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12 ...	F30
0	80	Qi								

Příklad smyčky - cykl 1

Instruction status

Instruction		j	k	iteration	Issue	Execution complete	Write Result
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1			
SD	F4	0	R1	1			
LD	F0	0	R1	2			
MULT	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	No		
Store2	No		
Store3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	No					
0	Mult2	No					

Code:

```
LD    F0   0  R1
MULT  F4   F0 F2
SD    F4   0  R1
SUBI  R1   R1 #8
BNEZ  R1   Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12 ...	F30
1	80	Qi	Load1							

Příklad smyčky - cykl 2

Instruction status

Instruction		j	k	iteration	Issue	Execution complete	Write Result
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1			
LD	F0	0	R1	2			
MULT	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	No		
Store2	No		
Store3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	Yes	MULTD		R(F2)	Load1	
0	Mult2	No					

Code:

```
LD    F0  0  R1
MULT  F4  F0 F2
SD    F4  0  R1
SUBI  R1  R1 #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12 ...	F30
2	80	Qi	Load1		Mult1					

Příklad smyčky - cykl 3

Instruction status

Instruction		j	k	iteration	Issue	Execution complete	Write Result
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2			
MULT	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	Yes	MULTD		R(F2)	Load1	
0	Mult2	No					

Code:

```
LD    F0   0  R1
MULT  F4   F0 F2
SD    F4   0  R1
SUBI  R1   R1 #8
BNEZ  R1   Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12 ...	F30
3	80	Qi	Load1		Mult1					

- **Poznámka: MULT1 nemá žádná jména registrů v RS**

Příklad smyčky - cykl 4

Instruction status

Instruction		j	k	iteration	Issue	Execution complete	Write Result
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2			
MULT	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	Yes	MULTD		R(F2)	Load1	
0	Mult2	No					

Code:

```
LD    F0  0  R1
MULT  F4  F0 F2
SD    F4  0  R1
SUBI  R1  R1 #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12 ...	F30
4	72	Qi	Load1		Mult1					

Příklad smyčky - cykl 5

Instruction status

Instruction		j	k	iteration	Issue	Execution complete	Write Result
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2			
MULT	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	Yes	MULTD		R(F2)	Load1	
0	Mult2	No					

Code:

```
LD    F0   0  R1
MULT  F4   F0 F2
SD    F4   0  R1
SUBI  R1   R1 #8
BNEZ  R1   Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12 ...	F30
5	72	Qi	Load1		Mult1					

Příklad smyčky - cykl 6

Instruction status												
Instruction		*j*	*k*	*iteration*	*Issue*	*Execution complete*	*Write Result*		Busy	Address		
LD	F0	0	R1	1	1			Load1	Yes	80		
MULT	F4	F0	F2	1	2			Load2	Yes	72		
SD	F4	0	R1	1	3			Load3	No		Qi	
LD	F0	0	R1	2	6			Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2				Store2	No			
SD	F4	0	R1	2				Store3	No			

Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load1		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	

Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
6		**72**	*Qi*	Load2		Mult1						

- **Poznámka: F0 nikdy nevidí výsledek operace Load1**

Příklad smyčky - cykl 7

Instruction status												
Instruction		*j*	*k*	*iteration*	*Issue*	*Execution complete*	*Write Result*		Busy	Address		
LD	F0	0	R1	1	1			Load1	Yes	80		
MULT	F4	F0	F2	1	2			Load2	Yes	72		
SD	F4	0	R1	1	3			Load3	No		Qi	
LD	F0	0	R1	2	6			Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2	7			Store2	No			
SD	F4	0	R1	2				Store3	No			

Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load1		SUBI	R1	R1	#8
	0	Mult2	Yes	MULTD		R(F2)	Load2		BNEZ	R1	Loop	

Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
7		**72**	*Qi*	Load2		Mult2						

- **Poznámka: MULT2 nemá žádná jména registrů v RS**

Příklad smyčky - cykl 8

Instruction status

Instruction		j	k	iteration	Issue	Execution complete	Write Result
LD	F0	0	R1	1	1		
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2	6		
MULT	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
Load1	Yes	80	
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	Yes	MULTD		R(F2)	Load1	
0	Mult2	Yes	MULTD		R(F2)	Load2	

Code:

```
LD    F0   0  R1
MULT  F4   F0 F2
SD    F4   0  R1
SUBI  R1   R1 #8
BNEZ  R1   Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12 ...	F30
8	72	Qi	Load2		Mult2					

Příklad smyčky - cykl 9

Instruction status

Instruction		j	k	iteration	Issue	Execution complete	Write Result
LD	F0	0	R1	1	1	9	
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2	6		
MULT	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
Load1	Yes	80	
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
0	Add3	No					
0	Mult1	Yes	MULTD		R(F2)	Load1	
0	Mult2	Yes	MULTD		R(F2)	Load2	

Code:

```
LD   F0   0  R1
MULT F4   F0 F2
SD   F4   0  R1
SUBI R1   R1 #8
BNEZ R1   Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12 ...	F30
9	64	Qi	Load2		Mult2					

- **Load1 dokončuje; co na ni čeká?**

Příklad smyčky - cykl 10

Instruction status

Instruction		j	k	iteration	Issue	Execution complete	Write Result
LD	F0	0	R1	1	1	9	10
MULT	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2	6	10	
MULT	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
Load1	No		
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
0	Add3	No					
4	Mult1	Yes	MULTD	M(80)	R(F2)		
0	Mult2	Yes	MULTD		R(F2)	Load2	

Code:

```
LD    F0   0  R1
MULT  F4   F0 F2
SD    F4   0  R1
SUBI  R1   R1 #8
BNEZ  R1   Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12 ...	F30
10	64	Qi	Load2		Mult2					

- **Load2 dokončuje; co na ni čeká?**

Příklad smyčky - cykl 11

Instruction status						Execution	Write					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2			Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2	7			Store2	Yes	72	Mult2	
SD	F4	0	R1	2	8			Store3	No			

Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	3	Mult1	Yes	MULTD	M(80)	R(F2)			SUBI	R1	R1	#8
	4	Mult2	Yes	MULTD	M(72)	R(F2)			BNEZ	R1	Loop	

Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
11		**64**	*Qi*	Load3		Mult2						

Příklad smyčky - cykl 12

Instruction status												
Instruction		*j*	*k*	*iteration*	*Issue*	*Execution complete*	*Write Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2			Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2	7			Store2	Yes	72	Mult2	
SD	F4	0	R1	2	8			Store3	No			

Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	2	Mult1	Yes	MULTD	M(80)	R(F2)			SUBI	R1	R1	#8
	3	Mult2	Yes	MULTD	M(72)	R(F2)			BNEZ	R1	Loop	

Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
12		**64**	*Qi*	Load3		Mult2						

Příklad smyčky - cykl 13

Instruction status												
Instruction		*j*	*k*	*iteration*	*Issue*	*Execution complete*	*Write Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2			Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2	7			Store2	Yes	72	Mult2	
SD	F4	0	R1	2	8			Store3	No			

Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	1	Mult1	Yes	MULTD	M(80)	R(F2)			SUBI	R1	R1	#8
	2	Mult2	Yes	MULTD	M(72)	R(F2)			BNEZ	R1	Loop	

Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
13		**64**	*Qi*	Load3		Mult2						

Příklad smyčky - cykl 14

Instruction status						Execution	Write					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14		Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	Mult1	
MULT	F4	F0	F2	2	7			Store2	Yes	72	Mult2	
SD	F4	0	R1	2	8			Store3	No			

Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD	M(80)	R(F2)			SUBI	R1	R1	#8
	1	Mult2	Yes	MULTD	M(72)	R(F2)			BNEZ	R1	Loop	

Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
14		**64**	*Qi*	Load3		Mult2						

- **Mult1 dokončuje; co na ni čeká?**

Příklad smyčky - cykl 15

Instruction status						Execution	Write					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	M(80)*R(F	
MULT	F4	F0	F2	2	7	15		Store2	Yes	72	Mult2	
SD	F4	0	R1	2	8			Store3	No			

Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	No						SUBI	R1	R1	#8
	0	Mult2	Yes	MULTD	M(72)	R(F2)			BNEZ	R1	Loop	

Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
15		**64**	*Qi*	Load3		Mult2						

- **Mult2 dokončuje; co na ni čeká?**

Příklad smyčky - cykl 16

Instruction status						Execution	Write					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	M(80)*R(F	
MULT	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(7	
SD	F4	0	R1	2	8			Store3	No			

Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	

Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
16		**64**	*Qi*	Load3		Mult1						

Příklad smyčky - cykl 17

Instruction status						Execution	Write					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3			Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	M(80)*R(F	
MULT	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(7	
SD	F4	0	R1	2	8			Store3	Yes	64	Mult1	
Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	
Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
17		**64**	*Qi*	Load3		Mult1						

Příklad smyčky - cykl 18

Instruction status						Execution	Write					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULT	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3	18		Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	Yes	80	M(80)*R(F	
MULT	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(7	
SD	F4	0	R1	2	8			Store3	Yes	64	Mult1	

Reservation Stations					*S1*	*S2*	*RS for j*	*RS for k*				
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULT	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	

Register result status												
Clock		**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
18		**56**	*Qi*	Load3		Mult1						

Příklad smyčky - cykl 19

instrukce status						Execution	zápis					
instrukce		*j*	*k*	*iteration*	*Issue*	*complete*	*výsledek*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTD	F4	F0	F2	1	2	14	15	Load2	No			
SD	F4	0	R1	1	3	18	19	Load3	Yes	64	Qi	
LD	F0	0	R1	2	6	10	11	Store1	No			
MULTD	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(72)	
SD	F4	0	R1	2	8	19		Store3	Yes	64	Mult1	

Reservation Stations					*S1*	*S2*						
	Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULTD	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	

registr výsledek status											
Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
19	**56**	*Qi*	Load3		Mult1						

Příklad smyčky - cykl 20

instrukce status						Execution	zápis				
instrukce		j	k	iteration	Issue	complete	výsledek		Busy	Address	
LD	F0	0	R1	1	1	9	10	Load1	No		
MULTD	F4	F0	F2	1	2	14	15	Load2	No		
SD	F4	0	R1	1	3	18	19	Load3	Yes	64	Qi
LD	F0	0	R1	2	6	10	11	Store1	No		
MULTD	F4	F0	F2	2	7	15	16	Store2	No		
SD	F4	0	R1	2	8	19	20	Store3	Yes	64	Mult1

Reservation Stations					S1	S2						
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
	0	Add1	No						LD	F0	0	R1
	0	Add2	No						MULTD	F4	F0	F2
	0	Add3	No						SD	F4	0	R1
	0	Mult1	Yes	MULTD		R(F2)	Load3		SUBI	R1	R1	#8
	0	Mult2	No						BNEZ	R1	Loop	

registr výsledek status											
Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
21	56	Qi	Load3		Mult1						

Tomasulo schéma přináší dvě hlavní výhody

1. **Distribuovanou logiku detekce hazardu**
   - distribuované rezervační stanice a CDB (Common Data Bus)
   - Čeká-li více instrukcí na jediný výsledek & každá instrukce má další operand, mohou být instrukce uvolněny současně operací broadcast po CDB
   - Jestliže byla použita centrální registrová sada, jednotky budou muset číst výsledky z registrů, jakmile jsou sběrnice registrů volné
2. **Vyloučení bublin způsobených hazardy WAW a WAR**

Nevýhody Tomasulova mechanizmu

- Složitost
  - opoždění 360/91, MIPS 10000, Alpha 21264, IBM PPC 620 !
- Mnoho rychlých asociativních pamětí (CDB)
- Výkon omezený sběrnicí „Common Data Bus“
  - Každý CDB musí procházet k většímu počtu funkčních jednotek ⇒velké kapacity, velká hustota propojení
  - Počet funkčních jednotek, které mohou dokončit v jednom cyklu je omezen na jednu!
    - Násobné CDB ⇒ více FU logiky pro paralelní asociativní paměti
- Neprecizní interupty!
  - Zmíníme se o nich později

Závěr #1

- Vliv implicitního paralelismu na výkon: instrukce Level paralelism
- Rozbalení smyček kompilátorem pro zvýšení ILP
- Predikce větvení pro zvýšení ILP
- Dynamický HW využívající ILP
  - Funkční i když nejsou známy závislosti v době překladu
  - Může maskovat výpadky L1 cache
  - Kód napsaný pro jeden stroj může dobře běžet i na jiném

Závěr #2

- Rezervační stanice: *renaming* na větší soubor registrů + ukládání zdrojových operandů
  - Potlačuje nepříznivý vliv malého počtu registrů
  - Vylučuje hazardy WAR, WAW
  - Umožňuje rozbalování smyček v HW
- Není omezeno jen na základní bloky
  (integer jednotky vykonávají instrukce dopředu, i za větvení)
- Pomáhá při výpadcích cache
- Trvalý přínos
  - Dynamické plánování
  - Přejmenování registrů
  - Potlačení nepříznivého vlivu Load/Store operací
- 360/91 následníky jsou Intel Pentium 4, IBM Power 5, AMD Athlon/Opteron, …