Computer Organization and Architecture

Instructor: Dr. Rushdi Abu Zneit

Slide Sources: Based on CA: aQA by Hennessy/Patterson.

Advanced Topic: Dynamic Scheduling with Tomasulo's Algorithm

Data Hazards Review

- *RAW* (*read after write*) hazard:
  - instruction I occurs before instruction J in the program but…
  - …instruction J tries to read an operand before instruction I writes to it, so J incorrectly gets the old value
  - Example:

```
…
I: LW R1, 0(R2)
…
J: DADDU R3, R1, R4
…
```

Note: see CA:aQA Sec. 2.12 for MIPS64 ISA information

- A RAW hazard is a *true data dependence*, where there is a programmer-mandated flow of data from one instruction (the producer) to another (the consumer)
  - therefore, the consumer *must wait* for the producer to finish computing and writing

Data Hazards Review

- *WAW* (*write after write*) hazard:
  - instruction I occurs before instruction J in the program but…
  - …instruction J tries to write an operand before instruction I writes to it, so the wrong order of writes causes the destination register to end up with the value from I rather than that from J
  - Example:

```
…
I: DSUBU R1, R2, R3
…
J: DADDU R1, R3, R4
…
```

- A WAW hazard is a not a true data dependence, but rather a kind of *name dependence*, called *output dependence* , because of the (avoidable?) same name of the destination registers
- WAW hazards cannot occur in the classic 5-stage MIPS integer pipeline. *Why…?*
  - registers are *written only in one stage*, the WB stage, and
  - instructions enter the pipeline *in order*
- However, we shall deal with situations where instructions may be executed *out of order*…

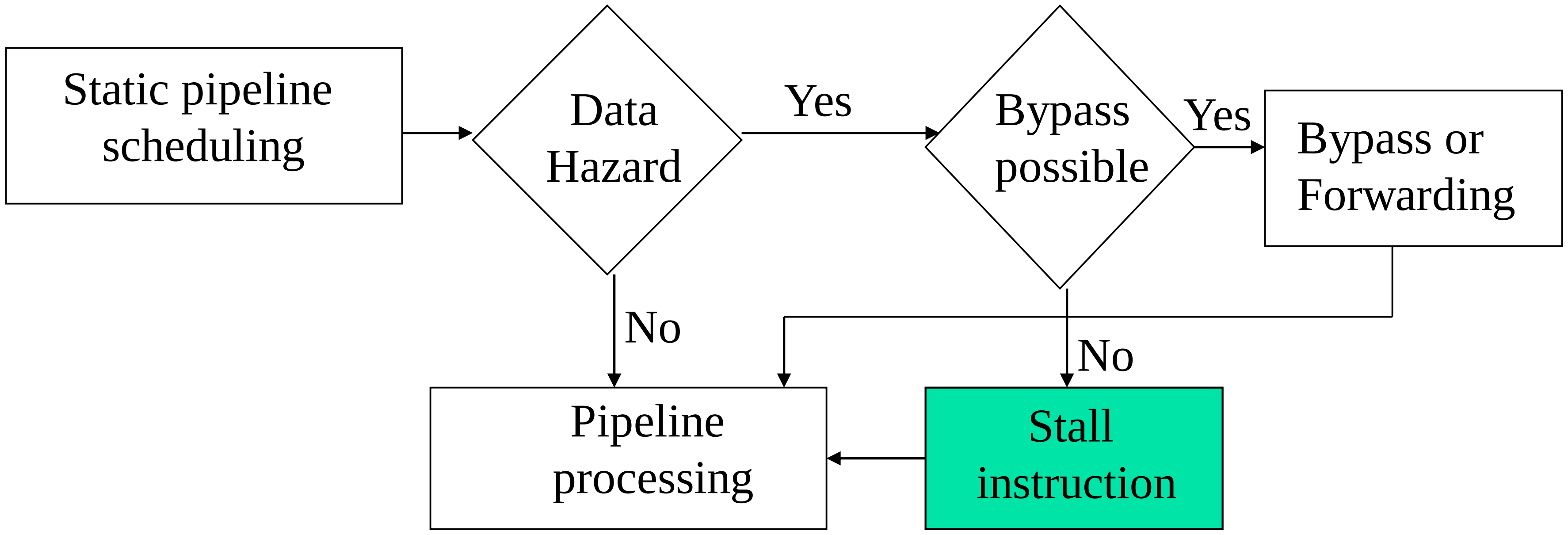
Data Hazards Review

- *WAR* (*write after read*) hazard:
  - instruction I occurs before instruction J in the program but…
  - …instruction J tries to write an operand before instruction I reads it, so I incorrectly gets the later value
  - Example:

```
…
I: DSUBU R2, R1, R3
…
J: DADDU R1, R3, R4
…
```

- A WAR hazard is a not a true data dependence, but rather a kind of *name dependence*, called *antidependence*, because of the (avoidable?) shared name of two registers
- WAR hazards cannot occur in the classic 5-stage MIPS integer pipeline. *Why…?*
  - registers are *read early* and *written late*
  - instructions enter the pipeline *in order*
- However, we shall deal with situations where instructions may be executed *out of order*…

Why Dynamic Scheduling…?



Dynamic scheduling
reduces this stall via
ILP: <u>Instruction Level Parallelism</u>

Goal of ILP: To get as many instructions as possible executing in parallel while respecting dependencies

Dynamic Scheduling: Key Ideas

- Old paradigm (classic MIPS 5-stage integer pipeline):
  - *in-order instruction issue* and *execution*
  - can cause *unnecessary delay* of instructions that also *wastes hardware resources* by keeping them idle through the delay
  - e.g.,

```
DIV.D F0, F2, F4
ADD.D F6, F0, F8    # ADD.D and S.D are stalled by
S.D   F6, 0(R1)     # true data dependences
SUB.D F8, F10, F14  # SUB.D and MUL.D are ready to execute
MUL.D F6, F10, F8   # but blocked by previous stalls!
```

Dynamic Scheduling: Key Ideas

- New paradigm:
  - *in-order issue* but allow *out-of-order execution* (i.e., ILP = parallel execution of instructions) and, therefore, *out-of-order completion*
  - e.g.,

```
DIV.D F0, F2, F4
ADD.D F6, F0, F8
S.D   F6, 0(R1)
SUB.D F8, F10, F14
MUL.D F6, F10, F8
```

  - *without waiting* for `ADD.D` and `S.D` to complete execution try to execute `SUB.D` and `MUL.D`
  - this out-of-order execution raises two *potential hazards* that do not exist in the classic pipeline with in-order execution
    - WAR hazard: the antidependence between `ADD.D` and `SUB.D`
    - WAW hazard: the output dependence between `ADD.D` and `MUL.D`

Dynamic Scheduling: Key Ideas

- solution: eliminate WAR and WAW hazards by *register renaming*
- e.g.,

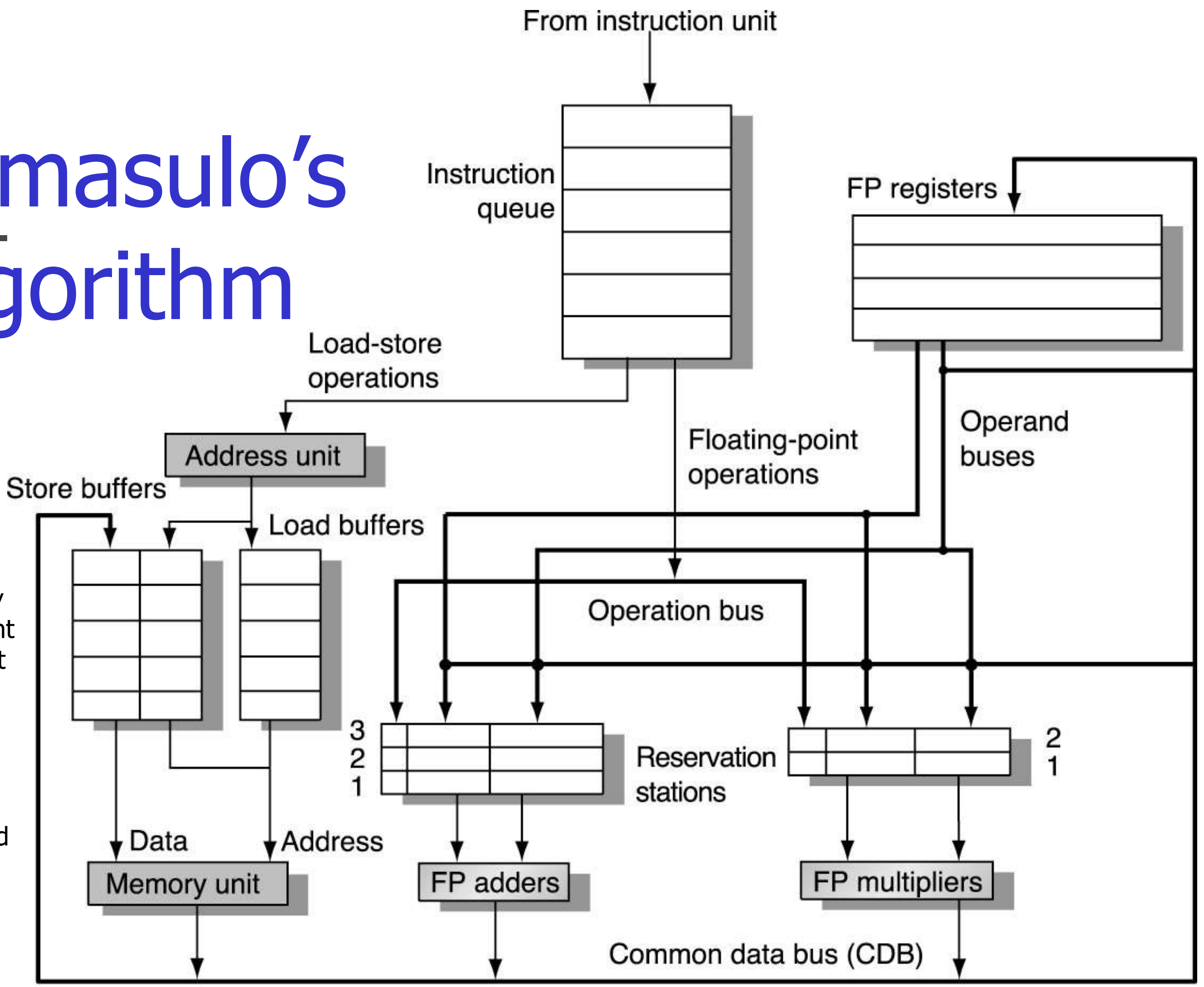
```
DIV.D F0, F2, F4
ADD.D S , F0, F8
S.D   S , 0(R1)
SUB.D  T, F10, F14
MUL.D F6, F10, T
```

- <u>Tomasulo provides register renaming via *reservation stations*</u>
  - reservation stations *fetch and buffer* an operand as soon as it is available, eliminating need to go to register to get operand
  - pending instructions designate reservation stations that will provide input values
  - results are passed directly from *functional units* where they are computed to the reservation stations where they are required over the *common data bus* (*CDB*) – *bypassing* registers

Tomasulo's Algorithm

Note: reservations stations do *not* form a queue! They all have independent access to FP op unit

Note: there may be multiple *or* pipelined FP op units – conceptually same!



Basic structure of MIPS floating-point unit based on Tomasulo

Tomasulo’s Algorithm: Three Stages

1. Issue: get instruction from Instruction Queue
   - if reservation station free (no structural hazard), control issues instruction to reservation station, and sends to reservation station operand values (or reservation station source for values)
2. Execution: operate on operands (EX)
   - when both operands ready then execute; if not ready, watch CDB for result
3. Write result: finish execution (WB)
   - write on CDB to all awaiting units; mark reservation station available

Tomasulo’s Algorithm: Data Structures

Reservation station fields

- **Op**: Operation to be performed on source operands S1 and S2
- **Qj**, **Qk**: The reservation stations that will produce the corresponding operand; value of 0 indicates source operand is *already available* in Vj or Vk, or is unnecessary
- **Vj**, **Vk**: The value of the source operands. Only one of the V or Q fields is valid for each operand. For loads, Vk field holds offset
- **A**: Holds information for the memory address calculation for load and store. Initially, immediate field of instruction is stored here; after address calculation, effective address is stored
- **Busy**: Reservation station and related functional unit occupied

Register file field

- **Qi**: Number of the reservation station that contains the operation whose results will be stored into this register; value of 0 (or blank) indicates value is register contents, i.e., no instruction targets this register

Examples

```
1.    L.D       F6,  34(R2)
2.    L.D       F2,  45(R3)
3.    MUL.D     F0,  F2, F4
4.    SUB.D     F8,  F2, F6
5.    DIV.D     F10, F0, F6
6.    ADD.D     F6,  F8, F2
```

We run Tomasulo's algorithm on the above code sequence in three different examples:

A. *Data structures when the only the first load has completed*
B. *Data structures when* `MUL.D` *is about to write*
C. *Data structures cycle by cycle*

Example A: Instructions

Instruction	Instruction Status		
	Issue	Execute	Write Results
L.D F6, 34(R2)	X	X	X
L.D F2, 45(R3)	X	X	
MUL.D F0, F2, F4	X		
SUB.D F8, F2, F6	X		
DIV.D F10, F0, F6	X		
ADD.D F6, F8, F2	X		

All instructions have issued but only the first `L.D` has completed and written its result

Example A: Reservation Stations

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	no						
Load2	yes	LOAD					45 + Regs[R3]
Add1	yes	SUB		Mem[34+Regs[R2]]	Load2		
Add2	yes	ADD			Add1	Load2	
Add3	no						
Mult1	yes	MUL		Regs[F4]	Load2		
Mult2	yes	DIV		Mem[34+Regs[R2]]	Mult1		

Addi indicates i*th* reservation station for the FP add unit, etc.

Example A: Registers

Field	F0	F2	F4	F6	F8	F10	F12……….F30
Qi	Mult1	Load2		Add2	Add1	Mult2	

Floating point registers

Notes

- *The CDB allows an operand to be broadcast* as soon as its value is computed in a functional unit
  - allows multiple instructions awaiting that value to be released simultaneously
- *WAW and WAR hazards are eliminated by renaming registers* using reservation stations and by storing operands into reservation stations as soon as they become available. E.g., the WAR hazard between `DIV.D` and `ADD.D` involving F6 is eliminated in both cases:
  - if the `L.D` instruction providing the 2nd operand of `DIV.D` *has completed* (case shown), then Vk stores the result, making `DIV.D` independent of `ADD.D`
  - If the `L.D` instruction providing the 2nd operand of `DIV.D` *has not completed*, then Qk points to the Load1 reservation station, again making `DIV.D` independent of `ADD.D`

Notes

- Instructions pass through the *issue stage in order* but can *bypass one another in the execute stage* and *complete out of order*.
- *Why must instructions issue in order?*
  - when an instruction issues to a free reservation station it looks up its *operand registers* for either the operand value itself (V value from the register's data) *or* the reservation station that will produce the value (Q value from the register's status field)
  - additionally, the instruction will write its own reservation station number to its *destination register's* status field
  - now suppose instructions

    ```
     SUB.D F2, F4, F6
    ADD.D F8, F2, F4
    ```

    issue in order. *How is the* `F2` *register's status field set and how are the* `ADD.D` *reservation station's Q and V fields set?*
  - *what happens if the instructions are issued in reverse order?!*
- See CA: aQA Fig. 3.5 for algorithm details of Tomasulo

Example B: Instructions

Instruction	Instruction Status		
	Issue	Execute	Write Results
L.D F6, 34(R2)	X	X	X
L.D F2, 45(R3)	X	X	X
MUL.D F0, F2, F4	X	X	
SUB.D F8, F2, F6	X	X	X
DIV.D F10, F0, F6	X		
ADD.D F6, F8, F2	X	X	X

When MUL.D is about to write

Example B: Reservation Stations

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	no						
Load2	no						
Add1	no						
Add2	no						
Add3	no						
Mult1	yes	MUL	Mem[45+Regs[R3]]	Regs[F4]			
Mult2	yes	DIV		Mem[34+Regs[R2]]	Mult1		

Addi indicates i*th* reservation station for the FP add unit, etc.

Example B: Registers

Field	F0	F2	F4	F6	F8	F10	F12……….F30
Qi	Mult1					Mult2	

Floating point registers

Latencies

- Assume operation latencies
  - load: 2 clock cycles
  - add/sub: 2 clock cycles
  - multiply: 10 clock cycles
  - divide: 40 clock cycles

Example C: Cycle 0

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34+	R2			
LD	F2	45+	R3			
MULTI	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

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

Register result status

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

Example C: Cycle 1

Instruction status					Execution	Write
Instruction		*j*	*k*	*Issue*	*complete*	*Result*
LD	F6	34+	R2	1		
LD	F2	45+	R3			
MULTI	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				*S1*	*S2*	*RS for j*	*RS for k*
Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*
0	Add1	No					
0	Add2	No					
	Add3	No					
0	Mult1	No					
0	Mult2	No					

Register result status

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

Example C: Cycle 2

<u>Instruction status</u>

Instruction		*j*	*k*	*Issue*	*Execution complete*	*Write Result*
LD	F6	34+	R2	1		
LD	F2	45+	R3	2		
MULTI	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	

<u>Reservation Stations</u>

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

<u>Register result status</u>

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

Example C: Cycle 3

<u>Instruction status</u>

Instruction		*j*	*k*	*Issue*	*Execution complete*	*Write Result*
LD	F6	34+	R2	1	3	
LD	F2	45+	R3	2		
MULTI	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	

<u>Reservation Stations</u>

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

<u>Register result status</u>

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

Example C: Cycle 4

Instruction status					Execution	Write
Instruction		j	k	Issue	complete	Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2		
MULTI	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				S1	S2	RS for j	RS for k
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
0	Add1	Yes	SUBD	M(34+R2)			Load2
0	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD		R(F4)	Load2	
0	Mult2	No					

Register result status										
Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
4	FU	Mult1	Load2		M(34+R2)	Add1				

Example C: Cycle 5

Instruction status		j	k	Issue	Execution complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	
MULTI	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
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 for j Qj	RS for k Qk
0	Add1	Yes	SUBD	M(34+R2)			Load2
0	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD		R(F4)	Load2	
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
5	FU	Mult1	Load2		M(34+R2)	Add1	Mult2			

Example C: Cycle 6

<u>Instruction status</u>

Instruction		*j*	*k*	*Issue*	*Execution complete*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULTI	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	

<u>Reservation Stations</u>

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

<u>Register result status</u>

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
6	*FU*	Mult1	M(45+R3)		Add2	Add1	Mult2			

Example C: Cycle 7

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULTI	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 for j Qj	RS for k Qk
1	Add1	Yes	SUBD	M(34+R2)	M(45+R3)		
0	Add2	Yes	ADDD		M(45+R3)	Add1	
	Add3	No					
9	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
7	FU	Mult1	M(45+R3)		Add2	Add1	Mult2			

Example C: Cycle 8

Instruction status					Execution	Write
Instruction		*j*	*k*	*Issue*	*complete*	*Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULTI	F0	F2	F4	3		
SUBD	F8	F6	F2	4	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 for j Qj*	*RS for k Qk*
0	Add1	Yes	SUBD	M(34+R2)	M(45+R3)		
0	Add2	Yes	ADDD		M(45+R3)	Add1	
	Add3	No					
8	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
8	*FU*	Mult1	M(45+R3)		Add2	Add1	Mult2			

Example C: Cycle 9

<u>Instruction status</u>

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

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

<u>Reservation Stations</u>

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

<u>Register result status</u>

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
9	FU	Mult1	M(45+R3)		Add2	M()–M()	Mult2			

Example C: Cycle 10

Instruction status					Execution	Write			
Instruction		*j*	*k*	*Issue*	*complete*	*Result*		Busy	Address
LD	F6	34+	R2	1	3	4	Load1	No	
LD	F2	45+	R3	2	5	6	Load2	No	
MULTI	F0	F2	F4	3			Load3	No	
SUBD	F8	F6	F2	4	8	9			
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6					

Reservation Stations				*S1*	*S2*	*RS for j*	*RS for k*
Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*
0	Add1	No					
2	Add2	Yes	ADDD	M()–M()	M(45+R3)		
	Add3	No					
6	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status										
Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
10	*FU*	Mult1	M(45+R3)		Add2	M()–M()	Mult2			

Example C: Cycle 11

Instruction status					Execution	Write
Instruction		*j*	*k*	*Issue*	*complete*	*Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULTI	F0	F2	F4	3		
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

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

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
11	*FU*	Mult1	M(45+R3)		Add2	M()–M()	Mult2			

Example C: Cycle 12

Instruction status

Instruction		*j*	*k*	*Issue*	*Execution complete*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULTI	F0	F2	F4	3		
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

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

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
12	*FU*	Mult1	M(45+R3)		Add2	M()–M()	Mult2			

Example C: Cycle 13

Instruction status

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

	Busy	Address
Load1	No	
Load2	No	
Load3	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					
	Add3	No					
3	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
13	*FU*	Mult1	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

Example C: Cycle 14

Instruction status

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

	Busy	Address
Load1	No	
Load2	No	
Load3	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					
	Add3	No					
2	Mult1	Yes	MULTD	M(45+R3)	R(F4)		
0	Mult2	Yes	DIVD		M(34+R2)	Mult1	

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
14	*FU*	Mult1	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

Example C: Cycle 15

Instruction status					Execution	Write		Busy	Address
Instruction		*j*	*k*	*Issue*	*complete*	*Result*			
LD	F6	34+	R2	1	3	4	Load1	No	
LD	F2	45+	R3	2	5	6	Load2	No	
MULTI	F0	F2	F4	3			Load3	No	
SUBD	F8	F6	F2	4	8	9			
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6	12	13			

Reservation Stations

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

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
15	*FU*	Mult1	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

Example C: Cycle 16

<u>Instruction status</u>

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULTD	F0	F2	F4	3	16	
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	13

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

<u>Reservation Stations</u>

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

<u>Register result status</u>

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

Example C: Cycle 17

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULTI	F0	F2	F4	3	16	17
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	13

	Busy	Address
Load1	No	
Load2	No	
Load3	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					
	Add3	No					
0	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
17	FU	M*F4	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

Example C: Cycle 18

Instruction status					Execution	Write		Busy	Address
Instruction	*j*	*k*		*Issue*	*complete*	*Result*			
LD	F6	34+	R2	1	3	4	Load1	No	
LD	F2	45+	R3	2	5	6	Load2	No	
MULTI	F0	F2	F4	3	16	17	Load3	No	
SUBD	F8	F6	F2	4	8	9			
DIVD	F10	F0	F6	5					
ADDD	F6	F8	F2	6	12	13			

Reservation Stations				*S1*	*S2*	*RS for j*	*RS for k*
Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*
0	Add1	No					
0	Add2	No					
	Add3	No					
0	Mult1	No					
40	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
18	*FU*	M*F4	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

Example C: Cycle 57

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULTI	F0	F2	F4	3	16	17
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	13

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

				S1	S2	RS for j	RS for k
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
0	Add1	No					
0	Add2	No					
	Add3	No					
0	Mult1	No					
1	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
57	FU	M*F4	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

Example C: Cycle 58

Instruction status

Instruction		j	k	Issue	Execution complete	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULTI	F0	F2	F4	3	16	17
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5	58	
ADDD	F6	F8	F2	6	12	13

	Busy	Address
Load1	No	
Load2	No	
Load3	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					
	Add3	No					
0	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(34+R2)		

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
58	FU	M*F4	M(45+R3)		(M–M)+M()	M()–M()	Mult2			

Example C: Cycle 59

Instruction status

Instruction		*j*	*k*	*Issue*	*Execution complete*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULTI	F0	F2	F4	3	16	17
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5	58	59
ADDD	F6	F8	F2	6	12	13

	Busy	Address
Load1	No	
Load2	No	
Load3	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					
	Add3	No					
0	Mult1	No					
0	Mult2	No					

Register result status

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
59	*FU*	M*F4	M(45+R3)		(M–M)+M()	M()–M()	M*F4/M			

Tomasulo Loop Example

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

- Assume multiply takes 4 clocks
- Assume first load takes 8 clocks (cache miss?), second load takes 4 clocks (hit)
- To be clear, will show clocks for `SUBI`, `BNEZ`
- Reality: integer instructions ahead

Loop Example Cycle 0

Instruction status

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

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

	Busy	Qi
Store1	No	
Store2	No	
Store3	No	

Reservation Stations

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

Code:

LD	F0	0	R1
MULTI	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									

Loop Example Cycle 1

Instruction status

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

	Busy	Address
Load1	Yes	80
Load2	No	
Load3	No	

	Busy	Qi
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
MULTI	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							

Loop Example Cycle 2

Instruction status

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

				S1	S2	RS for j	RS for k
Time	Name	Busy	Op	Vj	Vk	Qj	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
MULTD	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						

Loop Example Cycle 3

Instruction status

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

- **Note: MULT1 has no registers names in RS**

Loop Example Cycle 4

Instruction status

Instruction	j	k	iteration	Issue	Execution complete	Write Result		Busy	Address
LD F0	0	R1	1	1			Load1	Yes	80
MULTI F4	F0	F2	1	2			Load2	No	
SD F4	0	R1	1	3			Load3	No	

	Busy	Address	Qi
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

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

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
MULTI 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					

Loop Example Cycle 5

Instruction status						Execution	Write			
Instruction	j	k	iteration		Issue	complete	Result		Busy	Address
LD	F0	0	R1	1	1			Load1	Yes	80
MULTI	F4	F0	F2	1	2			Load2	No	
SD	F4	0	R1	1	3			Load3	No	
LD	F0	0	R1	2						
MULTI	F4	F0	F2	2						
SD	F4	0	R1	2						

	Busy	Address	Qi
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

Reservation Stations

				S1	S2	RS for j	RS for k
Time	Name	Busy	Op	Vj	Vk	Qj	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
MULTI	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						

Loop Example Cycle 6

Instruction status						*Execution*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1			Load1	Yes	80		
MULTI	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	
MULTI	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						MULTI	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						

- **Note: F0 never sees Load1 result**

Loop Example Cycle 7

Instruction status						Execution	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1			Load1	Yes	80		
MULTI	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	
MULTI	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						MULTI	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						

- **Note: MULT2 has no registers names in RS**

Loop Example Cycle 8

Instruction status

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

Loop Example Cycle 9

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

				S1	S2	RS for j	RS for k
Time	Name	Busy	Op	Vj	Vk	Qj	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
MULTD 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 completing; what is waiting for it?**

Loop Example Cycle 10

Instruction status

Instruction		j	k	iteration	Issue	Execution complete	Write Result
LD	F0	0	R1	1	1	9	10
MULTI	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0	R1	2	6	10	
MULTI	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
MULTI	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 completing; what is waiting for it?**

Loop Example Cycle 11

Instruction status						*Execution*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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	
MULTI	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						MULTI	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						

Loop Example Cycle 12

Instruction status						*Execution*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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	
MULTI	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						MULTI	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						

Loop Example Cycle 13

Instruction status						*Execution*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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	
MULTI	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						MULTI	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						

Loop Example Cycle 14

Instruction status						*Execution*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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	
MULTI	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						MULTI	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 completing; what is waiting for it?**

Loop Example Cycle 15

Instruction status						*Execution*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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	
MULTI	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						MULTI	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 completing; what is waiting for it?**

Loop Example Cycle 16

Instruction status						Execution	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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	
MULTI	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						MULTI	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						

Loop Example Cycle 17

Instruction status						*Execution*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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	
MULTI	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						MULTI	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						

Loop Example Cycle 18

Instruction status						*Execution*	*Write*					
Instruction		*j*	*k*	*iteration*	*Issue*	*complete*	*Result*		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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	
MULTI	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(72	
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						MULTI	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						

Loop Example Cycle 19

Instruction status						Execution	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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			
MULTI	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						MULTI	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
19		56	Qi	Load3		Mult1						

Loop Example Cycle 20

Instruction status						Execution	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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			
MULTI	F4	F0	F2	2	7	15	16	Store2	Yes	72	M(72)*R(72	
SD	F4	0	R1	2	8	20		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						MULTI	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*
20		**56**	*Qi*	Load3		Mult1						

Loop Example Cycle 21

Instruction status						Execution	Write					
Instruction		j	k	iteration	Issue	complete	Result		Busy	Address		
LD	F0	0	R1	1	1	9	10	Load1	No			
MULTI	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			
MULTI	F4	F0	F2	2	7	15	16	Store2	No			
SD	F4	0	R1	2	8	20	21	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						MULTI	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*
21		**56**	*Qi*	Load3		Mult1						

Tomasulo Summary

- Advantages
  - prevents registers from being the bottleneck
  - eliminates WAR, WAW hazards
  - allows loop unrolling in HW
  - common data bus (CDB) broadcasts results to multiple instructions
- Disadvantages
  - hardware complexity
  - performance limited by associative stores required from CDB to reservation stations
  - performance limited by CDB bandwidth (CDB = bottleneck)
- Lasting Contributions
  - dynamic scheduling
  - register renaming
  - load/store disambiguation
- Original Tomasulo implementation was on IBM 360/91
  - famous modern descendants: Pentiums, PowerPCs, MIPS R10000,…

Notes

- See Tomasulo simulations at our Additional Resources page
  - webHase Tomasulo Simulation
  - McGill University Tomasulo Simulation