CPEN 411: Computer Architecture

Slide Set #7: Tomasulo’s Algorithm

Original slides: Tor Aamodt

aamodt@ece.ubc.ca

Background: Pentium Pro die photo (first x86 with out of order execution)

Introduction to Slide Set 7

Introduction to Slide Set 7

- Once we enable out-of-order execution we have to consider both WAW and WAR hazards. The (out-of-order) scoreboard algorithm handles these hazards by **stalling**.

Introduction to Slide Set 7

- Once we enable out-of-order execution we have to consider both WAW and WAR hazards.   The (out-of-order) scoreboard algorithm handles these hazards by **stalling**.
- Notice that no value is communicated between instructions involved in these WAW or WAR hazards.

Introduction to Slide Set 7

- Once we enable out-of-order execution we have to consider both WAW and WAR hazards. The (out-of-order) scoreboard algorithm handles these hazards by **stalling**.
- Notice that no value is communicated between instructions involved in these WAW or WAR hazards.
- In this lecture we will see that it is possible to eliminate the “name dependencies” that require stalling by having the hardware “rename” the locations causing the “name dependencies”.

Introduction to Slide Set 7

- Once we enable out-of-order execution we have to consider both WAW and WAR hazards. The (out-of-order) scoreboard algorithm handles these hazards by **stalling**.
- Notice that no value is communicated between instructions involved in these WAW or WAR hazards.
- In this lecture we will see that it is possible to eliminate the "name dependencies" that require stalling by having the hardware "rename" the locations causing the "name dependencies".
- The specific hardware algorithm we will examine for doing this is called "Tomasulo's Algorithm". It is widely used in today's computers.

Learning Objectives

After we finish going through these slides you will be able to…

- Describe Tomasulo’s algorithm in detail.
- Explain the key difference between the Scoreboard algorithm and Tomasulo’s algorithm.
- Evaluate instruction timing of MIPS assembly code sequences using Tomasulo’s algorithm.

Removing Name Dependencies

Removing Name Dependencies

- Name dependencies result from reusing a register:

Removing Name Dependencies

- Name dependencies result from reusing a register:

```
LD        R1, 0(R2)
```

Removing Name Dependencies

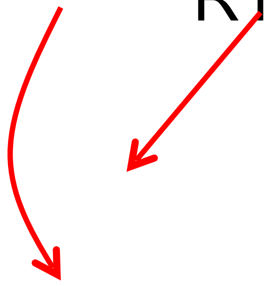
- Name dependencies result from reusing a register:

```
LD          R1, 0(R2)
```

Removing Name Dependencies

- Name dependencies result from reusing a register:

```
LD          R1, 0(R2)
```



Removing Name Dependencies

- Name dependencies result from reusing a register:

```
LD          R1, 0(R2)
DADD        R3,R1,R3
```

Removing Name Dependencies

- Name dependencies result from reusing a register:

```
LD        R1, 0(R2)
DADD      R3,R1,R3
DADDUI    R2,R2,#8     ; reuse R2
```

Removing Name Dependencies

- Name dependencies result from reusing a register:

```
LD          R1, 0(R2)
DADD        R3,R1,R3
DADDUI      R2,R2,#8      ; reuse R2
LD          R1, 0(R2)     ; reuse R1
```

Removing Name Dependencies

- Name dependencies result from reusing a register:

```
LD          R1, 0(R2)
DADD        R3,R1,R3
DADDUI      R2,R2,#8       ; reuse R2
LD          R1, 0(R2)      ; reuse R1
```

Removing Name Dependencies

- Name dependencies result from reusing a register:

```
LD        R1, 0(R2)
DADD      R3,R1,R3
DADDUI    R2,R2,#8     ; reuse R2
LD        R1, 0(R2)    ; reuse R1
```

- Adding registers to the ISA can <u>reduce</u> name dependencies. Requires recompilation to use the additional registers.

Removing Name Dependencies

- Name dependencies result from reusing a register:

```
LD        R1, 0(R2)
DADD      R3,R1,R3
DADDUI    R2,R2,#8      ; reuse R2
LD        R1, 0(R2)     ; reuse R1
```

- Adding registers to the ISA can <u>reduce</u> name dependencies. Requires recompilation to use the additional registers.
- Ideally, we want to add registers to hardware and have hardware use them <u>without</u> requiring recompilation.

Removing Name Dependencies

- Name dependencies result from reusing a register:

```
LD          R1, 0(R2)
DADD        R3,R1,R3
DADDUI      R2,R2,#8       ; reuse R2
LD          R1, 0(R2)      ; reuse R1
```

- Adding registers to the ISA can <u>reduce</u> name dependencies. Requires recompilation to use the additional registers.
- Ideally, we want to add registers to hardware and have hardware use them <u>without</u> requiring recompilation.
- Also, if we can somehow <u>remove</u> the name dependencies, we can entirely avoid WAW or WAR hazards.

Tomasulo Algorithm: Historical Context

Tomasulo Algorithm: Historical Context

- Developed in mid 1960’s for floating-point unit of IBM 360 model 91

Tomasulo Algorithm: Historical Context

- Developed in mid 1960's for floating-point unit of IBM 360 model 91
- IBM 360 model 91 designed for scientific computing (e.g., NASA)

Tomasulo Algorithm: Historical Context

- Developed in mid 1960's for floating-point unit of IBM 360 model 91
- IBM 360 model 91 designed for scientific computing (e.g., NASA)
- Notion of instruction set compatibility invented for IBM System/360.

Tomasulo Algorithm: Historical Context

- Developed in mid 1960’s for floating-point unit of IBM 360 model 91
- IBM 360 model 91 designed for scientific computing (e.g., NASA)
- Notion of instruction set compatibility invented for IBM System/360.
- 4 FP registers

Tomasulo Algorithm: Historical Context

- Developed in mid 1960’s for floating-point unit of IBM 360 model 91
- IBM 360 model 91 designed for scientific computing (e.g., NASA)
- Notion of instruction set compatibility invented for IBM System/360.
- 4 FP registers
- Anticipated FP unit latencies of 6 cycles for multiply, 18 for divide, 2 for addition; 8 cycles to access memory; no cache.

Historical Motivation

Historical Motivation

- Design goal for System/360 model 91:
  - Sustain CPI as close as possible to 1 on floating-point code

Historical Motivation

- Design goal for System/360 model 91:
  - Sustain CPI as close as possible to 1 on floating-point code
- Constraints:
  1. Binary compatibility had to be maintained. Four registers visible to programmer. Code with many WAW hazards since required fewer instructions in 360/91 ISA.
  2. Long floating-point function unit latencies
  3. Separate pipelined floating point adder and multiplier (higher clock frequency)
  - **Today's Motivation:** Above + cache misses + parallelism between branches < 2.

Historical Motivation

- Design goal for System/360 model 91:
  - Sustain CPI as close as possible to 1 on floating-point code
- Constraints:
  1. Binary compatibility had to be maintained. Four registers visible to programmer. Code with many WAW hazards since required fewer instructions in 360/91 ISA.
  2. Long floating-point function unit latencies
  3. Separate pipelined floating point adder and multiplier (higher clock frequency)
  - **Today's Motivation:** Above + cache misses + parallelism between branches < 2.
- Design tradeoffs Tomasulo <u>considered</u>
  - Adding busy bits to register file (in-order scoreboard)
  - Adding "working registers" near function units to improve clock frequency
  - Additional function units to eliminate structural hazards
  - Impact of software techniques such as "loop unrolling"

Historical Motivation

- Design goal for System/360 model 91:
  - Sustain CPI as close as possible to 1 on floating-point code
- Constraints:
  1. Binary compatibility had to be maintained. Four registers visible to programmer. Code with many WAW hazards since required fewer instructions in 360/91 ISA.
  2. Long floating-point function unit latencies
  3. Separate pipelined floating point adder and multiplier (higher clock frequency)
  - **Today's Motivation:** Above + cache misses + parallelism between branches < 2.
- Design tradeoffs Tomasulo <u>considered</u>
  - Adding busy bits to register file (in-order scoreboard)
  - Adding "working registers" near function units to improve clock frequency
  - Additional function units to eliminate structural hazards
  - Impact of software techniques such as "loop unrolling"
- Tomasulo proposed a design that combined elements of these solutions and was significant improvement. Still used today (e.g. Intel Core i5, i7)

Which code is faster? Option A or Option B?
Both compute "A+B+C+D*E".

```
/* Option A */
LD       F0, D
LD       F1, C
LD       F2, B
MULT.D   F0, E
ADD.D    F1, F0
ADD.D    F2, A
ADD.D    F1,F2
```

```
/* Option B */
LD       F0, E
MULT.D   F0, D
ADD.D    F0, C
ADD.D    F0,B
ADD.D    F0, A
```

A: Code for "Option A" is faster

vation

oint code

gisters visible to programmer. Code
ructions in 360/91 ISA.

lier (higher clock frequency)

rallelism between branches < 2.

rd)

nprove clock frequency

ards

ling"

nents of these solutions and was

tel Core i5, i7)

Which code is faster? Option A or Option B?
Both compute "A+B+C+D*E".

Assume we use the Scoreboard Algorithm, access to both memory and register operands takes a single cycle, multiply takes 10 cycles, and add takes 2 cycles. Also, assume 3 adders, 1 multiplier unit, and that we can overlap as many load instructions as we like.

```
/* Option A */              /* Option B */
LD       F0, D              LD       F0, E
LD       F1, C              MULT.D   F0, D
LD       F2, B              ADD.D    F0, C
MULT.D   F0, E              ADD.D    F0,B
ADD.D    F1, F0             ADD.D    F0, A
ADD.D    F2, A
ADD.D    F1,F2
```

A: Code for "Option A" is faster

vation

oint code

gisters visible to programmer. Code
ructions in 360/91 ISA.

lier (higher clock frequency)

rallelism between branches < 2.

rd)

nprove clock frequency

ards

ling"

nents of these solutions and was

tel Core i5, i7)

vation

oint code

gisters visible to programmer. Code
ructions in 360/91 ISA.

Which code is faster? Option A or Option B?
Both compute "A+B+C+D*E".

Assume we use the Scoreboard Algorithm, access to both memory and register operands takes a single cycle, multiply takes 10 cycles, and add takes 2 cycles. Also, assume 3 adders, 1 multiplier unit, and that we can overlap as many load instructions as we like.

```
/* Option A */ ✓
LD        F0, D
LD        F1, C
LD        F2, B
MULT.D    F0, E
ADD.D     F1, F0
ADD.D     F2, A
ADD.D     F1,F2

(24 cycles)
```

```
/* Option B *
LD        F0, E
MULT.D    F0, D
ADD.D     F0, C
ADD.D     F0,B
ADD.D     F0, A

(32 cycles)
```

Option A	IS	RO	EC	WB
LD F0, D	1	2	3	4
LD F2, C	2	3	4	5
LD F4, B	3	4	5	6
MULT.D F0, E	4	5	15	16
ADD.D F2, F0	5	17	19	20
ADD.D F4, A	6	7	8	9
ADD.D F2, F4	7	21	23	**24**

Option B	IS	RO	EC	WB
LD F0, E	1	2	3	4
MULT.D F0,D	5	6	16	17
ADD.D F0,C	18	19	21	22
ADD.D F0,B	23	24	26	27
ADD.D F0,A	28	29	31	**32**

A: Code for "Option A" is faster ✓

Historical Motivation

- Design goal for System/360 model 91:
  - Sustain CPI as close as possible to 1 on floating-point code
- Constraints:
  1. Binary compatibility had to be maintained. Four registers visible to programmer. Code with many WAW hazards since required fewer instructions in 360/91 ISA.
  2. Long floating-point function unit latencies
  3. Separate pipelined floating point adder and multiplier (higher clock frequency)
  - **Today's Motivation:** Above + cache misses + parallelism between branches < 2.
- Design tradeoffs Tomasulo <u>considered</u>
  - Adding busy bits to register file (in-order scoreboard)
  - Adding "working registers" near function units to improve clock frequency
  - Additional function units to eliminate structural hazards
  - Impact of software techniques such as "loop unrolling"
- Tomasulo proposed a design that combined elements of these solutions and was significant improvement. Still used today (e.g. Intel Core i5, i7)

Tomasulo Algorithm

Tomasulo Algorithm

- Control & buffers distributed with Function Units (FU)
  - Buffers called reservation stations; contain pending operands

Tomasulo Algorithm

- Control & buffers distributed with Function Units (FU)
  - Buffers called reservation stations; contain pending operands
- Register numbers in instructions get replaced by “tags” each of which “points to” a reservation station (RS)
  - Effect of this is to “rename” registers
  - This “renaming” avoids WAR, WAW hazards which result from reusing the same register name even though no data is passed between the instructions involved.
  - Renaming allowed more reservation stations than registers
  - Results sent over a “Common Data Bus” which broadcasts results to all Function Units (Fus), and identifies producer “tag” not the consumer

Tomasulo Algorithm

- Control & buffers distributed with Function Units (FU)
  - Buffers called reservation stations; contain pending operands
- Register numbers in instructions get replaced by “tags” each of which “points to” a reservation station (RS)
  - Effect of this is to “rename” registers
  - This “renaming” avoids WAR, WAW hazards which result from reusing the same register name even though no data is passed between the instructions involved.
  - Renaming allowed more reservation stations than registers
  - Results sent over a “Common Data Bus” which broadcasts results to all Function Units (Fus), and identifies producer “tag” not the consumer
- Load and Stores treated as function units with RSs as well

Tomasulo Algorithm

- Control & buffers distributed with Function Units (FU)
  - Buffers called reservation stations; contain pending operands
- Register numbers in instructions get replaced by “tags” each of which “points to” a reservation station (RS)
  - Effect of this is to “rename” registers
  - This “renaming” avoids WAR, WAW hazards which result from reusing the same register name even though no data is passed between the instructions involved.
  - Renaming allowed more reservation stations than registers
  - Results sent over a “Common Data Bus” which broadcasts results to all Function Units (Fus), and identifies producer “tag” not the consumer
- Load and Stores treated as function units with RSs as well
- Tomasulo was working on floating-point hardware unit (similar to example in next few slides). Idea extends to integer instructions as well.

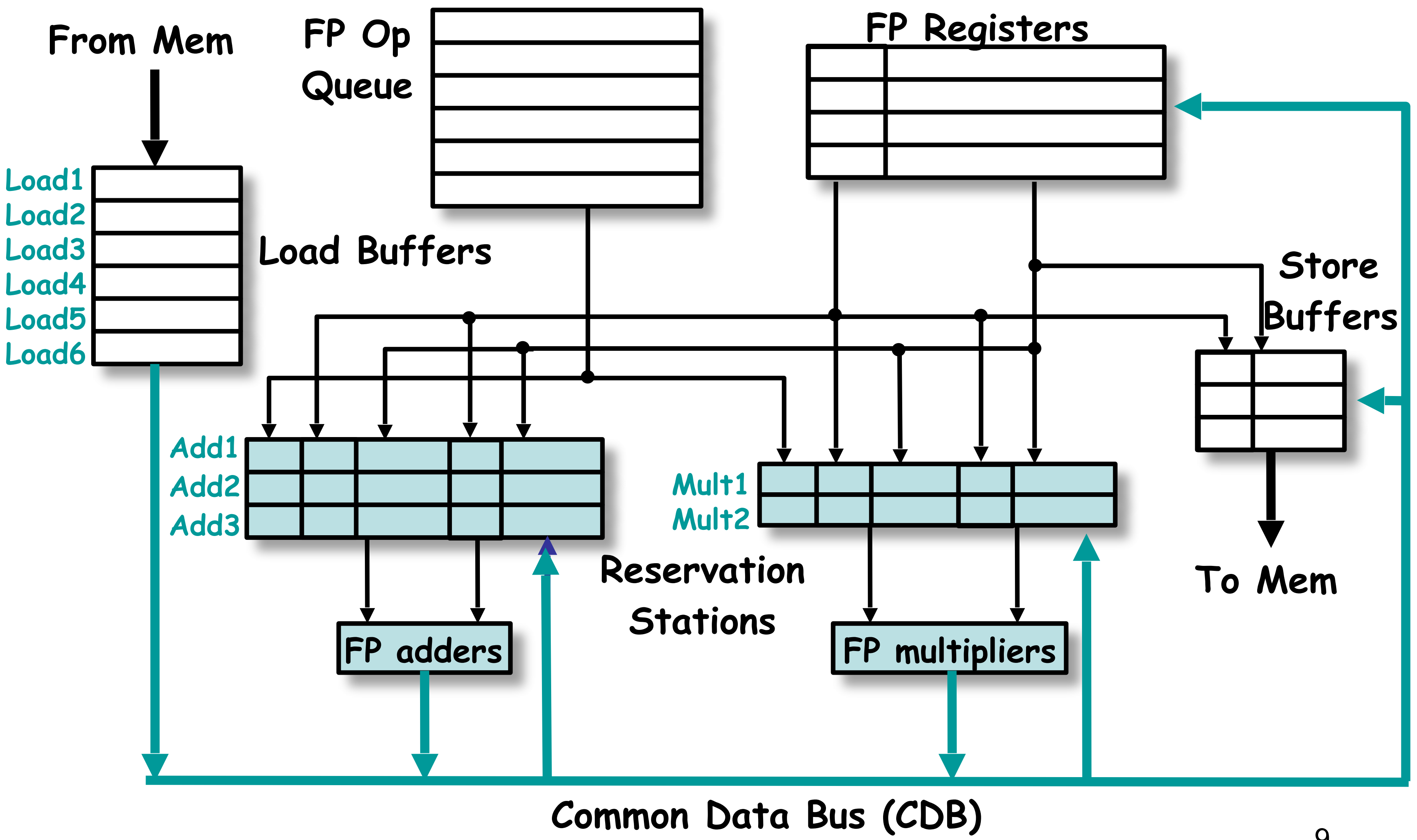
Three Stages of Tomasulo Algorithm

1. Issue—Get instruction from Instruction (FP Op) Queue
   (called "dispatch" in superscalar processors)
   If reservation station free (no structural hazard),
   (a) lookup source operand registers in "register result status" table while allocating reservation station.
   (b) update "register result status" table entry of destination register with reseveration station (this renames destination register)
2. Execute—Operate on operands
   (called "issue" in superscalar processors)
   When both operands ready then execute;
   if not ready, watch Common Data Bus for result
3. Write result—Finish execution
   Write on Common Data Bus to all units waiting for result;
   mark reservation station available

- Normal data bus = data + destination address ("go to" bus)

- Common data bus = data + source tag ("come from" bus)
  - 64 bits of data + 4 bits of Reservation Station source address (tag)

Tomasulo Algorithm



Tomasulo Algorithm Components

Reservation Station:

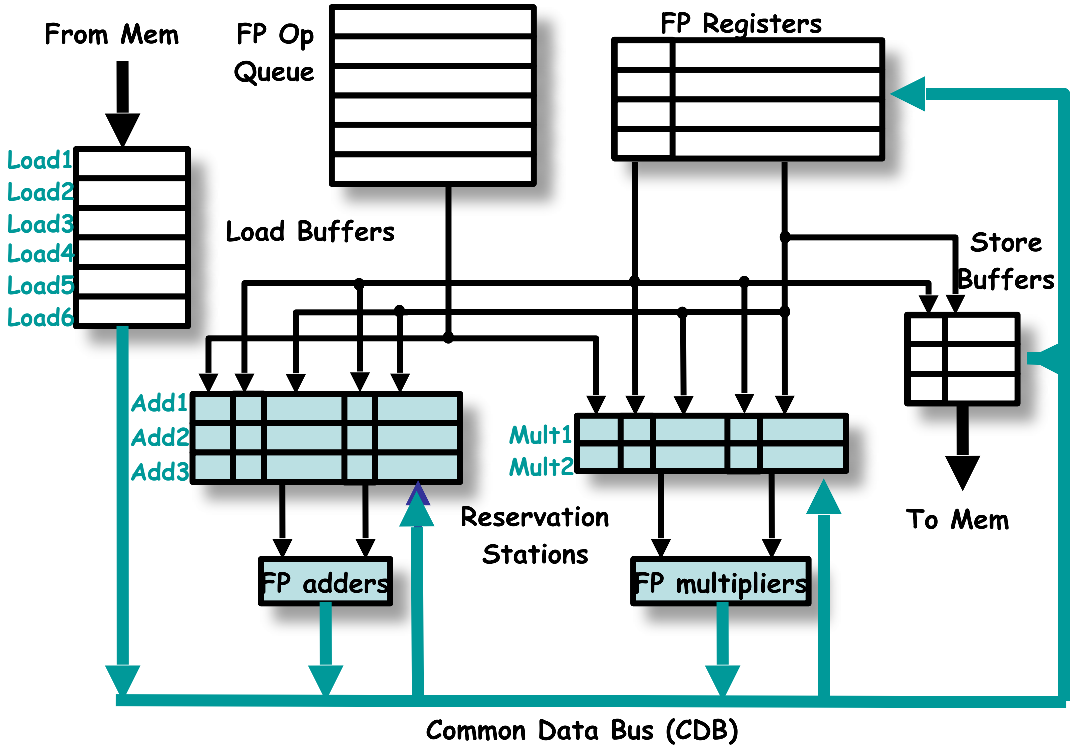
- Busy: Indicates whether reservation station is busy
- Op: Operation to perform in FU (e.g., “add” or “subtract”)
- Vj, Vk: **Value** of Source operands
- Qj, Qk: **Tags**: Identify reservation station producing Vj, Vk
  - When Qj=0 (no tag), Vj holds valid data
  - When Qk=0 (no tag), Vk holds valid data

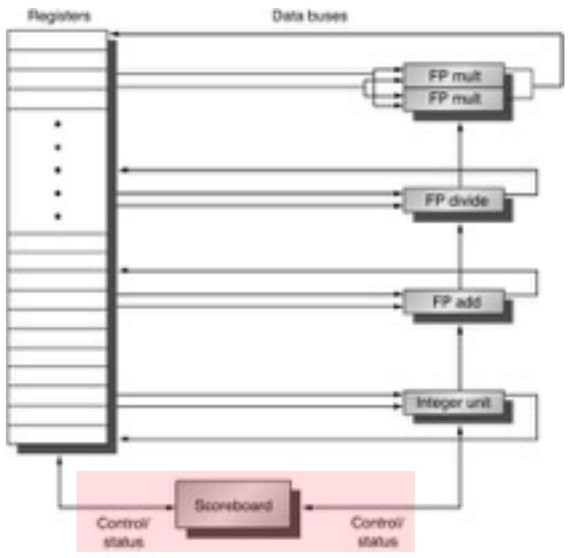
- Address: Used to hold effective address for memory (initially set to immediate offset)

Register File (each register has):

- Qi—Indicates which reservation station will write register, if such a reservation station exists. Zero if no pending instructions write register.

Tomasulo vs. Scoreboard





Tomasulo:
Decentralized Control
Reservation Stations/Renaming

Scoreboard:
Centralized Control
WAR, WAW hazards

Assumptions for example

- Execution Latency:
  - 2 clocks for floating point add, subtract;
  - 10 clocks for floating point multiply
  - 40 clocks for floating point divide
  - Load/Store: 2 cycles
    - 1st cycle effective address
    - 2nd cycle access memory
- Pipelined function units (start one operation per cycle)
- 3 floating-point add/subtract reservation stations
- 2 floating-point multiple reservation stations
- Read value into reservation station same cycle as writeback of value to register file (even if the reservation station was “allocated” the same cycle); however, if that means all operands ready, still have to wait until following cycle to “begin execution”

Tomasulo Example Cycle 1

Instruction stream

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1		
L.D	F2	45+	R3			
MULT.D	F0	F2	F4			
SUB.D	F8	F6	F2			
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	No	
Load3	No	

3 Load/Buffers

Reservation Stations:

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

FU count down

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

Register result status:

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

Clock cycle counter

Tomasulo Example Cycle 2

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Begin*	*Write Result*
L.D	F6	34+	R2	1	2	
L.D	F2	45+	R3	2		
MULT.D	F0	F2	F4			
SUB.D	F8	F6	F2			
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

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

Reservation Stations:

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

Register result status:

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

Tomasulo Example Cycle 3

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	
L.D	F2	45+	R3	2	3	
MULT.D	F0	F2	F4	3		
SUB.D	F8	F6	F2			
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

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

Reservation Stations:

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

Shorthand for contents of register F4

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...
3	Qi	Mult1	Load2		Load1				

Tomasulo Example Cycle 3

Instruction status:

Instruction	j	k		*Issue*	*Exec Begin*	*Write Result*
L.D	F6	34+	R2	1	2	
L.D	F2	45+	R3	2	3	
MULT.D	F0	F2	F4	3		
SUB.D	F8	F6	F2			
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

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

Reservation Stations:

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

Shorthand for contents of register F4

Register result status:

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

- **Note: registers names are removed ("renamed") in Reservation Stations**
- **MUL.D issued; Load1 completing - what is waiting for it?**

Tomasulo Example Cycle 4

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	
MULT.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4		
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vi	S2 Vk	RS Qi	RS Qk
	Add1	Yes	SUB.D	M(A1)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D		R(F4)	Load2	
	Mult2	No					

Shorthand for contents of memory at address A1, where A1 is effective addr. of first load (34+R2)

Register result status:

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

Tomasulo Example Cycle 4

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Begin*	*Write Result*
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	
MULT.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4		
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vi*	*S2 Vk*	*RS Qi*	*RS Qk*
	Add1	Yes	SUB.D	M(A1)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D		R(F4)	Load2	
	Mult2	No					

Shorthand for contents of memory at address A1, where A1 is effective addr. of first load (34+R2)

Register result status:

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

- **Issue SUB.D; Load2 completing; <u>what is waiting for Load2?</u>**

Tomasulo Example Cycle 4

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	$\boxed{4}$
L.D	F2	45+	R3	2	3	
MULT.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	$\boxed{4}$		
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vi	S2 Vk	RS Qi	RS Qk
	Add1	Yes	SUB.D	M(A1)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D		R(F4)	Load2	
	Mult2	No					

Shorthand for contents of memory at address A1, where A1 is effective addr. of first load (34+R2)

Register result status:

	F0	F2	F4	F6	F8	F10	F12	...
Qi	Mult1	Load2		$\boxed{\text{M(A1)}}$	Add1			

A: Nothing
B: ADD.D
C: Load2,Add1, F2
D: Mult1,Add1, F2
E: Load2,F2,Mult1

- Issue SUB.D; Load2 completing; <u>what is waiting for Load2?</u>

Tomasulo Example Cycle 4

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	$\boxed{4}$
L.D	F2	45+	R3	2	3	
MULT.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	$\boxed{4}$		
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	Yes	SUB.D	M(A1)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D		R(F4)	Load2	
	Mult2	No					

Shorthand for contents of memory at address A1, where A1 is effective addr. of first load (34+R2)

[Register result st]atus:

	F0	F2	F4	F6	F8	F10	F12	...
Qi	Mult1	Load2		$\boxed{\text{M(A1)}}$	Add1			

A: Nothing
B: ADD.D
C: Load2,Add1, F2
D: Mult1,Add1, F2 ✓
E: Load2,F2,Mult1

- Issue SUB.D; Load2 completing; <u>what is waiting for Load2?</u>

Tomasulo Example Cycle 4

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	
MULT.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4		
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vi	S2 Vk	RS Qi	RS Qk
	Add1	Yes	SUB.D	M(A1)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D		R(F4)	Load2	
	Mult2	No					

Shorthand for contents of memory at address A1, where A1 is effective addr. of first load (34+R2)

Register result status:

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

- **Issue SUB.D; Load2 completing; <u>what is waiting for Load2?</u>**

Tomasulo Example Cycle 5

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4		
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

				S1	S2	RS	RS
Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*
2	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
10	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

Tomasulo Example Cycle 5

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3		
SUB.D	F8	F6	F2	4		
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
2	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
10	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

- **Load2 writes to CDB; Timer starts down for Add1, Mult1 (they "begin execution" following cycle -- clock cycle 6)**

Tomasulo Example Cycle 6

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	$\boxed{6}$	
SUB.D	F8	F6	F2	4	$\boxed{6}$	
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
1	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
9	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...
6	Qi	Mult1	M(A2)		M(A1)	Add1	Mult2		

Tomasulo Example Cycle 6

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	$\boxed{6}$	
SUB.D	F8	F6	F2	4	$\boxed{6}$	
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

				S1	S2	RS	RS
Time	Name	Busy	Op	Vj	Vk	Qj	Qk
1	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
9	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...
6	Qi	Mult1	M(A2)		M(A1)	Add1	Mult2		

- **Issue ADD.D here despite name dependency on F6?**

A: Yes
B: No
C: Not sure

Tomasulo Example Cycle

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	$\boxed{6}$	
SUB.D	F8	F6	F2	4	$\boxed{6}$	
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
1	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
9	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...
6	Qi	Mult1	M(A2)		M(A1)	Add1	Mult2		

- **Issue ADD.D here despite name dependency on F6?**

Tomasulo Example Cycle

[✓ A: Yes]
B: No
C: Not sure

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	$\boxed{6}$	
SUB.D	F8	F6	F2	4	$\boxed{6}$	
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
1	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	Yes	ADD.D		M(A2)		
	Add3	Add1					
9	Mult1						
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...
6	Qi	Mult1	M(A2)		$\boxed{\text{Add2}}$	Add1	Mult2		

- **Issue ADD.D here despite name dependency on F6?**

Tomasulo Example Cycle 7

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	Yes	ADD.D		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

Tomasulo Example Cycle 7

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	Yes	ADD.D		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

- **Add1 (SUB.D) completing; <u>what is waiting for it?</u>**

Tomasulo Example Cycle 7

Instruction status:

Instruction		j	k	*Issue*	*Exec Begin*	*Write Result*
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
0	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	Yes	ADD.D		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

A: Mult2,F0
B: Add2,F8
C: Load1,Add1,F10
D: F6
E: Nothing

Register result status:

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

- **Add1 (SUB.D) completing; what is waiting for it?**

Tomasulo Example Cycle 7

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	Yes	ADD.D		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

A: Mult2,F0
B: Add2,F8 ✓
C: Load1,Add1,F10
D: F6
E: Nothing

Register result status:

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

- **Add1 (SUB.D) completing; <u>what is waiting for it?</u>**

Tomasulo Example Cycle 7

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Yes	SUB.D	M(A1)	M(A2)		
	Add2	Yes	ADD.D		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

- **Add1 (SUB.D) completing; <u>what is waiting for it?</u>**

Tomasulo Example Cycle 8

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	$\boxed{8}$
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
2	Add2	Yes	ADD.D	$\boxed{\text{(M-M)}}$	M(A2)		
	Add3	No					
7	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...
8	Qi	Mult1	M(A2)		Add2	$\boxed{\text{(M-M)}}$	Mult2		

Tomasulo Example Cycle 9

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Begin*	*Write Result*
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	$\boxed{9}$	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
1	Add2	Yes	ADD.D	(M-M)	M(A2)		
	Add3	No					
6	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

Tomasulo Example Cycle 10

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
0	Add2	Yes	ADD.D	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock
10

	F0	F2	F4	F6	F8	F10	F12	...
Qi	Mult1	M(A2)		Add2	(M-M)	Mult2		

Tomasulo Example Cycle 10

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
0	Add2	Yes	ADD.D	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

- **Add2 (ADD.D) completing; <u>what is waiting for it?</u>**

Tomasulo Example Cycle 10

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
0	Add2	Yes	ADD.D	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

A: Nothing
B: Add1
C: Mult1
D: F2
E: F6

Register result status:

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

- **Add2 (ADD.D) completing: <u>what is waiting for it?</u>**

Tomasulo Example Cycle 10

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
0	Add2	Yes	ADD.D	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

A: Nothing
B: Add1
C: Mult1
D: F2
E: F6 ✓

Register result status:

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

- **Add2 (ADD.D) completing: <u>what is waiting for it?</u>**

Tomasulo Example Cycle 10

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Begin*	*Write Result*
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

				S1	*S2*	*RS*	*RS*
Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*
	Add1	No					
0	Add2	Yes	ADD.D	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

- ***Add2 (ADD.D) completing; <u>what is waiting for it?</u>***

Tomasulo Example Cycle 11

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...
11	Qi	Mult1	M(A2)			(M-M)	Mult2		

Tomasulo Example Cycle 11

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Begin*	*Write Result*
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

Register result status:

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

- <u>***Write result of ADD.D here?***</u>

Tomasulo Example Cycle 11

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...
11	Qi	Mult1	M(A2)			(M-M)	Mult2		

- **Write result of ADD.D here?**

Tomasulo Example Cycle 11

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

A: Yes, very sure ✓
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...
11	Qi	Mult1	M(A2)			(M-M)	Mult2		

- **Write result of ADD.D here?**

Tomasulo Example Cycle 12

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

Tomasulo Example Cycle 13

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

Tomasulo Example Cycle 14

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

Tomasulo Example Cycle 15

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

Tomasulo Example Cycle 15

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

- **Mult1 (MUL.D) completing; what is waiting for it?**

Tomasulo Example Cycle 15

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

A: Nothing
B: F0
C: Mult1, F0
D: F10
E: Mult2,F0

Register result status:

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

- **Mult1 (MUL.D) completing; what is waiting for it?**

Tomasulo Example Cycle 15

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Begin*	*Write Result*
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

A: Nothing
B: F0
C: Mult1, F0
D: F10
E: Mult2,F0 ✓

Register result status:

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

- **Mult1 (MUL.D) completing; what is waiting for it?**

Tomasulo Example Cycle 15

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MUL.D	M(A2)	R(F4)		
	Mult2	Yes	DIV.D		M(A1)	Mult1	

Register result status:

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

- **Mult1 (MUL.D) completing; what is waiting for it?**

Tomasulo Example Cycle 16

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	16
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5	16	
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

Register result status:

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

Tomasulo Example Cycle 16

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	16
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5	16	
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	Mult2	Yes	DIV.D	$\boxed{\text{M*F4}}$	M(A1)		

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...
16	Qi	$\boxed{\text{M*F4}}$	M(A2)		(M-M+M	(M-M)	Mult2		

- **Just waiting for Mult2 (DIVD) to complete**

Faster than light computation (skip a couple of cycles)

Tomasulo Example Cycle 55

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	16
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5	16	
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

Register result status:

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

Tomasulo Example Cycle 56

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	16
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5	16	
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

Register result status:

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

Tomasulo Example Cycle 56

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	16
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5	16	
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

Register result status:

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

- **Mult2 (DIVD) is completing; what is waiting for it?**

Tomasulo Example Cycle 57

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	16
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5	16	57
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

Register result status:

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

Tomasulo Example Cycle 57

Instruction status:

Instruction		j	k	Issue	Exec Begin	Write Result
L.D	F6	34+	R2	1	2	4
L.D	F2	45+	R3	2	3	5
MUL.D	F0	F2	F4	3	6	16
SUB.D	F8	F6	F2	4	6	8
DIV.D	F10	F0	F6	5	16	57
ADD.D	F6	F8	F2	6	9	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

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

Register result status:

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

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

Tomasulo Drawbacks

Tomasulo Drawbacks

- Complexity
  - delays of 360/91, MIPS 10000, Alpha 21264, IBM PPC 620 (in CA:AQA 2/e, but not in silicon!)

Tomasulo Drawbacks

- Complexity
  - delays of 360/91, MIPS 10000, Alpha 21264, IBM PPC 620 (in CA:AQA 2/e, but not in silicon!)
- Many associative comparisons (CDB) at high speed

Tomasulo Drawbacks

- Complexity
  - delays of 360/91, MIPS 10000, Alpha 21264, IBM PPC 620 (in CA:AQA 2/e, but not in silicon!)
- Many associative comparisons (CDB) at high speed
- Performance limited by Common Data Bus
  - Each CDB must go to multiple functional units ⇒high capacitance, high wiring density
  - Number of functional units that can complete per cycle limited to one!
    - Multiple CDBs ⇒ more FU logic for parallel assoc. stores

Tomasulo Drawbacks

- Complexity
  - delays of 360/91, MIPS 10000, Alpha 21264, IBM PPC 620 (in CA:AQA 2/e, but not in silicon!)
- Many associative comparisons (CDB) at high speed
- Performance limited by Common Data Bus
  - Each CDB must go to multiple functional units ⇒high capacitance, high wiring density
  - Number of functional units that can complete per cycle limited to one!
    - Multiple CDBs ⇒ more FU logic for parallel assoc. stores
- Non-precise interrupts!
  - We will see how to solve this problem later

Register Renaming: The Idea

Each new value gets a new name

Original code:

```
LD        R1, 0(R2)
DADD      R3,R1,R3
DADDUI    R2,R2,#8
LD        R1, 0(R2)
```

Rewrite using T1, T2, etc… for each <u>new value</u>:

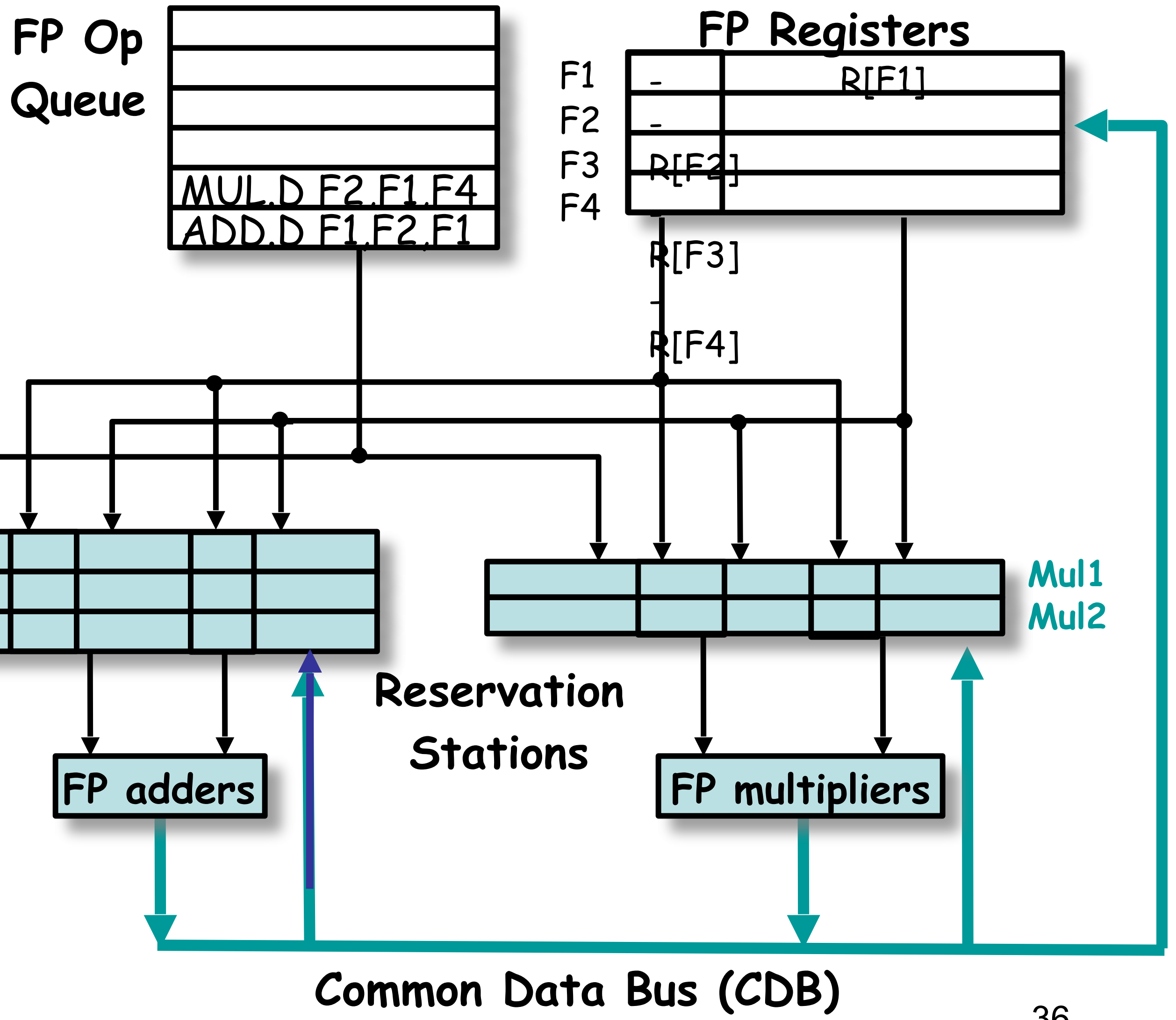
```
LD        T1, 0(R2)     ; R1 renamed to T1
DADD      T2,T1,R3      ; R3 renamed to T2
DADDUI    T3,R2,#8      ; R2 renamed to T3
LD        T4, 0(T3)     ; R1 renamed to T4
```

Tomasulo Algorithm, Renaming Implemented

Program:

ADD.D F1,F2,F1

MUL.D F2,F1,F4

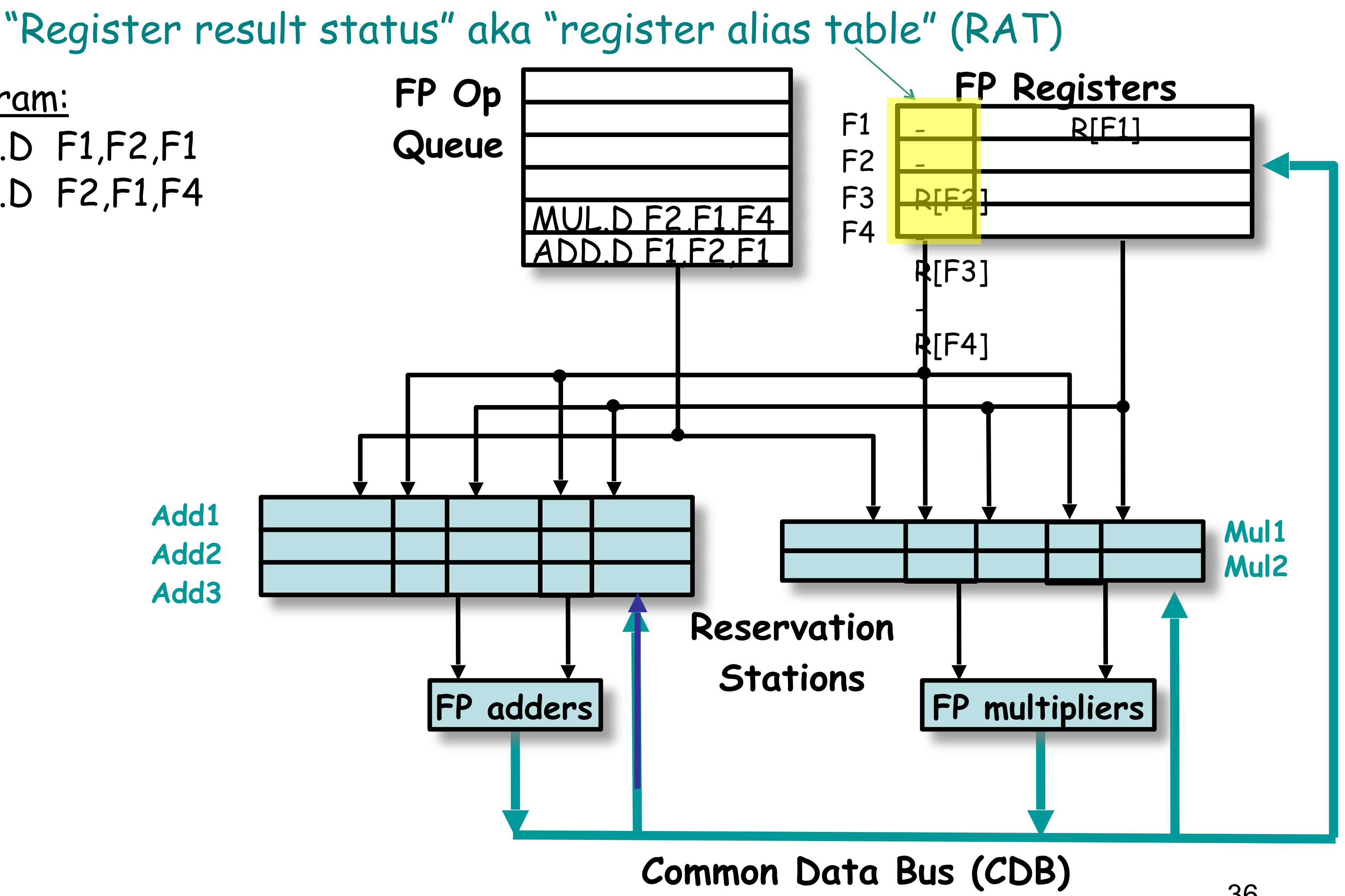


Cycle 1

Tomasulo Algorithm, Renaming Implemented

Program:

ADD.D F1,F2,F1
MUL.D F2,F1,F4



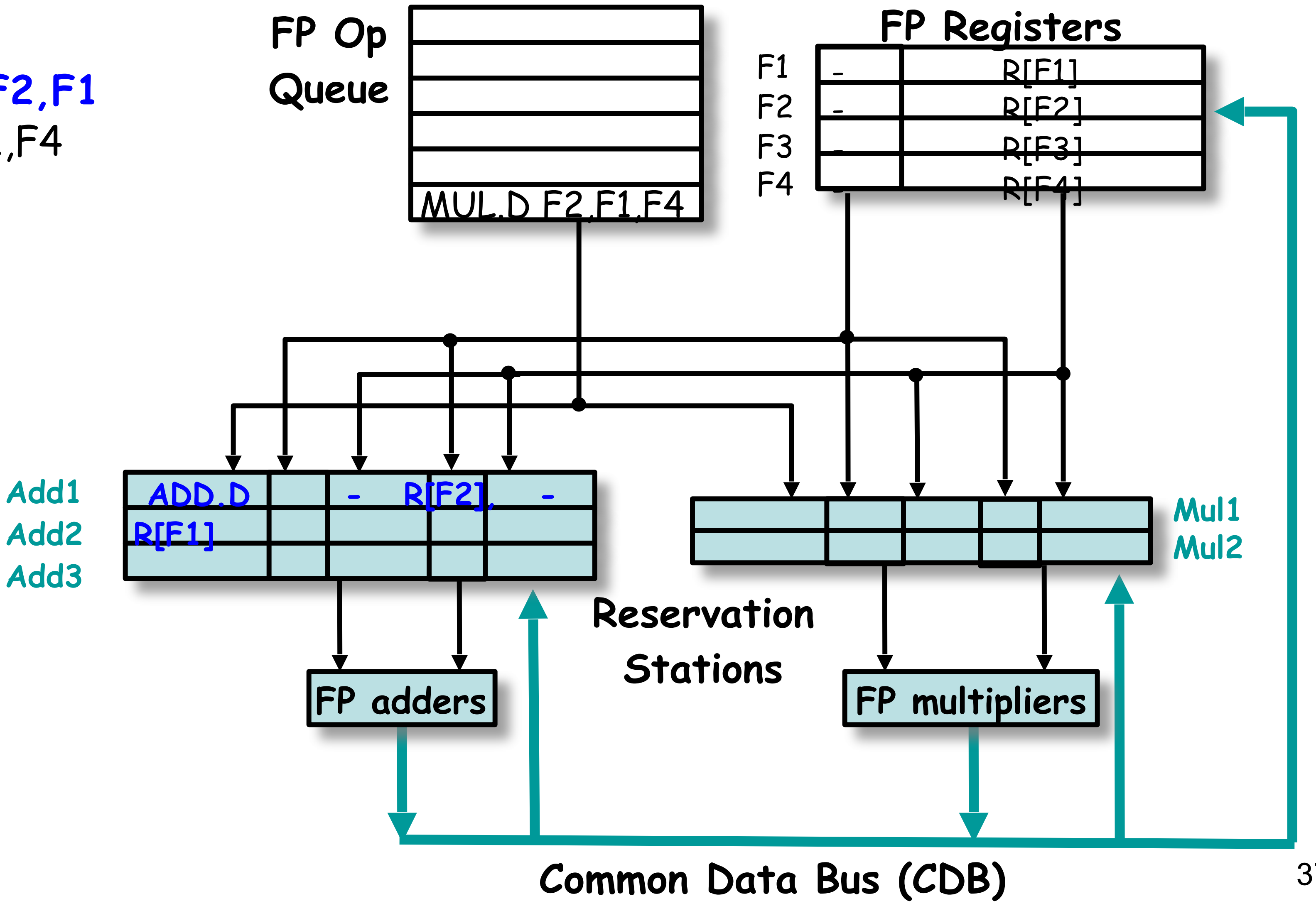
Cycle 1

Tomasulo Algorithm, Renaming Implemented

Program:

ADD.D F1,F2,F1

MUL.D F2,F1,F4



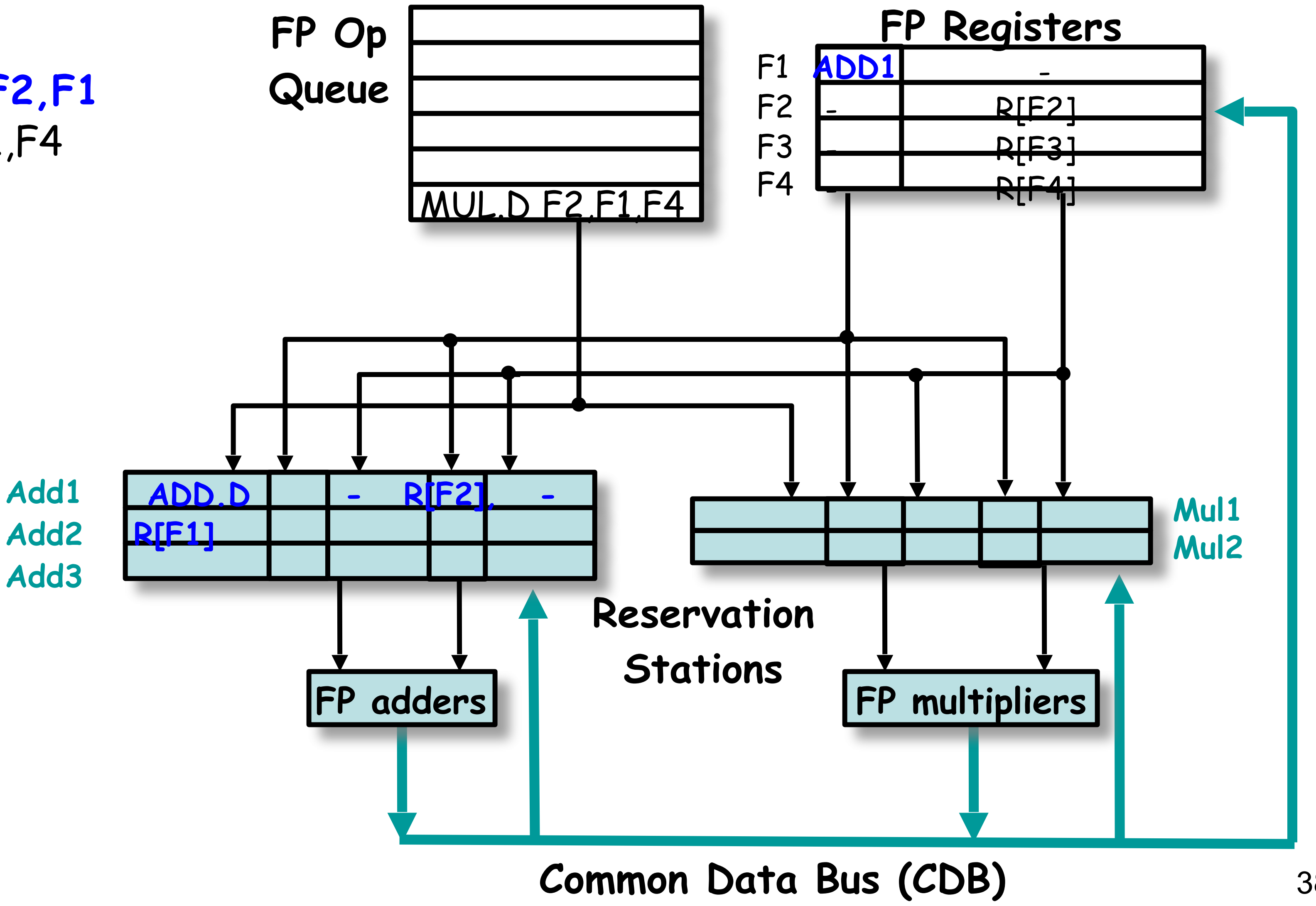
Cycle 2a: ADD.D reads R[F2] & R[F1] values from FP Registers

Tomasulo Algorithm, Renaming Implemented

Program:

ADD.D F1,F2,F1

MUL.D F2,F1,F4



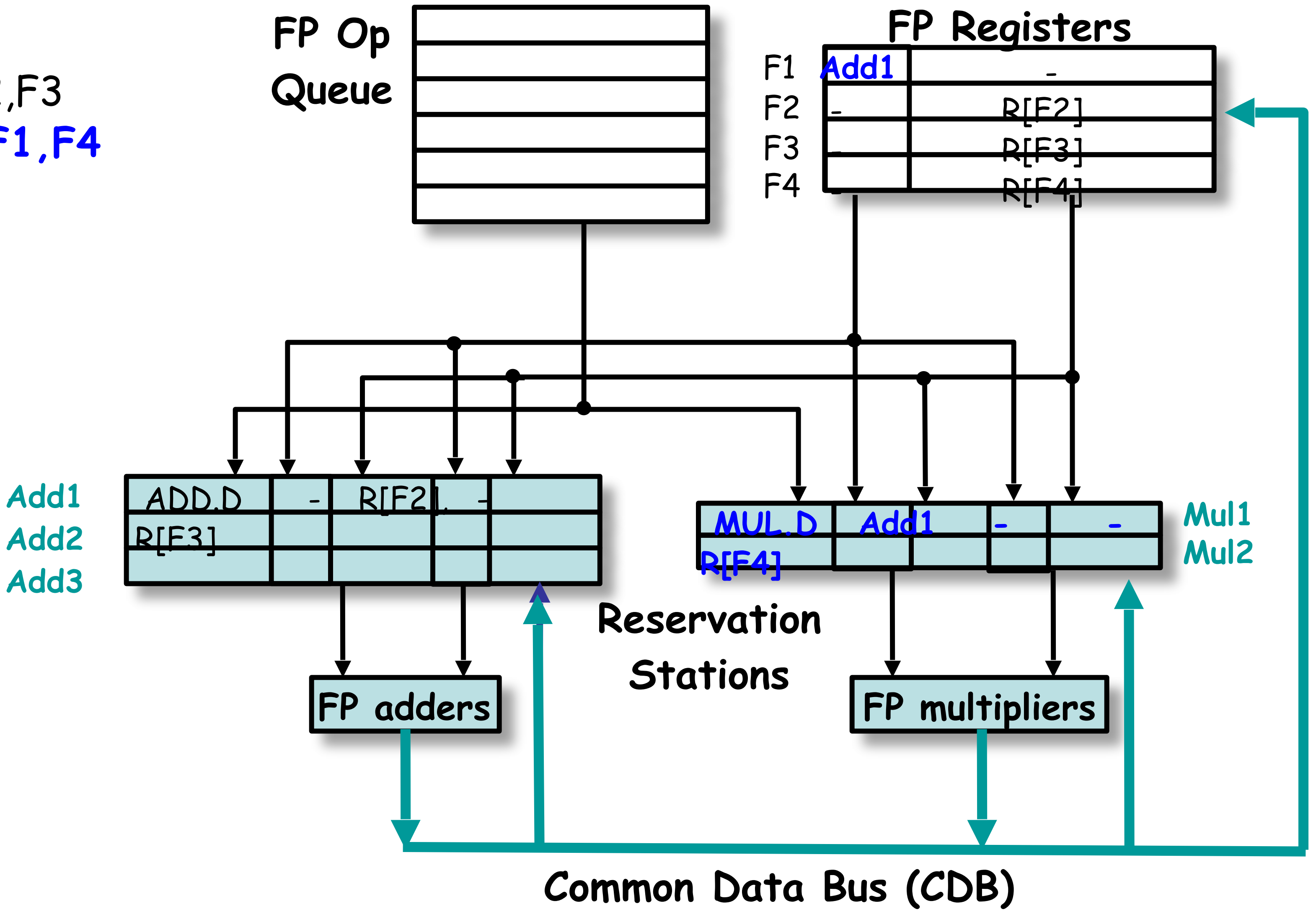
Cycle 2b: F1 "renamed" to "ADD1" (in "register result status")

Tomasulo Algorithm, Renaming Implemented

Program:

ADD.D F1,F2,F3

MUL.D F2,F1,F4



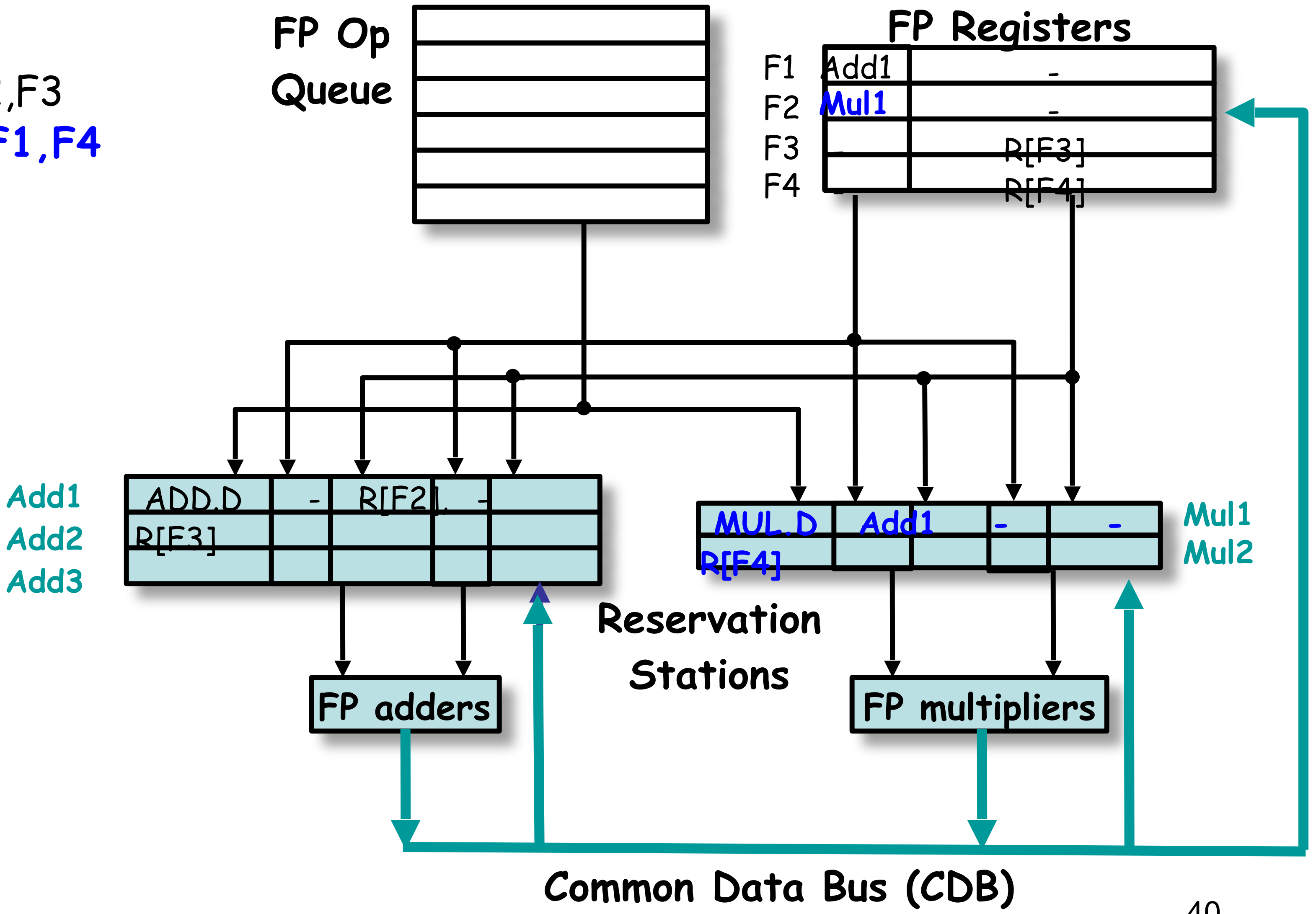
Cycle 3a: MUL.D gets tag “Add1” instead of value “R[F1]”

Tomasulo Algorithm, Renaming Implemented

Program:

ADD.D F1,F2,F3

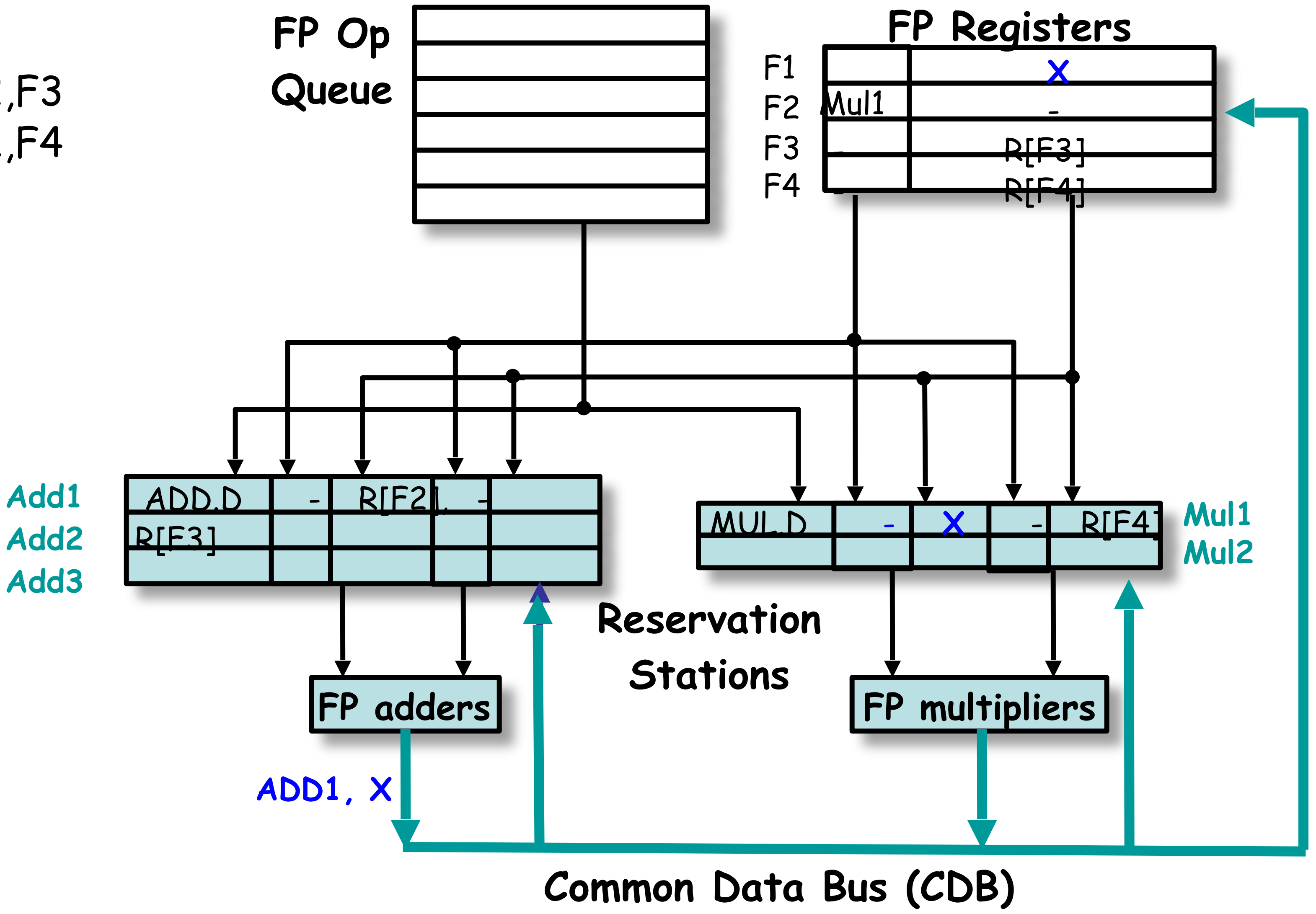
MUL.D F2,F1,F4



Cycle 3b: F2 renamed to "Mul1"

Tomasulo Algorithm, Renaming Implemented

Program:
ADD.D F1,F2,F3
MUL.D F2,F1,F4



Cycle 4: result value "X" and name "ADD1" broadcast on CDB