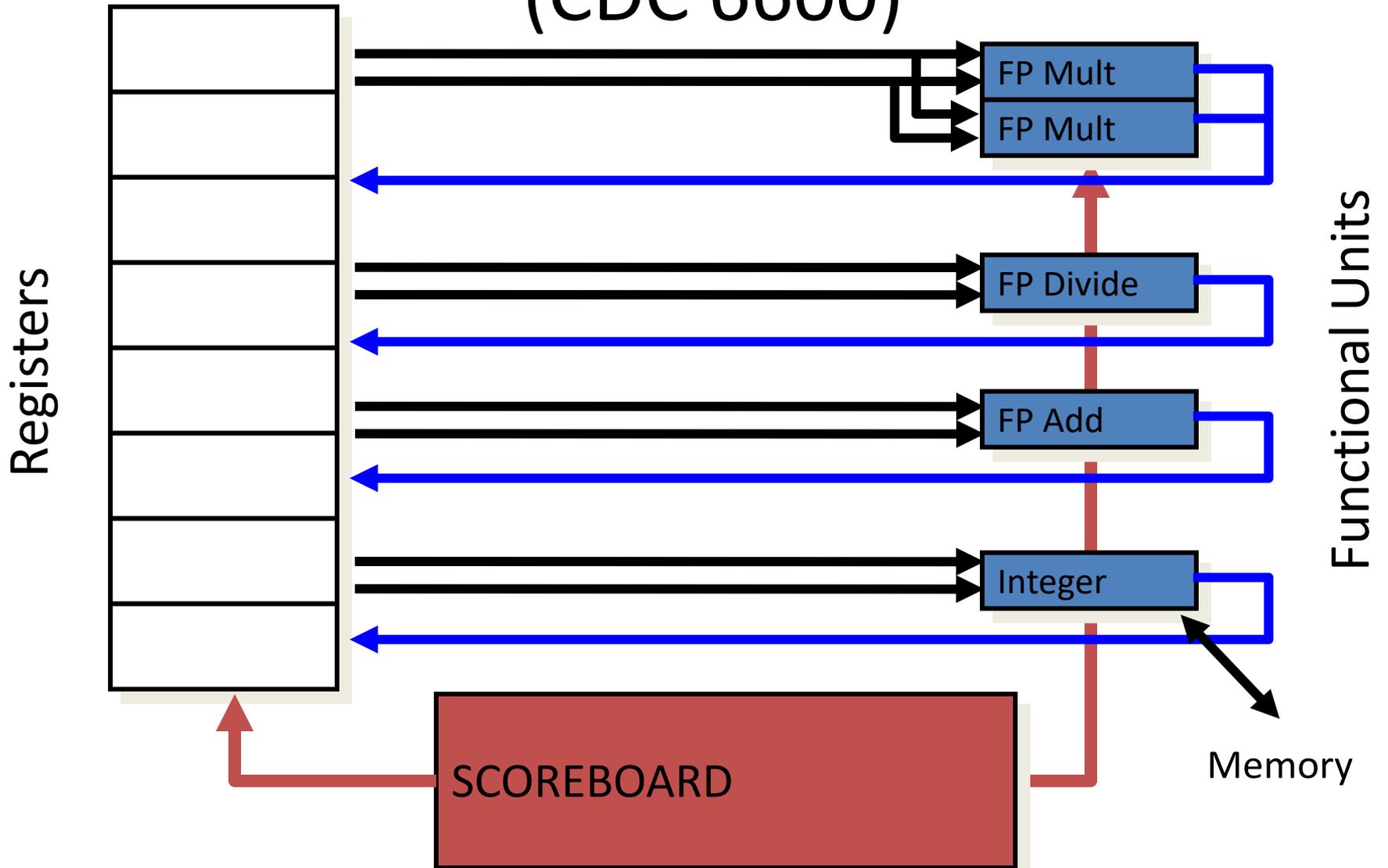


Scoreboarding i Tomasulo algoritam

Scoreboard

- Instrukcije slati ka funkcionalnim jedinicama na osnovu tipa instrukcija i da tamo čekaju argumente
- Za svaku funkcionalnu jedinicu postoji samo jedna instrukcija koja može da se rasporedi na nju u jednom trenutku – bilo da je spremna ili ne (inicijalna pretpostavka je da nema pipeline-a i tako je u primeru, a samo je različito trajanje izvršavanja na različitim funkcionalnim jedinicama)
- Dokle slati instrukcije (izdavati – issue) – ograničen broj mesta za instrukcije koje potencijalno mogu da se izvrše (zbog ograničenog broja funkcionalnih jedinica) – mašinsko ograničenje veličine data-flow prozora
- Kontrola zavisnosti po podacima i ugrađene dataflow mašine radi skup statusnih tabela - Scoreboard

Scoreboard Architecture (CDC 6600)



Dinamički raspoređivati – kako?

- Na slici – prozor je maksimalno 5 instrukcija i to dva FP množenja, po jedno FP deljenje i sabiranje i jedna jedinica load/store koja se zove integer
- Kako argumente? Ne dozvoliti dve instrukcije izdate za izvršavanje, a koje upisuju u isti registar da bi jednoznačno odredili argumente. Zadržava se in-order redosled upisa u sve registre – izlazne zavisnosti (WAW) se poštuju na račun smanjenja paralelizma!!

Čekanje argumenata

- Argument je spreman ako nijedna izdata, a nezavršena, instrukcija ne upisuje u registar za argument ili se upravo upisuje rezultat u registar za argument od strane neke instrukcije
- Kada oba argumenta postanu spremna – može izvršavanje – prave zavisnosti (RAW)
- Da su bile dozvoljene dve instrukcije koje upisuju, ne bi znali koji je upis relevantan za argument naše instrukciju, jer nema preimenovanja

Izvršavanje i upis rezultata

- Na kraju izvršavanja se mora zabeležiti da je gotovo izvršavanje
- Upis – da li smemo bezuslovno da upisujemo?
- NE – jer neka instrukcija koja koristi taj registar kao argument, možda nije očitala registar, jer još čeka drugi argument – ovo je ograničenje paralelizma zbog antizavisnosti (WAR)
- Za taj slučaj treba zakočiti funkcionalnu jedinicu (FJ)!!!
- Čim poslednja instrukcija koja je čitala taj registar pročita taj argument, upis kreće i odkoči se zakočena FJ

Kako voditi evidenciju - Scoreboard

- Tri statusne tabele
 - **Status svih izdatih instrukcija SII** – faza u izvršavanju (Izdavanje uz popunjavanje scoreboarda, čekanje i čitanje argumenata, izvršavanje, upis)
 - **Status svake funkcionalne jedinice SFJ**(da li je zauzeta, koja se operacija izvršava, odredišni registar, registri za čitanje argumenata i upis, funkcionalne jedinice koje upisuju argumente, ako nije već izračunat argument i ready flagovi za argumente)
 - **Status svih registara – SR** koje jedinice će da upišu rezultat u svaki registar, ako ima izdatih instrukcija koje će da upišu u taj registar

Zašto svaka statusna tabela (DDG)

Status svih izdatih instrukcija

- Ako je pročitala argumente, ne zadržava prepisivanje registara, ako je poslednja koja koristi argument. (smanji kašnjenje zbog antizavisnosti)
- Kada vrši upis rezultata u neki registar, sve instrukcije u prozoru koje čekaju taj rezultat ga mogu čitati od narednog ciklusa (argumenti postaju data ready)

Status svake funkcionalne jedinice

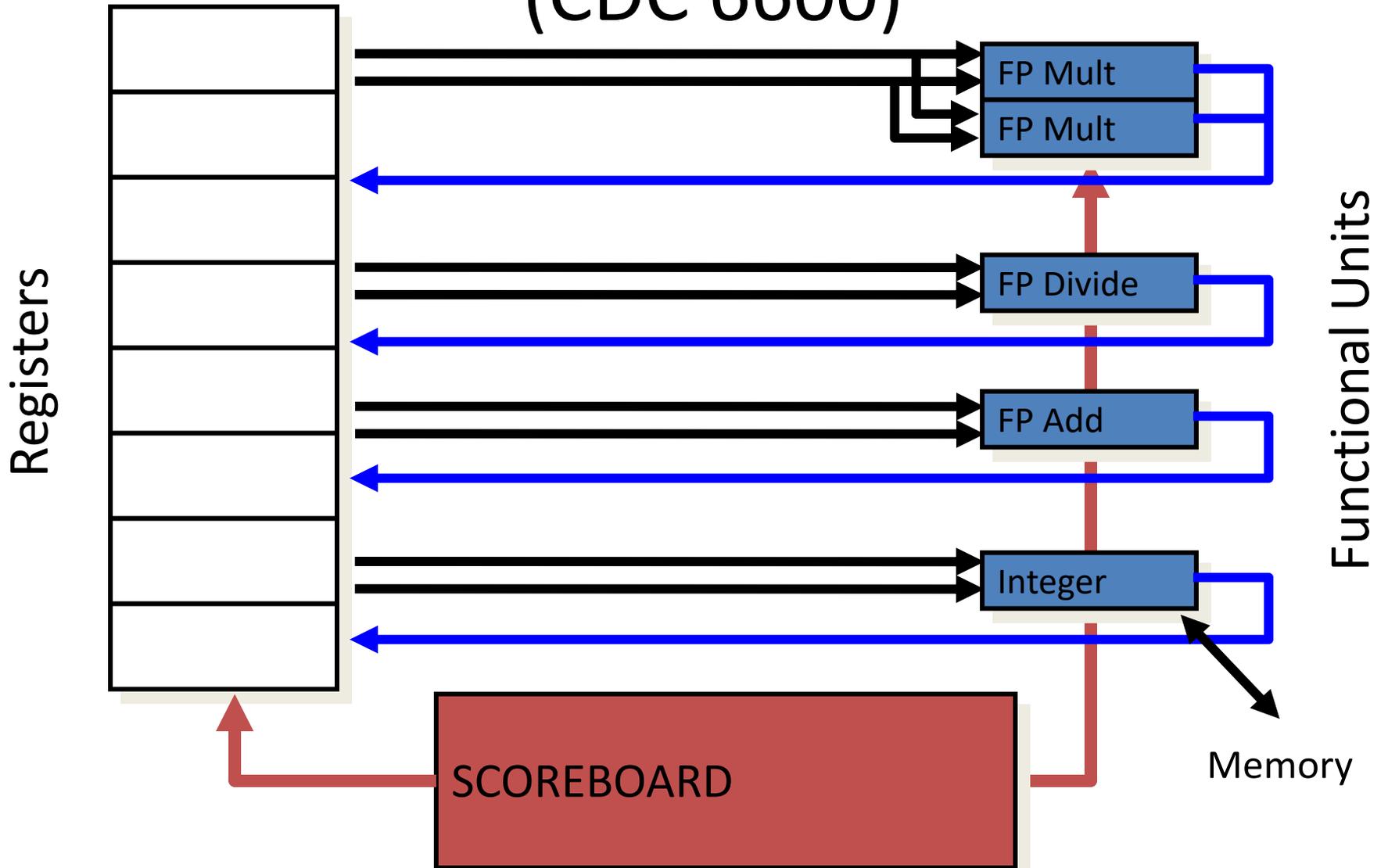
- Zauzetost (od ovoga zavisi da li neku instrukciju možemo da ubacimo u prozor)
- Registri argumenata i rezultata (zbog formiranja grafa zavisnosti)
- Koje jedinice generišu argumente (da preslikamo graf na funkcionalne jedinice)
- Da li su argumenti spremni (dinamički data ready skup)

Status svih registara (da se spreči izdavanje dve instrukcije koje upisuju u isti registar - WAW)

Koliko ima vrednosti za arhitekturnalne registre

- Jedna u svakom registru koja se koristi dok nije pročitana od strane poslednje instrukcije koja je koristi.
- Kako znamo koja je poslednja – dolazi novi upis u taj arhitekturnalni registar, pa mora biti ili u instrukcijskom prozora, ili su sve završene
- Jedna može biti zakočena na izlazu. Ta zakočena se prosleđuje do svih instrukcija koje čekaju preko registra, ali tek kada je prethodna očitana od svih WAR
- Praktično nema preimenovanja, ali je dozvoljeno napredovanje instrukcije koja prepisuje registar do upisa, kada prethodna vrednost nije očitana od svih
- Nadalje primer sa kursa na Berkley, prozor max 5 instrukcija

Scoreboard Architecture (CDC 6600)



Scoreboard Implications

- Out-of-order completion => WAR, WAW hazards?
- Solutions for WAR:
 - Stall writeback until registers have been read
 - Read registers only during Read Operands stage
- Solution for WAW:
 - Detect hazard and stall issue of new instruction until other instruction completes
- No register renaming!
- Need to have multiple instructions in execution phase => multiple execution units or pipelined execution units
- Scoreboard keeps track of dependencies between instructions that have already issued.
- Scoreboard replaces ID, EX, WB with 4 stages

Four Stages of Scoreboard Control

- **Issue**—decode instructions & check for structural hazards (ID1)
 - Instructions issued in program order (for hazard checking)
 - Don't issue if **structural hazard** (*busy u SFJ*)
 - Don't issue if instruction is **output dependent** on any previously issued but uncompleted instruction (no WAW hazards) (*destinacioni registar u SR ima nezavršen upis*)
- **Read operands**—wait until no data hazards, then read operands (ID2)
 - All real dependencies (RAW hazards) resolved in this stage, since we wait for instructions to write back data. (*čekaju se ready bitovi oba argumenta u SFJ*)
 - **No forwarding of data** in this model! (*svaki rezultat se prvo upiše u registar, pre korišćenja kao argument*)

Four Stages of Scoreboard Control

- **Execution**—operate on operands (EX)
 - The functional unit begins execution upon receiving operands. When the result is ready, it notifies the scoreboard that it has completed execution.
- **Write result**—finish execution (WB)
 - Stall until no WAR hazards with previous instructions:

Example:

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

scoreboard would stall SUBD until ADDD reads operands

Three Parts of the Scoreboard

- **Instruction status:**
Which of 4 steps the instruction is in
- **Functional unit status:**—Indicates the state of the functional unit (FU). 9 fields for each functional unit
 - Busy:** Indicates whether the unit is busy or not
 - Op:** Operation to perform in the unit (e.g., + or −)
 - Fi:** Destination register
 - Fj,Fk:** Source-register numbers
 - Qj,Qk:** Functional units producing source registers Fj, Fk
 - Rj,Rk:** Flags indicating when Fj, Fk are ready
- **Register result status**—Indicates which functional unit will write each register, if one exists. Blank when no pending instructions will write that register

Detailed Scoreboard Pipeline Control

Instruction status	Wait until	Bookkeeping
Issue	Not busy (FU) and not result(D)	$Busy(FU) \leftarrow \text{yes}; Op(FU) \leftarrow op;$ $Fi(FU) \leftarrow 'D'; Fj(FU) \leftarrow 'S1';$ $Fk(FU) \leftarrow 'S2'; Qj \leftarrow \text{Result}('S1');$ $Qk \leftarrow \text{Result}('S2'); Rj \leftarrow \text{not } Qj;$ $Rk \leftarrow \text{not } Qk; \text{Result}('D') \leftarrow FU;$
Read operands	Rj and Rk	$Rj \leftarrow \text{No}; Rk \leftarrow \text{No}$
Execution complete	Functional unit done	
Write result	$\forall f((Fj(f) \neq Fi(FU) \text{ or } Rj(f) = \text{No}) \&$ $(Fk(f) \neq Fi(FU) \text{ or } Rk(f) = \text{No}))$	$\forall f(\text{if } Qj(f) = FU \text{ then } Rj(f) \leftarrow \text{Yes});$ $\forall f(\text{if } Qk(f) = FU \text{ then } Rj(f) \leftarrow \text{Yes});$ $\text{Result}(Fi(FU)) \leftarrow 0; Busy(FU) \leftarrow \text{No}$

Scoreboard Example: Cycle 1

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Read	Exec	Write
			<i>Issue</i>	<i>Oper</i>	<i>Comp Result</i>
LD	F6	34+	R2		
LD	F2	45+	R3		
MULTD	F0	F2	F4		
SUBD	F8	F6	F2		
DIVD	F10	F0	F6		
ADDD	F6	F8	F2		

Functional unit status:

Time	Name	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				<i>Ft</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
1				Integer					

Scoreboard Example: Cycle 2

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Read Oper	Exec Comp	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			

Functional unit status:

Time	Name	Busy	Op	dest <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
2	FU Integer								

- Issue 2nd LD?

Scoreboard Example: Cycle 3

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

Functional unit status:

Time	Name	Busy	Op	dest <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	Yes	Load	F6		R2				No
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
3	FU Integer								

- Issue MULT?

Scoreboard Example: Cycle 4

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Read Oper	Exec Comp	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				

Functional unit status:

Time	Name	Busy	Op	dest Ft	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
4	FU Integer								

Scoreboard Example: Cycle 5

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Read Oper</i>	<i>Exec Comp</i>	<i>Write Result</i>
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				

Functional unit status:

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

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
5		Integer							

Scoreboard Example: Cycle 6

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Read Oper</i>	<i>Exec Comp</i>	<i>Write Result</i>
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				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest Fi</i>	<i>S1 Fj</i>	<i>S2 Fk</i>	<i>FU Qj</i>	<i>FU Qk</i>	<i>Fj? Rj</i>	<i>Fk? Rk</i>
	Integer	Yes	Load	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	No								
	Divide	No								

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
6	FU Mult1 Integer								

Scoreboard Example: Cycle 7

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
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				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	Yes	Load	F2		R3				No
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No
	Divide	No								

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
7									
<i>FU</i>	Mult1	Integer			Add				

- Read multiply operands?

Scoreboard Example: Cycle 8a

(First half of clock cycle)

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
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				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
Integer	Yes	Load	F2			R3				No
Mult1	Yes	Mult	F0	F2	F4	Integer			No	Yes
Mult2	No									
Add	Yes	Sub	F8	F6	F2		Integer		Yes	No
Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
8	<i>FU</i> Mult1	Integer			Add	Divide			

Scoreboard Example: Cycle 8b

(Second half of clock cycle)

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
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				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>						
	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:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
8	<i>FU</i> Mult1				Add	Divide			

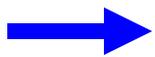
Scoreboard Example: Cycle 9

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Read	Exec	Write		
			Issue	Oper	Comp	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				

Functional unit status:

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

Note  Remaining

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
9	Mult1				Add	Divide			

- Read operands for MULT & SUB? Issue ADDD?

Scoreboard Example: Cycle 10

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
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				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>						
	Integer	No									
9	Mult1	Yes	Mult	F0	F2	F4				No	No
	Mult2	No									
1	Add	Yes	Sub	F8	F6	F2				No	No
	Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
10	<i>FU</i> Mult1				Add	Divide			

Scoreboard Example: Cycle 11

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Read Oper</i>	<i>Exec Comp</i>	<i>Write Result</i>
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				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest Fi</i>	<i>S1 Fj</i>	<i>S2 Fk</i>	<i>FU Qj</i>	<i>FU Qk</i>	<i>Fj? Rj</i>	<i>Fk? Rk</i>
	Integer	No								
8	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
0	Add	Yes	Sub	F8	F6	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
11	FU								
	Mult1				Add	Divide			

Scoreboard Example: Cycle 12

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
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				

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>						
	Integer	No									
7	Mult1	Yes	Mult	F0	F2	F4				No	No
	Mult2	No									
	Add	No									
	Divide	Yes	Div	F10	F0	F6	Mult1			No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>	
12	<i>FU</i> Mult1 Divide									

- Read operands for DIVD?

Scoreboard Example: Cycle 13

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
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:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i> <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	No								
6	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
13	Mult1			Add		Divide			

Scoreboard Example: Cycle 14

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
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	14		

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i> <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	No								
5	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
2	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
14	Mult1			Add		Divide			

Scoreboard Example: Cycle 15

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
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	14		

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i> <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	No								
4	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
1	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
15	Mult1			Add		Divide			

Scoreboard Example: Cycle 16

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
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	14	16	

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
3	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
0	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
16	<i>FU</i>	Mult1			Add		Divide			

Scoreboard Example: Cycle 17

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Read Oper	Exec Comp	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	14	16	

WAR Hazard!

Functional unit status:

Time	Name	Busy	Op	dest <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	No								
2	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		NO	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
17									
<i>FU</i>	Mult1			Add		Divide			

- Why not write result of ADD???

Scoreboard Example: Cycle 18

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
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	14	16	

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
1	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
18	<i>FU</i> Mult1			Add		Divide			

Scoreboard Example: Cycle 19

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i> <i>Fi</i>	<i>S1</i> <i>Fj</i>	<i>S2</i> <i>Fk</i>	<i>FU</i> <i>Qj</i>	<i>FU</i> <i>Qk</i>	<i>Fj?</i> <i>Rj</i>	<i>Fk?</i> <i>Rk</i>
	Integer	No								
0	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
19	Mult1			Add		Divide			

Scoreboard Example: Cycle 20

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>			<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>				
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
20				Add		Divide			

Scoreboard Example: Cycle 21

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21		
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>			<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>				
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status:

Clock		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
21	<i>FU</i>				Add		Divide			

- WAR Hazard is now gone...

Scoreboard Example: Cycle 22

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21		
ADDD	F6	F8	F2	13	14	16	22

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>						
	Integer	No									
	Mult1	No									
	Mult2	No									
	Add	No									
39	Divide	Yes	Div	F10	F0	F6				No	No

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
22	<div style="display: flex; align-items: center;"> <i>F11</i> <div style="border: 1px solid black; padding: 2px;"> Divide </div> </div>								

Scoreboard Example: Cycle 61

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	
ADDD	F6	F8	F2	13	14	16	22

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>						
	Integer	No									
	Mult1	No									
	Mult2	No									
	Add	No									
0	Divide	Yes	Div	F10	F0	F6				No	No

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
61	FU Divide								

Scoreboard Example: Cycle 62

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp</i>	<i>Result</i>	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	62
ADDD	F6	F8	F2	13	14	16	22

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>	
	Integer	No									
	Mult1	No									
	Mult2	No									
	Add	No									
	Divide	No									

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
62	<i>FU</i>								

Review: Scoreboard Example: Cycle 62

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Read Exec Write</i>				
			<i>Issue</i>	<i>Oper</i>	<i>Comp Result</i>		
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	62
ADDD	F6	F8	F2	13	14	16	22

Functional unit status:

<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>dest</i>		<i>S1</i>	<i>S2</i>	<i>FU</i>	<i>FU</i>	<i>Fj?</i>	<i>Fk?</i>
				<i>Fi</i>	<i>Fj</i>						
	Integer	No									
	Mult1	No									
	Mult2	No									
	Add	No									
	Divide	No									

Register result status:

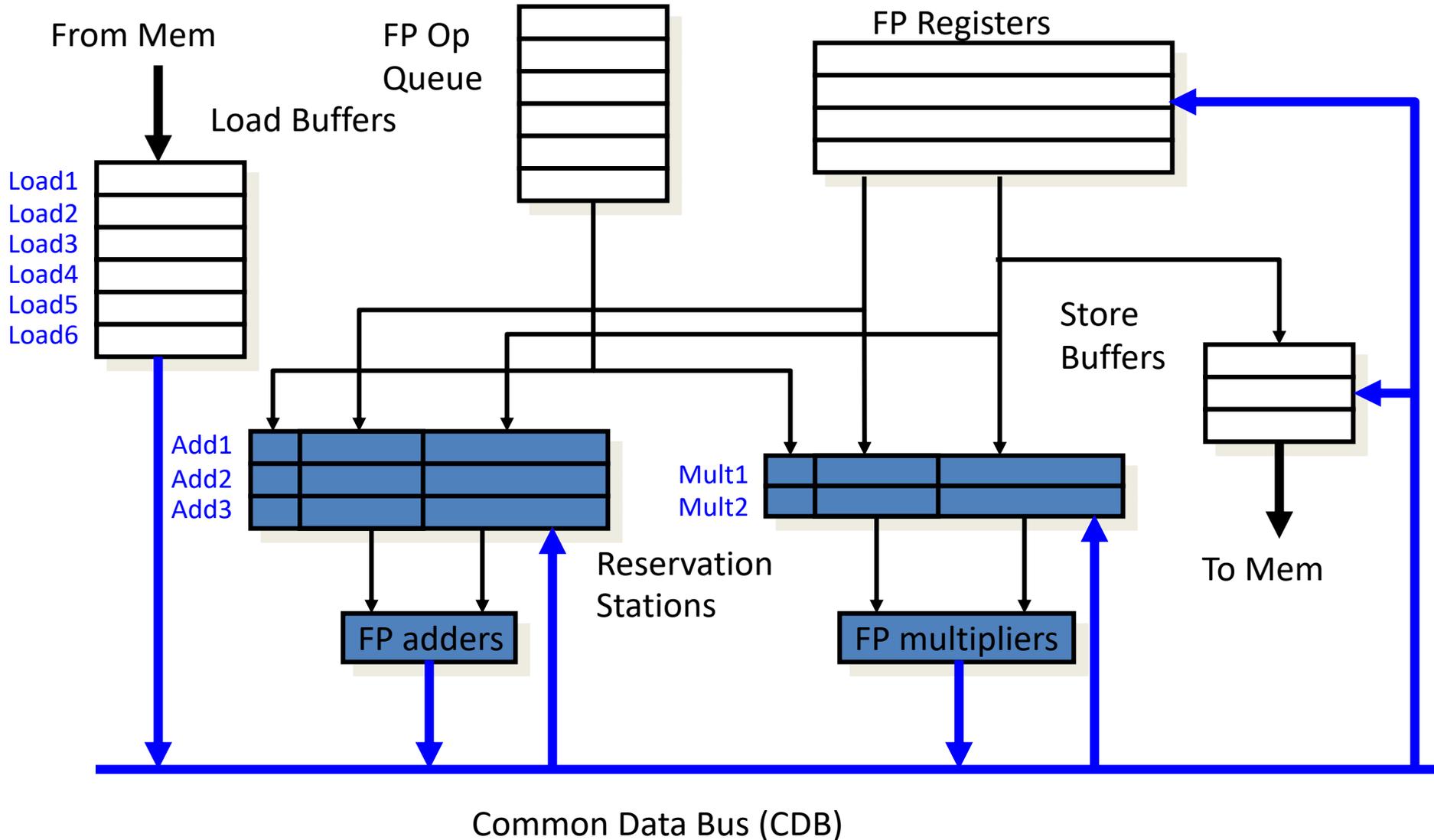
Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
62	<i>FU</i>								

- In-order issue; out-of-order execute & commit

Tomasulo algoritam

- Kako da se izbegnu izlazne zavisnosti i antizavisnosti kao ograničenje scoreboarda, eksplicitno ili implicitno preimenovanje?
- Neka se instrukcije odmah vezuju za funkcionalne jedinice koje mogu da ih izvrše, ali dozvoliti veći broj instrukcija **da se poveća instrukcijski prozor**
- Neka se za sve **izdate instrukcije rezervišu pozicije** za funkcionalne jedinice potrebne za njihovo izvršavanje (**rezervacione stanice – RS** za svaku novu instrukciju izdatu po redosledu)
- Ako **nema slobodne odgovarajuće rezervacione stanice**, zaustavlja se izdavanje instrukcija

Tomasulo Organization



Tomasulo implicitno preimenovanje

- Da li nekako može da se izvede **implicitno preimenovanje argumenata i rezultata** na osnovu mesta – rezervacione stanice operacija koje generišu taj rezultat
- Tada je preimenovani argument pokazivač na mesto u rezervacionim stanicama funkcionalne jedinice koja generiše argument!!!
- Implicitna preimenovanja – ako sva korišćenja nekog rezultata vežemo za odgovarajuću rezervacionu stanicu ⇔ mestu operacije koja vrši upis "samo jednom"
- Argument je jednoznačno vezan za rezultat operacije sa odgovarajućim mestom u rezervacionoj stanici – Obezbediti distribuciju rezultata do svih argumenata!!!

Tomasulo implicitno preimenovanje (2)

- Neka **reference za argumente** svake nove instrukcije budu **rezervacione stanice instrukcija koje generišu rezultate za argumente**, ako neka nedovršena instrukcija treba da prosledi argument
- U suprotnom, rezultat je već upisan u registar i **ODMAH** se uzima **vrednost** iz arhitekturnog registra (upisana ranije) – tada preimenovanje nije ni potrebno
- Kada kod izdavanja instrukcija naiđe novi upis u neki arhitekturni registar, njegov trenutni sadržaj nije više potreban, jer su sve rezervacione stanice koje su trebale da koriste prethodnu vrednost iz tog registra već uzele tu **vrednost** i zapamtile kod sebe u RS!!!!

Tomasulo distribucija rezultata

- Neka postoji **zajednički data bus** kojim se distribuira **jedan rezultat** sa izlaza neke rezervacione stanice **po ciklusu**
- Neka sve ostale rezervacione stanice oslušuju zajednički data bus (koji pored podataka šalje i kod rezervacione stanice čiji je to rezultat) i neka preuzimaju poslate **VREDNOSTI** za svoje instrukcije koje traže argument sa **TE REZERVACIONE STANICE EMITERA REZULTATA (Forwarding!!!)**
- Sada instrukcije koje su čekale taj rezultat u svojim RS umesto reference čuvaju **VREDNOST** preimenovanog registra preuzetu sa zajedničkog data bus-a, pa može biti više implicitnih preimenovanja za isti registar
- Sa zajedničkog data bus-a ide i u odgovarajući registar, samo poslednja vrednost upisa u arhitekturni registar po originalnom redosledu. Za **SVE prethodne nezavršene upise** u isti arhitekturni registar će se čitanje **VREDNOSTI** svih argumenta raditi **sa zajedničkog data bus-a**

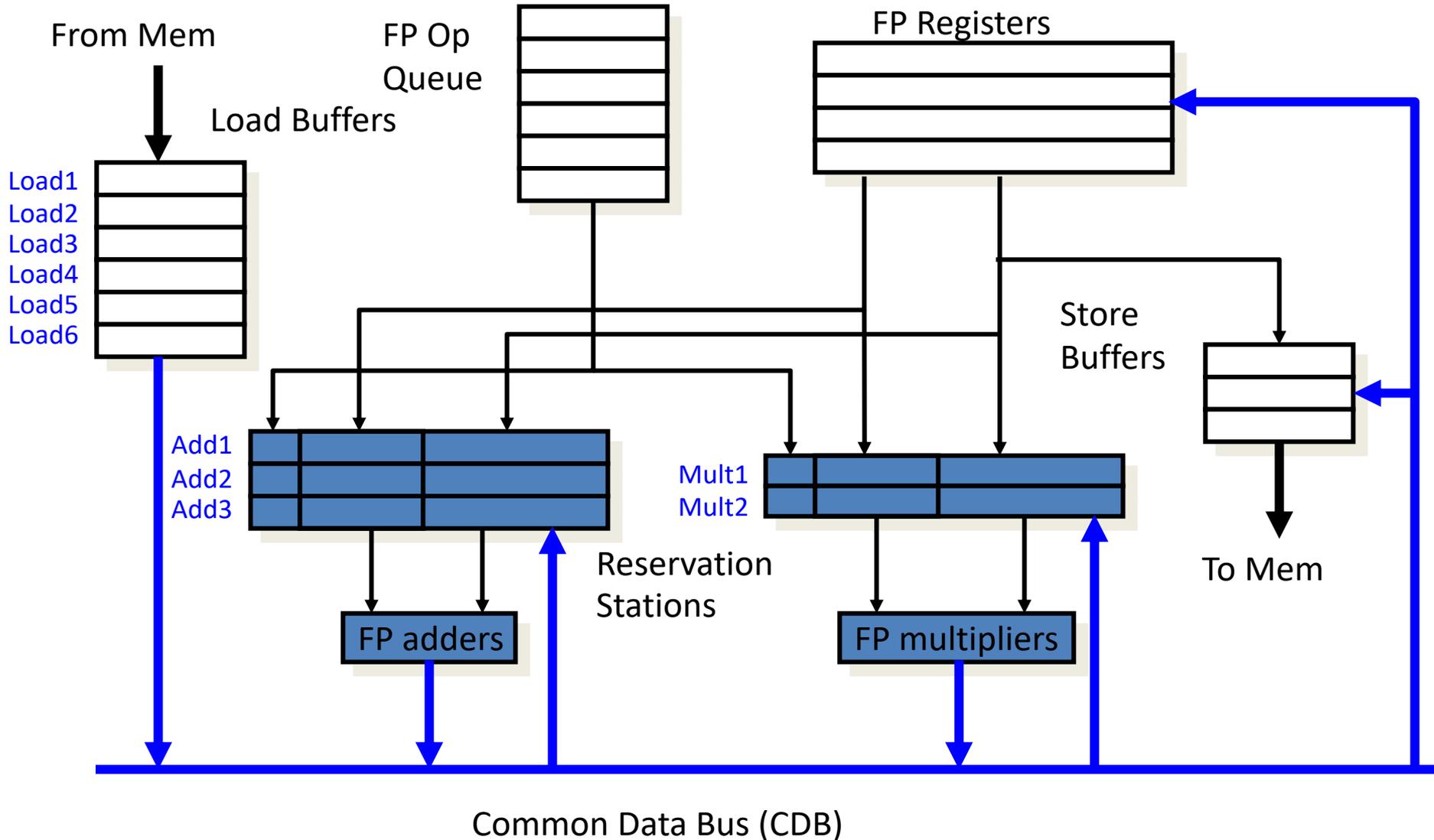
Tomasulo distribucija rezultata (2)

- Ako nijedna instrukcija u rezervacionim stanicama ne upisuje u neki registar argumenta neke instrukcije I , po izdavanju instrukcije I se povlači vrednost iz registra tog argumenta u rezervacionu stanicu instrukcije I i odmah je taj argument ready
- Registri treba da znaju koja lokacija rezervacione stanice kod izbacivanja rezultata na zajednički data bus upisuje u koji registar – zato postoji tabela za registarski status
- Prazno polje u toj tabeli – znači da nijedna izdata instrukcija ne prepisuje staru vrednost
- Vrednost polja za registar u tabeli registarskog statusa određuje da li treba da se ubacuje vrednost sa zajedničkog data bus-a u registar, a na osnovu izvora

Uloga registarskog statusa u uspostavljanju zavisnosti

- **Čuva sve poslednje upise** koji treba da se odigraju u **svakom arhitekturnom registru**, a nema ulaza ako je vrednost u registarskom fajlu jedina za arhitekturni registar
- Referisanje ovog statusa definiše šta se upisuje u argumente novih izdatih instrukcija i odmah se zna da li je rezultat ready.
- Ready je ako nema funkcionalne jedinice koja upisuje u registar argumenta ni u jednoj rezervacionoj stanici
- Svako ubacivanje nove instrukcije u rezervacione stanice (izuzev store) dovodi do novog upisa u registarski status

Tomasulo Organization



Reservation Station Components

Op: Operation to perform in the unit (e.g., + or −)

Vj, Vk: Value of Source operands

- Store buffers has V field, result to be stored

Qj, Qk: Reservation stations producing source registers (value to be written)

- Note: No ready flags as in Scoreboard; $Q_j, Q_k=0 \Rightarrow$ ready
- Store buffers only have Q_i for RS producing result

Busy: Indicates reservation station or FU is busy

Register result status—Indicates which functional unit will last write each register, if one exists. Blank when no pending instructions that will write that register.

Three Stages of Tomasulo Algorithm

1. Issue—get instruction from FP Op Queue

If reservation station free (no structural hazard),
control issues instr & sends operands (renames registers).

2. Execution—operate on operands (EX)

When both operands ready then execute;
if not ready, watch Common Data Bus for result

3. Write result—finish execution (WB)

Write on Common Data Bus to all awaiting units;
mark reservation station available

- Normal data bus: data + destination (“go to” bus)
- Common data bus: data + source (“come from” bus)
 - 64 bits of data + 4 bits of Functional Unit source address
 - Write if matches expected Functional Unit (produces result)
 - Does the broadcast

Tomasulo Example Cycle 1

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec	Write	Comp	Result	Busy	Address
LD	F6	34+	R2	1				Load1	Yes 34+R2
LD	F2	45+	R3					Load2	No
MULTD	F0	F2	F4					Load3	No
SUBD	F8	F6	F2						
DIVD	F10	F0	F6						
ADDD	F6	F8	F2						

Reservation Stations:

Time	Name	Busy	Op	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS</i> <i>Qj</i>	<i>RS</i> <i>Qk</i>
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
1				Load1					

Tomasulo Example Cycle 2

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Exec		Write Result	Busy	Address
			Issue	Comp			
LD	F6	34+	R2	1		Yes	34+R2
LD	F2	45+	R3	2		Yes	45+R3
MULTD	F0	F2	F4			No	
SUBD	F8	F6	F2			No	
DIVD	F10	F0	F6			No	
ADDD	F6	F8	F2			No	

Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
Add1		No					
Add2		No					
Add3		No					
Mult1		No					
Mult2		No					

Register result status:

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

Note: Unlike 6600, can have multiple loads outstanding

Tomasulo Example Cycle 3

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>		Busy	Address		
			<i>Issue</i>	<i>Comp Result</i>				
LD	F6	34+	R2	1	3	Load1	Yes	34+R2
LD	F2	45+	R3	2		Load2	Yes	45+R3
MULTD	F0	F2	F4	3		Load3	No	
SUBD	F8	F6	F2					
DIVD	F10	F0	F6					
ADDD	F6	F8	F2					

Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
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	Mult1	Load2		Load1					

- Note: registers names are removed (“renamed”) in Reservation Stations; MULT issued vs. scoreboard
- Load1 completing; what is waiting for Load1?

Tomasulo Example Cycle 4

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4		Load2	Yes 45+R3
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4				
DIVD	F10	F0	F6					
ADDD	F6	F8	F2					

Reservation Stations:

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

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
4	Mult1	Load2		M(A1)	Add1				

- Load2 completing; what is waiting for Load1?

Tomasulo Example Cycle 5

Instruction status:

Instruction	j	k	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4				
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2					

Reservation Stations:

Time	Name	Busy	<i>Op</i>	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
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		

Tomasulo Example Cycle 6

Instruction status:

Instruction	j	k	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4				
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

Reservation Stations:

Time	Name	Busy	Op	$S1$	$S2$	RS	RS
				V_j	V_k	Q_j	Q_k
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			

- Issue ADDD here vs. scoreboard? *Da, preimenovano je implicitno*

Tomasulo Example Cycle 7

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7			
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
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	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
7	FU	Mult1	M(A2)		Add2	Add1	Mult2		

- Add1 completing; what is waiting for it?

Tomasulo Example Cycle 8

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	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	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
8	FU								
	Mult1	M(A2)		Add2	(M-M)	Mult2			

Tomasulo Example Cycle 9

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6				

Reservation Stations:

Time	Name	Busy	Op	<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
				<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	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	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
9	FU								
	Mult1	M(A2)		Add2	(M-M)	Mult2			

Tomasulo Example Cycle 10

Instruction status:

Instruction	j	k	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10			

Reservation Stations:

Time	Name	Busy	Op	$S1$	$S2$	RS	RS
				V_j	V_k	Q_j	Q_k
	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 completing; what is waiting for it?

Tomasulo Example Cycle 11

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

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

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
11	FU								
	Mult1	M(A2)		(M-M+N)	(M-M)	Mult2			

- Write result of ADDD here vs. scoreboard? RS Mult 2 je već kupila argument iz F6

Tomasulo Example Cycle 12

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

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

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
12	FU	Mult1	M(A2)		(M-M+N)	(M-M)	Mult2		

Tomasulo Example Cycle 13

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

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

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
13	FU	Mult1	M(A2)		(M-M+N)	(M-M)	Mult2		

Tomasulo Example Cycle 14

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3			Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

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

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
14	FU	Mult1	M(A2)		(M-M+N)	(M-M)	Mult2		

Tomasulo Example Cycle 15

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15		Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

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

Register result status:

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
15	FU	Mult1	M(A2)		(M-M+N)	(M-M)	Mult2		

Tomasulo Example Cycle 16

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15	16	Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

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

Register result status:

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

Preskočeno niz ciklusa

Tomasulo Example Cycle 55

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec	Write	Busy	Address	
				Comp	Result			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15	16	Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

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

Register result status:

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

Tomasulo Example Cycle 56

Instruction status:

Instruction	<i>j</i>	<i>k</i>	<i>Exec Write</i>			Busy	Address	
			<i>Issue</i>	<i>Comp</i>	<i>Result</i>			
LD	F6	34+	R2	1	3	4	Load1	No
LD	F2	45+	R3	2	4	5	Load2	No
MULTD	F0	F2	F4	3	15	16	Load3	No
SUBD	F8	F6	F2	4	7	8		
DIVD	F10	F0	F6	5	56			
ADDD	F6	F8	F2	6	10	11		

Reservation Stations:

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

Register result status:

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

- Mult2 is completing; what is waiting for it?

Tomasulo Example Cycle 57

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Exec Comp	Write Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3	15	16	Load3
SUBD	F8	F6	F2	4	7	8	
DIVD	F10	F0	F6	5	56	57	
ADDD	F6	F8	F2	6	10	11	

Reservation Stations:

Time	Name	Busy	Op	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS</i> <i>Qj</i>	<i>RS</i> <i>Qk</i>
	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+N)	(M-M)	Mult2		

- Once again: In-order issue, out-of-order execution and completion.

Compare to Scoreboard Cycle 62

Instruction status:

Instruction	<i>j</i>	<i>k</i>	Issue	Read Oper	Exec Comp	Write Result	Issue	Exec Comp	Write Result	
LD	F6	34+	R2	1	2	3	4	1	3	4
LD	F2	45+	R3	5	6	7	8	2	4	5
MULTD	F0	F2	F4	6	9	19	20	3	15	16
SUBD	F8	F6	F2	7	9	11	12	4	7	8
DIVD	F10	F0	F6	8	21	61	62	5	56	57
ADDD	F6	F8	F2	13	14	16	22	6	10	11

- Why take longer on scoreboard/6600?
 - Structural Hazards (*vezano za RS, a ne FJ*)
 - Lack of forwarding (*forwarding štedi ciklus*)
 - Prozor je veći jer može da ima mnogo više RS od FJ

Tomasulo v. Scoreboard (IBM 360/91 v. CDC 6600)

Pipelined Functional Units (6 load, 3 store, 3 +, 2 x/÷)	Multiple Functional Units (1 load/store, 1 +, 2 x, 1 ÷)
window size: ≤ 14 instructions	≤ 5 instructions
No issue on structural hazard	same
WAR: renaming avoids	stall completion
WAW: renaming avoids	stall issue
Broadcast results from FU	Write/read registers
Control: reservation stations	central scoreboard

Mane Tomasulo algoritma

- Kompleksnost
- Puno brzih asocijativnih memorija vezanih za zajednički data bus
- Performanse ograničene zajedničkim data bus-om
- Rešenje sa više zajedničkih data bus-ova, ali to podiže kompleksnost – paralelno asocijativno pretraživanje

Zaključci

- HW koristi paralelizam na nivou instrukcija
- Zavisnosti se egzaktno određuju u izvršavanju
- Tabele pomažu u implicitnom preimenovanju - umesto registara, rezultati se jednoznačno vezuju za rezervacionu stanicu instrukcije koja je generisala rezultat
- referenca izvora podataka na zajedničkom data bus-u pomaže da sve instrukcije u RS istovremeno učitaju odgovarajuće argumente, ako je izdata instrukcija koja radi novi upis u taj arhitekturni registar
- Implicitno preimenovanje

Zaključci (2)

- In-order issue, out-of-order execution => out-of-order completion
- Prilikom dekodovanja se određuje da li postoje resursi (rezervaciona stanica) za zahtevanu funkcionalnu jedinicu, ali se odmah određuju reference na argumente ili se učitavaju vrednosti samih argumenata, ako su već izračunati
- Ključni doprinosi
 - Dinamičko raspoređivanje većih DDG (max veličina instrukcijskog prozora ugrađene data flow mašine je = broju rezervacionih stanica)
 - Implicitno preimenovanje registara