



고려대학교
KOREA UNIVERSITY

COM515 Advanced Computer Architecture

Lecture 5. Dynamic Scheduling I

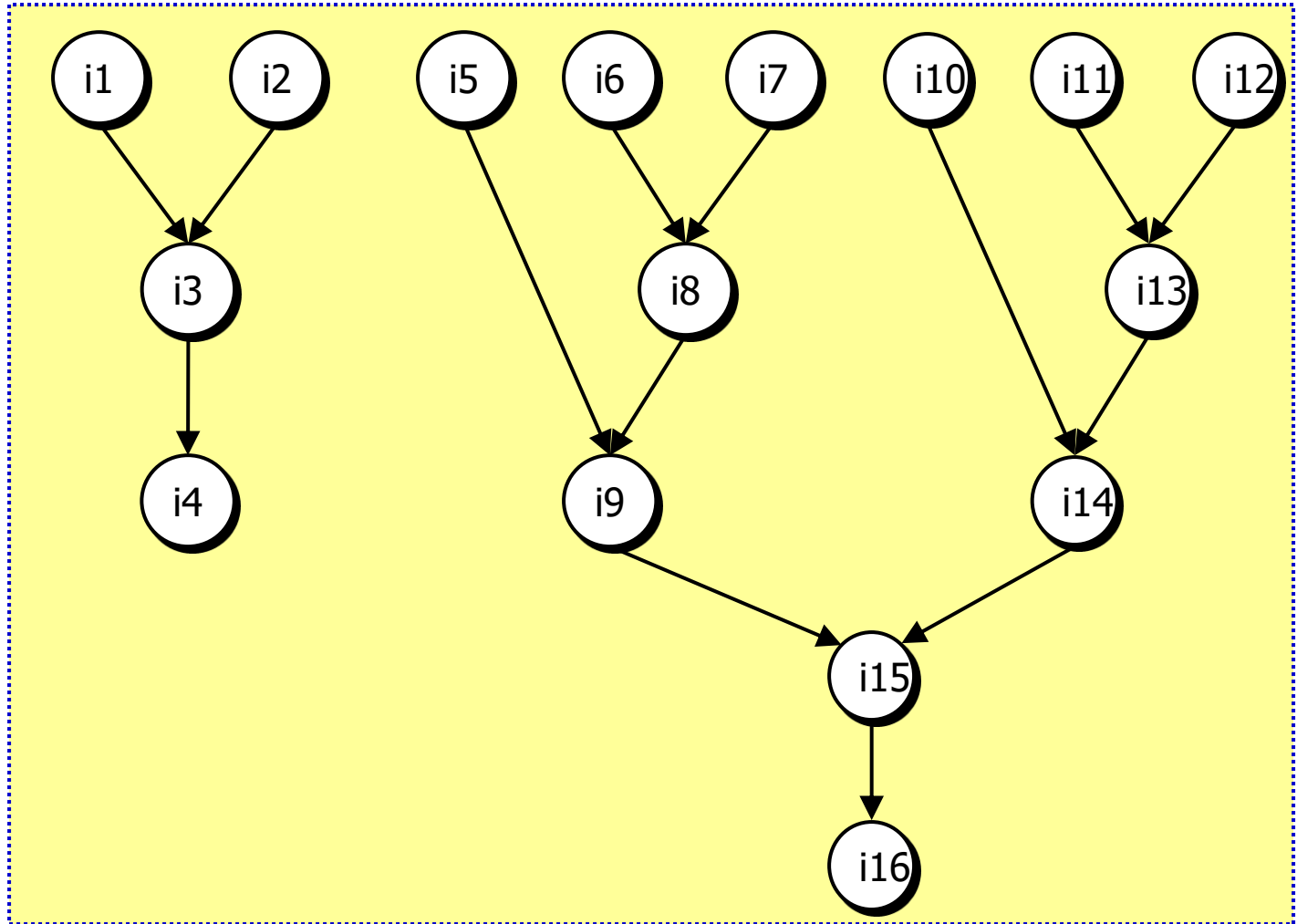
Prof. Taeweon Suh

Computer Science Education

Korea University

Data Flow Graph (DFG)

i1: $r2 = 4(r22)$
i2: $r10 = 4(r25)$
i3: $r10 = r2 + r10$
i4: $4(r26) = r10$
i5: $r14 = 8(r27)$
i6: $r6 = (r22)$
i7: $r5 = (r23)$
i8: $r5 = r6 - r5$
i9: $r4 = r14 * r5$
i10: $r15 = 12(r27)$
i11: $r7 = 4(r22)$
i12: $r8 = 4(r23)$
i13: $r8 = r7 - r8$
i14: $r8 = r15 * r8$
i15: $r8 = r4 - r8$
i16: $(r28) = r8$



Data Flow Graph (or Data Dependency Graph)

Data Flow Execution Model

- To exploit maximal ILP
- An instruction can be executed immediately after
 - All source operands are ready
 - Execution unit available
 - Destination is ready (to be written)

Dynamic Scheduling

- Exploit ILP at run-time
- Execute instructions out-of-order by a *restricted* data flow execution model (still use PC!)
- Hardware will
 - Maintain true dependency (data flow manner)
 - Maintain exception behavior
 - Find ILP within an Instruction Window (pool)
 - Need an accurate branch predictor
- Pros
 - Handle cases where dependency is unknown at compile-time
 - For example, they may involve a memory reference
 - Simplify the compiler
 - Allow code to be compiled with one pipeline in mind to run efficiently on a different pipeline
- Cons
 - Hardware complexity (main argument from the VLIW/EPIC camp)

Out-of-Order Execution

i1: $r2 = 4(r22)$

i2: $r10 = 4(r25)$

i3: $r10 = r2 + r10$

i4: $4(r26) = r10$

i5: $r14 = 8(r27)$

i6: $r6 = (r22)$

i7: $r5 = (r23)$

i8: $r5 = r6 - r5$

i9: $r4 = r14 * r5$

i10: $r15 = 12(r27)$

i11: $r7 = 4(r22)$

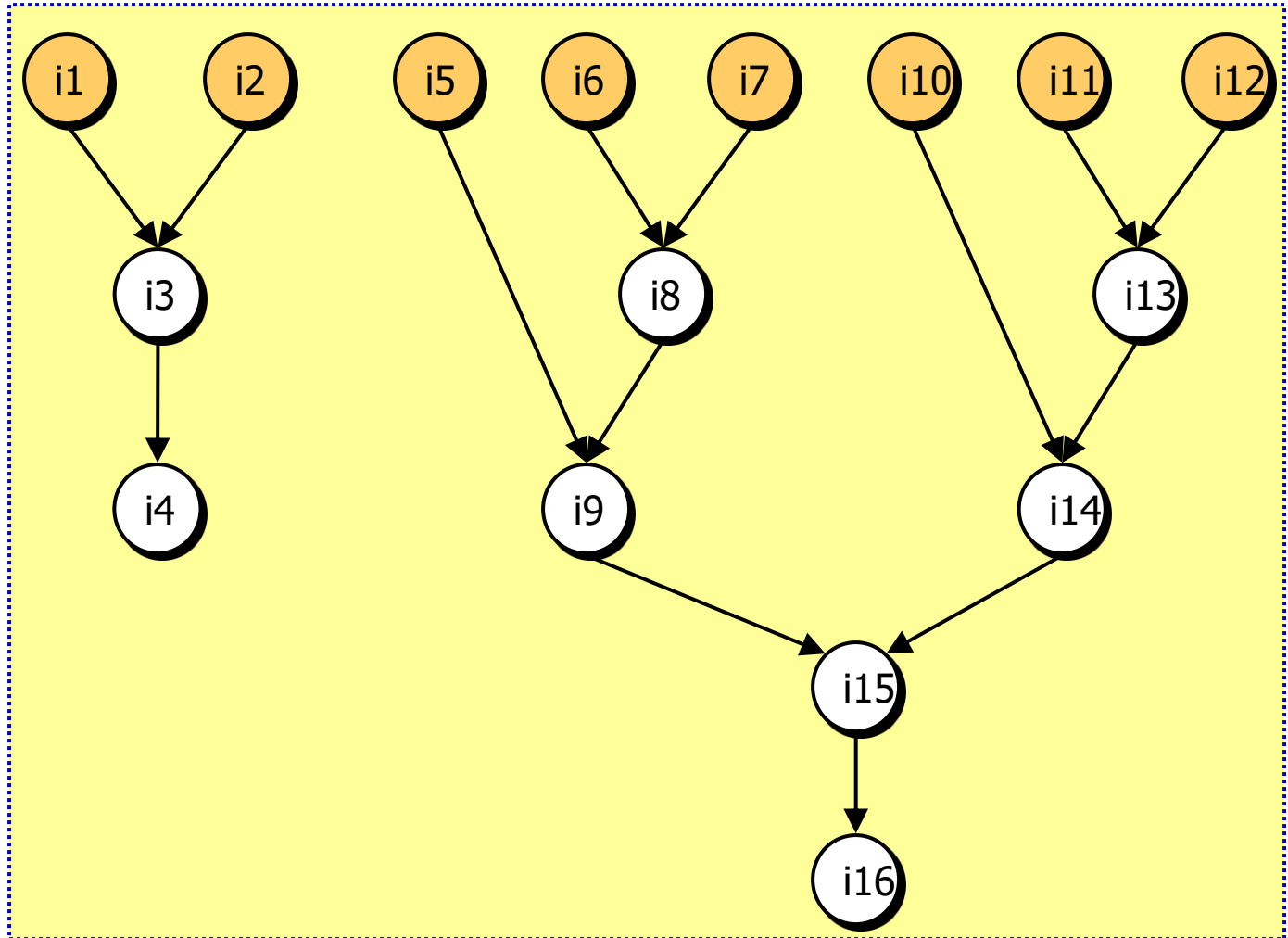
i12: $r8 = 4(r23)$

i13: $r8 = r7 - r8$

i14: $r8 = r15 * r8$

i15: $r8 = r4 - r8$

i16: $(r28) = r8$

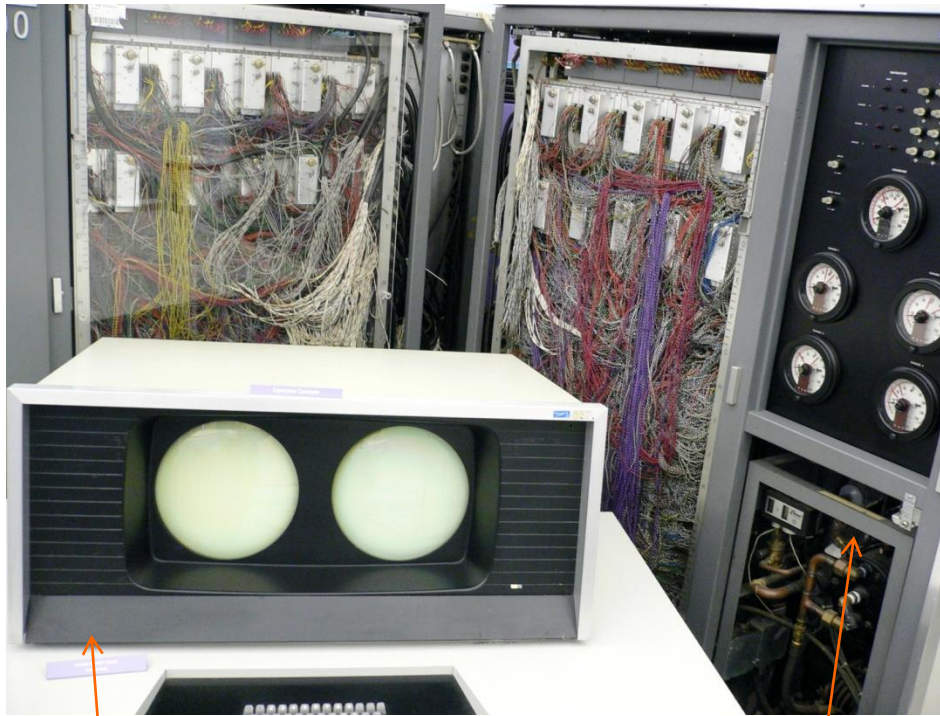


OOO Execution

- OOO execution $\equiv$ out-of-order completion
- OOO execution $\neq$ out-of-order retirement (commit)
- No (speculative) instruction allowed to retire until it is confirmed on the right path
- Fetch, decode, issue (i.e., front-end) are still done in the program order

CDC6600 (1964)

CDC: Control Data Corporation



console

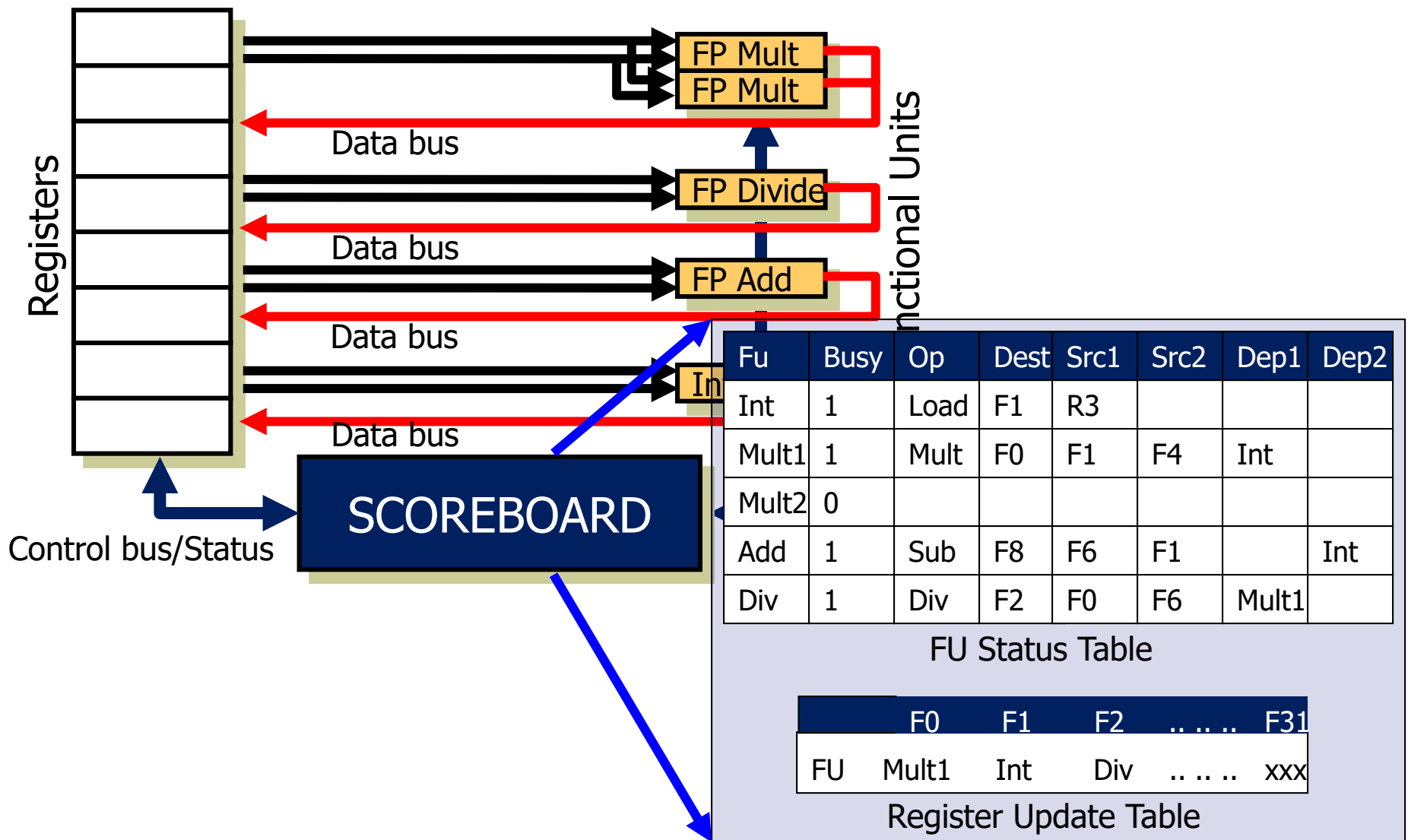
Cooling
system

- OOO execution requires
 - Multiple functional units
 - Pipelined functional units
- CDC6600 has 16 separate functional units
 - 4 FP units
 - 5 units for memory references
 - 7 units for integer operations

CDC 6600 Scoreboard Algorithm

- Enable OOO Execution to address long-latency FP instructions
- Use scoreboard tables to track
 - Functional unit status
 - Register update status
- Issue and execute instructions whenever
 - No structural hazard
 - No data hazard
- Cons
 - Stop issue when WAW is detected
 - Stop writeback when WAR is detected

CDC6600 Scoreboard (Simplified)



RAW, WAW & WAR in Scoreboard

- ID, EX, WB in the standard MIPS is replaced by **Issue**, **Read Operands**, **Execution**, **Write Result** in Scoreboard
- When the source operands are available, the scoreboard tells the functional unit to proceed to read the operands from the registers and begin execution – RAW hazard handling
- The scoreboard **issues** the instruction to the functional unit if a functional unit for the instruction is free and no other active instruction has the same destination register – WAW hazard handling
- Once the scoreboard is aware that the functional unit has completed execution, the scoreboard checks for WAR hazards and stalls the completing instruction, if necessary, in the **Write Result** stage
 - Stall the SUB.D in its **Write Result** stage until ADD.D reads its operands

```
DIV.D  F0, F2, F4
ADD.D  F10, F0, F8
SUB.D  F8, F8, F14
```

IBM 360 (first announced in 1964)

- IBM 360 introduced
 - 8-bit = 1 byte
 - 32-bit = 1 word
 - Byte-addressable memory
 - Commercial use of microcoded CPUs
 - Differentiate an “architecture” from an “implementation”
- IBM 360/91 FPU about 3 years after CDC 6600 (1966-7)
 - **Scalar** supercomputer of that time
- Tomasulo algorithm
 - Dynamic scheduling
 - Register renaming

IBM System/360 Model 91 (1967)



Tomasulo Algorithm

- Goal: High Performance without special compilers
 - Dynamic scheduling done completely by HW
 - We generally use “supercalar processor” for such category as opposed to “VLIW” or “EPIC”
- Differences between IBM 360 and CDC 6600 ISA
 - IBM has only 2 register specifiers per inst vs. 3 in CDC 6600
 - Make WAW and WAR much worse
 - IBM has 4 FP registers vs. 8 in CDC 6600
 - Smaller number of architectural registers, compiler is incapable of exploiting better register allocation
 - IBM has memory-to-register operations
- Why study? Lead to Pentium Pro/II/III/4, Core, Alpha 21264, MIPS R10000, HP 8000, PowerPC 604

IBM 360/91 FPU w/ Tomasulo Algorithm

- To not stall floating point instructions due to long latency
 - Two function units — FP Add + FP Mult/Div
 - 360/91 FPU is not pipelined
- Three new Mechanisms
 - **Reservation Stations (RS)**
 - 3 in FP Add, 2 in FP mult/div
 - Register name is discarded when issued to reservation station
 - **Tags**
 - 4-bit tag for one of the 11 possible sources (5 RSs + 6 FLB for loads)
 - Written for unavailable sources whose results are being generated by one of the sources (5 RS or 6 FLB)
 - New tag assignment eliminates false dependency
 - **Common Data Bus (CDB)**, driven by
 - 11 Sources: 5 RS + 6 FLB
 - 17 Destinations: 2*5 RS + 3 SDB + 4 FLR

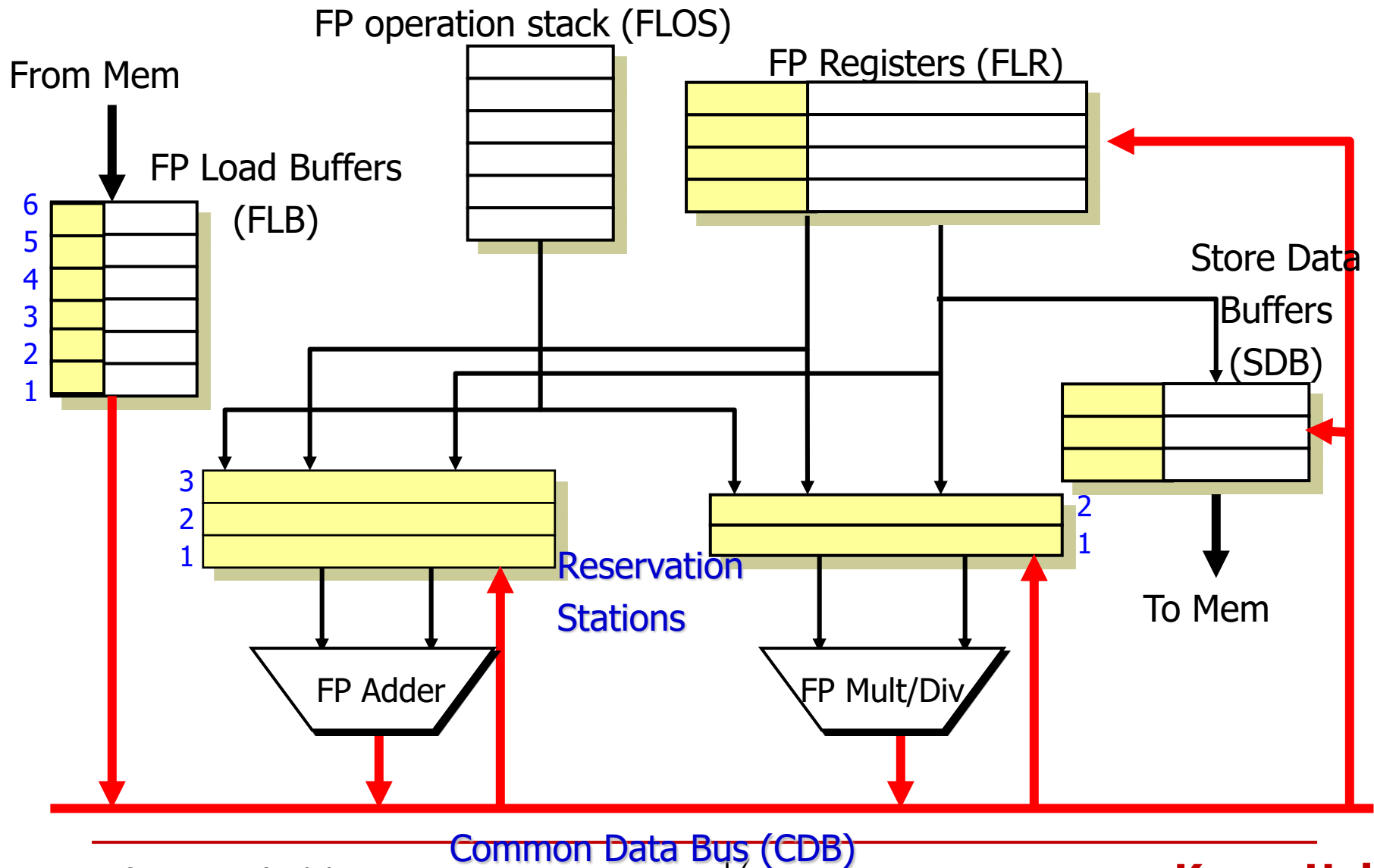
Basic Principles

- Do not rely on a centralized register file !
- RS fetches and buffers an operand as soon as it is available via CDB
 - Eliminating the need to get it from a register (No WAR)
 - Data Flow execution model
- Pending instructions designate the RS that will provide their input (renaming and maintain RAW)
- Due to in-order issue, the register status table always keeps the latest write (No WAW issue)

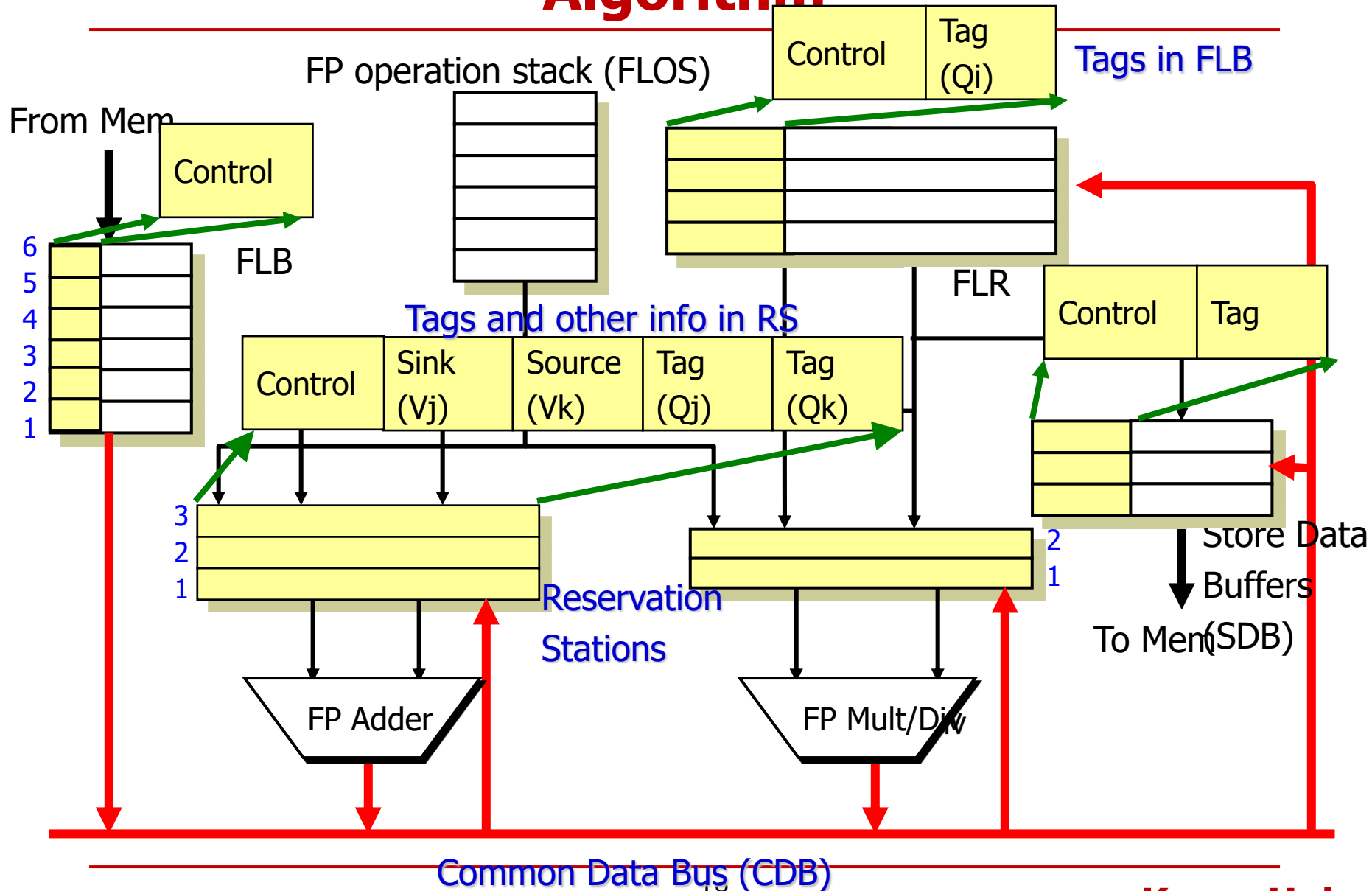
Key Representation

- Op — Operation to perform in the units
- Vj — **Value** of Source 1 (called SINK in 360/91)
- Vk — **Value** of Source 2 (called SOURCE in 360/91)
- Qj — The **RS** (tag) will produce source 1
- Qk — The **RS** (tag) will produce source 2
- A(ddress) — Hold info for the memory address generation for a load or store
- Qi — Whose value should be stored into the register

IBM 360/91 FPU w/ Tomasulo Algorithm



IBM 360/91 FPU w/ Tomasulo Algorithm



RAW Example:

i: $R2 \leftarrow R0 + R4$ (2 clks)

j: $R8 \leftarrow R0 + R2$ (2 clks)

Cycle #0:

RS	Tag	Sink	Tag	Src
1				
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2			3.5
4			10.0
8			7.8

Cycle #1: Issue i

RS	Tag	Sink	Tag	Src
1	0	6.0	0	10.0
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2	1	1	---
4			10.0
8			7.8

Cycle #2: Issue j

RS	Tag	Sink	Tag	Src
1	0	6.0	0	10.0
2	0	6.0	1	---
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2	1	1	---
4			10.0
8	1	2	---

RAW Example:

i: **R2** $\leftarrow$ R0 + R4 (2 clks)

j: R8 $\leftarrow$ R0 + **R2** (2 clks)

Cycle #3: Broadcasts tag and result: CDB_a=<RS1,16.0>

RS	Tag	Sink	Tag	Src
1				
2	0	6.0	0	16.0
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2			16.0
4			10.0
8	1	2	---

Cycle #5: Broadcasts tag and result: CDB_a=<RS2,22.0>

RS	Tag	Sink	Tag	Src
1				
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2			16.0
4			10.0
8			22.0

WAR Example:

i: $R4 \leftarrow R0 \times R8$ (3)

j: $R0 \leftarrow R4 \times R2$ (3)

k: $R2 \leftarrow R2 + R8$ (2)

Cycle #0:

RS	Tag	Sink	Tag	Src
1				
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2			3.5
4			10.0
8			7.8

Cycle #1: Issue i

RS	Tag	Sink	Tag	Src
1				
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4	0	6.0	0	7.8
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2			3.5
4	1	4	---
8			7.8

Cycle #2: Issue j

RS	Tag	Sink	Tag	Src
1				
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4	0	6.0	0	7.8
5	4	---	0	3.5

Multiplier/Divider

FLR	Busy	Tag	Data
0	1	5	---
2			3.5
4	1	4	---
8			7.8

WAR Example:

i: $R4 \leftarrow R0 \times R8$ (3)

j: $R0 \leftarrow R4 \times R2$ (3)

k: $R2 \leftarrow R2 + R8$ (2)

Cycle #3: Issue k

RS	Tag	Sink	Tag	Src
1	0	3.5	0	7.8
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4	0	6.0	0	7.8
5	4	---	0	3.5

Multiplier/Divider

FLR	Busy	Tag	Data
0	1	5	---
2	1	1	---
4	1	4	---
8			7.8

Cycle #4: Broadcasts CDB_m=<RS4,46.8>;

RS	Tag	Sink	Tag	Src
1	0	3.5	0	7.8
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5	0	46.8	0	3.5

Multiplier/Divider

FLR	Busy	Tag	Data
0	1	5	---
2	1	1	---
4			46.8
8			7.8

Cycle #5: Broadcasts CDB_a=<RS1,11.3>

RS	Tag	Sink	Tag	Src
1				
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5	0	46.8	0	3.5

Multiplier/Divider

FLR	Busy	Tag	Data
0	1	5	---
2			11.3
4			46.8
8			7.8

WAR Example:

i: $R4 \leftarrow R0 \times R8$ (3)

j: $R0 \leftarrow R4 \times R2$ (3)

k: $R2 \leftarrow R2 + R8$ (2)

Cycle #7: Broadcasts CDB_m=<RS5,163.8>

RS	Tag	Sink	Tag	Src
1				
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			163.8
2			11.3
4			46.8
8			7.8

WAW Example:

i: **R4** $\leftarrow$ R0 x R8 (3)

j: R2 $\leftarrow$ R0 + R4 (2)

k: **R4** $\leftarrow$ R0 + R8 (2)

Cycle #0:

RS	Tag	Sink	Tag	Src
1				
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2			3.5
4			10.0
8			7.8

Cycle #1: Issue i

RS	Tag	Sink	Tag	Src
1				
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4	0	6.0	0	7.8
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2			3.5
4	1	4	---
8			7.8

Cycle #2: Issue j

RS	Tag	Sink	Tag	Src
1	0	6.0	4	---
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4	0	6.0	0	7.8
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2	1	1	---
4	1	4	---
8			7.8

WAW Example:

i: **R4** $\leftarrow$ R0 x R8 (3)

j: R2 $\leftarrow$ R0 + R4 (2)

k: **R4** $\leftarrow$ R0 + R8 (2)

Cycle #3: Issue k

RS	Tag	Sink	Tag	Src
1	0	6.0	4	---
2	0	6.0	0	7.8
3				

Adder

RS	Tag	Sink	Tag	Src
4	0	6.0	0	7.8
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2	1	1	---
4	1	2	---
8			7.8

Cycle #4: Broadcasts CDB_m=<RS4,46.8>

RS	Tag	Sink	Tag	Src
1	0	6.0	0	46.8
2	0	6.0	0	7.8
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2	1	1	---
4	1	2	---
8			7.8

Cycle #5: Broadcasts CDB_a=<RS2,13.8>

RS	Tag	Sink	Tag	Src
1	0	6.0	0	46.8
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2	1	1	---
4			13.8
8			7.8

WAW Example:

i: **R4** $\leftarrow$ R0 x R8 (3)

j: R2 $\leftarrow$ R0 + R4 (2)

k: **R4** $\leftarrow$ R0 + R8 (2)

Cycle #6: Broadcasts CDB_a=<RS1,52.8>

RS	Tag	Sink	Tag	Src
1				
2				
3				

Adder

RS	Tag	Sink	Tag	Src
4				
5				

Multiplier/Divider

FLR	Busy	Tag	Data
0			6.0
2			52.8
4			13.8
8			7.8

Tomasulo Example (H&P Text)

Instruction status:

Instruction		<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Exec</i>	<i>Write</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			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

These are RS, we have only one FU for each type (MUL, ADD, LD). We reduce Load from 6 to 3 for simplicity. SDB is not shown either

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>
0	<i>Qi</i>									

Assumption

- INT (load) — 1 cycle
- MULT — 10 cycles
- ADD — 2 cycles
- DIVIDE — 40 cycles

Tomasulo Example Cycle 1

Instruction status:

				Exec	Write		
Instruction	j	k	Issue	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	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				Load1					

Tomasulo Example Cycle 2

Instruction status:

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

Reservation Stations:

				<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</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		<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	<i>Qi</i>		Load2		Load1					

Note: Unlike CDC6600, RS enables multiple outstanding loads

Load is calculating the effective address

Tomasulo Example Cycle 3

Instruction status:

				Exec	Write		
Instruction		j	k	Issue	Comp Result	Busy	Address
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:

				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	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	Qi	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:

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

Reservation Stations:

			S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj
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	Qi	Mult1	Load2		M(A1)	Add1			

- Load1 write to CDB; Load2 completing; what is waiting for Load2?

Tomasulo Example Cycle 5

Instruction status:

				Exec		Write		
Instruction		j	k	Issue	Comp	Result	Busy	Address
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:

on Stations:

				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
2	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
10	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		R(F6)	Mult1	

Register result status:

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

Tomasulo Example Cycle 6

Instruction status:

				Exec	Write		
Instruction	j	k	Issue	Comp	Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3			Load3
SUBD	F8	F6	F2	4			
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6			

Reservation Stations:

on Stations:			S1	S2	RS	RS	
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
1	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		R(F2)	Add1	
	Add3	No					
9	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		R(F6)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
6	Qi	Mult1			Add2	Add1	Mult2			

- R(F6) was entered in RS for SUBD in Cycle 5
- Issue ADDD here vs. scoreboard?

Tomasulo Example Cycle 7

Instruction status:

				Exec	Write		
Instruction	j	k	Issue	Comp	Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3			Load3
SUBD	F8	F6	F2	4	7		
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6			

Reservation Stations:

on Stations:			S1	S2	RS	RS	
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
0	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		R(F2)	Add1	
	Add3	No					
8	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		R(F6)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
7	Qi	Mult1			Add2	Add1	Mult2			

- Add1 completing; what is waiting for it?

Tomasulo Example Cycle 8

Instruction status:

				Exec		Write		
Instruction		j	k	Issue	Comp	Result	Busy	Address
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	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
2	Add2	Yes	ADDD	M1-M2	R(F2)		
	Add3	No					
7	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		R(F6)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
8	Qi	Mult1			Add2	M1-M2	Mult2			

Tomasulo Example Cycle 9

Instruction status:

				Exec	Write		
Instruction	j	k	Issue	Comp	Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3			Load3
SUBD	F8	F6	F2	4	7	8	
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6			

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
1	Add2	Yes	ADDD	M1-M2	R(F2)		
	Add3	No					
6	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		R(F6)	Mult1	

Register result status:

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

Tomasulo Example Cycle 10

Instruction status:

				Exec	Write		
Instruction		<i>j</i>	<i>k</i>	Issue	Comp	Result	Busy Address
LD	F6	34+	R2	1	3	4	Load1 Load2 Load3
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		

Reservation Stations:

				<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
Time	Name	Busy	Op	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	Add1	No					
0	Add2	Yes	ADDD	M1-M2	R(F2)		
	Add3	No					
5	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		R(F6)	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>
10	FU	Mult1			Add2		Mult2			

- Add2 completing; what is waiting for it?

Tomasulo Example Cycle 11

Instruction status:

				Exec	Write		
Instruction	j	k	Issue	Comp	Result	Busy	Address
LD	F6	34+	R2	1	3	4	Load1
LD	F2	45+	R3	2	4	5	Load2
MULTD	F0	F2	F4	3			Load3
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	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		R(F6)	Mult1	

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
11	FU	Mult1		(M1-M2+M(A2))		Mult2			

- Write result of ADDD here vs. scoreboard?
- All quick instructions complete in this cycle!

Tomasulo Example Cycle 12

Instruction status:

				<i>Exec</i>		<i>Write</i>		
Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</i>		Busy	Address
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:

on Stations:				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		R(F6)	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					Mult2			

Tomasulo Example Cycle 13

Instruction status:

Instruction		<i>j</i>	<i>k</i>	Issue	Exec Comp	Write Result		Busy	Address
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	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		R(F6)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
13	FU	Mult1					Mult2			

Tomasulo Example Cycle 14

Instruction status:

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

Reservation Stations:

on Stations:

				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		R(F6)	Mult1	

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
14	FU								
	Mult1					Mult2			

Tomasulo Example Cycle 15

Instruction status:

				Exec	Write		
Instruction	j	k	Issue	Comp	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		Load3
SUBD	F8	F6	F2	4	7	8	
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6	10	11	

Reservation Stations:

on Stations:				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		R(F6)	Mult1	

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
15	FU								
	Mult1					Mult2			

Tomasulo Example Cycle 16

Instruction status:

				<i>Exec</i>		<i>Write</i>		
Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>Comp</i>	<i>Result</i>		Busy	Address
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:

on Stations:

				<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</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	R(F6)		

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 Mult2								

**Faster than light computation
(skip a couple of cycles)**

Tomasulo Example Cycle 55

Instruction status:

				Exec	Write		
Instruction	<i>j</i>	<i>k</i>	Issue	Comp	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			
ADDD	F6	F8	F2	6	10	11	

Reservation Stations:

				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
1	Mult2	Yes	DIVD	M*F4	R(F6)		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
55	FU								
						Mult2			

Tomasulo Example Cycle 56

Instruction status:

				Exec	Write		
Instruction		<i>j</i>	<i>k</i>	Issue	Comp	Result	Busy Address
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:

				<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
Time	Name	Busy	Op	<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	R(F6)		

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						Mult2			

- Mult2 is completing; what is waiting for it?

Tomasulo Example Cycle 57

Instruction status:

				Exec	Write		
Instruction		<i>j</i>	<i>k</i>	Issue	Comp	Result	Busy Address
LD	F6	34+	R2	1	3	4	Load1 No Load2 No Load3 No
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	

Reservation Stations:

				<i>S1</i>	<i>S2</i>	<i>RS</i>	<i>RS</i>
Time	Name	Busy	Op	<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		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
57	FU						Result			

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

Compare to Scoreboard Cycle 62

Scoreboard

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	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

	Issue	Exec Comp	Write Result
	1	3	4
	2	4	5
	3	15	16
	4	7	8
	5	56	57
	6	10	11

- Why take longer on scoreboard/6600?
 - No register renaming (WAR, WAW handling)
 - Lack of forwarding (CDB in IBM)

Issues in Tomasulo Algorithm

- CDB at high speed?
- Precise exception issues
- Speculative instructions
 - Branch prediction enlarges instruction window
 - How to rollback when mispredicted?

Backup Slides