Lecture 12: Dynamic Scheduling w/ Recovery

some slides were adapted from Prof. John Kubiatowicz's "Reorder Buffers and Eplicit Register Renaming

Tomasulo Summary

- prevents register as bottleneck
- avoids/removes WAR, WAW hazards
- lasting contributions
  - ✓ dynamic scheduling
  - ✓ register renaming (in what way does the register name change?)
  - ✓ load/store disambiguation

Objectives

- revising the Tomasulo algorithm to support speculation/in-order completion

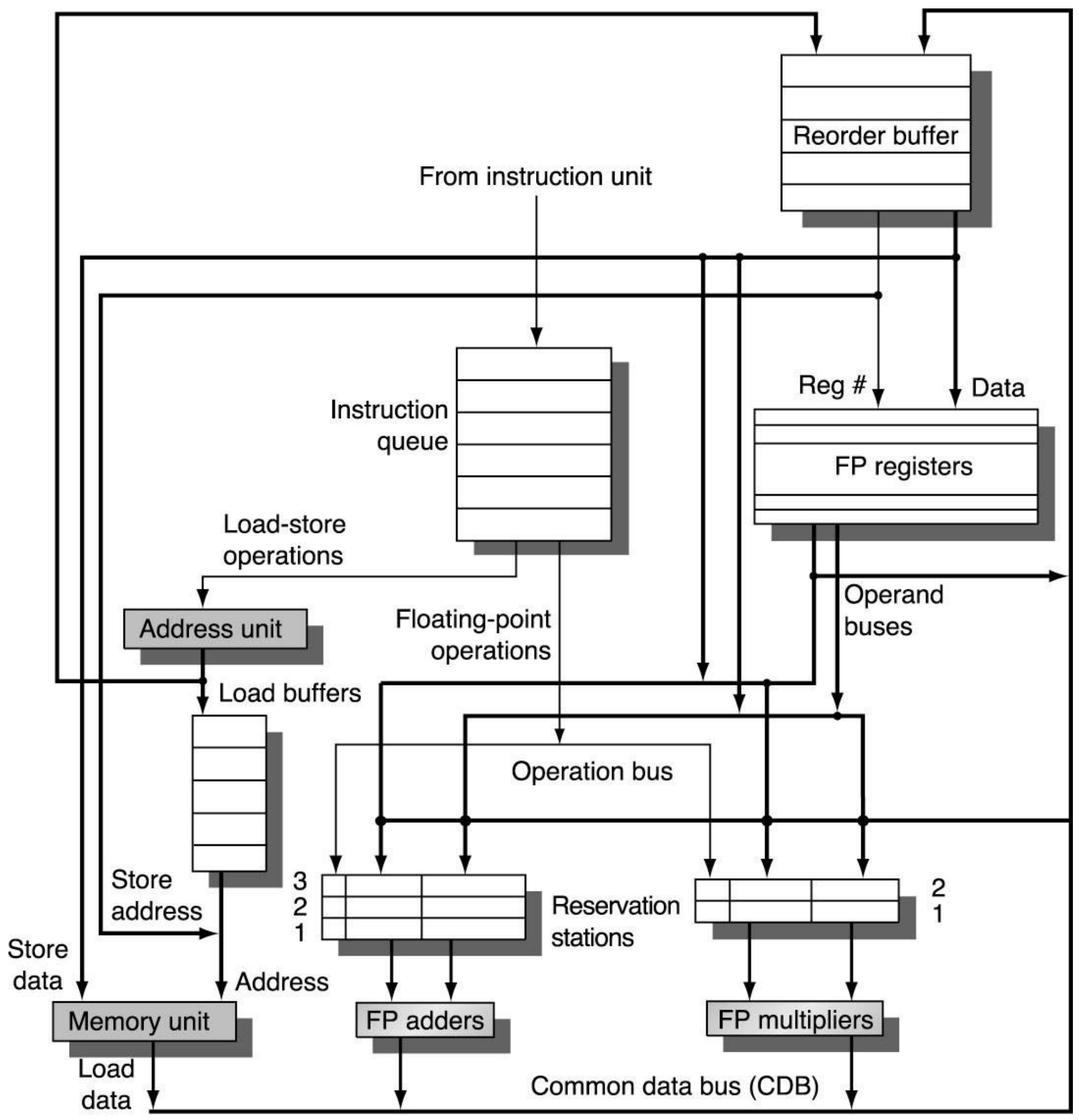
Support for Speculation

- speculation:
  - ✓ allow an instruction to issue that is dependent on branch without any consequences (including exceptions) if branch is predicted incorrectly (“HW undo”)
- Tomasulo:
  - ✓ when instruction no longer speculative, write results (instruction commit or instruction retire)
  - ✓ execute out-of-order but commit in order
  - ✓ requires some kind of intermediate storage

Hardware Speculative Execution

- need HW buffer for results of uncommitted instructions: reorder buffer
  - ✓ reorder buffer can be operand source
  - ✓ once operand commits, result is found in register
  - ✓ 3 fields: instr. type, destination, & value
  - ✓ use reorder buffer number instead of reservation station as “name” of result
  - ✓ instructions commit in order
  - ✓ as a result, its easy to undo speculated instructions on mispredicted branches or on exceptions

Speculative Tomasulo



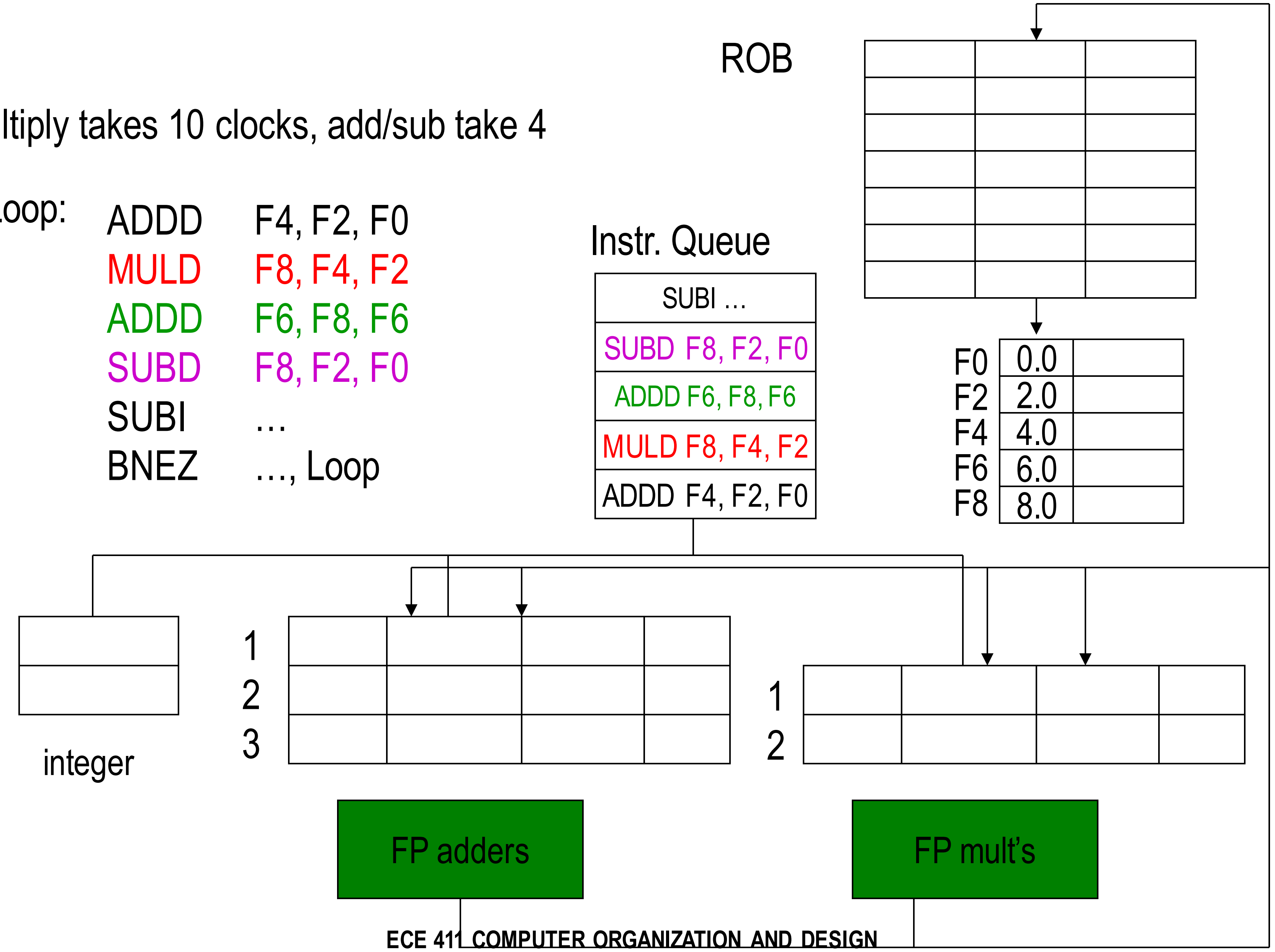
Four Steps of Speculative Tomasulo

- issue—get instruction from FP Op Queue
  - ✓ if RS and ROB slots are free, issue instr & send operands & ROB # for destination.
- execution—operate on operands (EX)
  - ✓ when both operands ready then execute; if not ready, watch CDB for result; when both in RS, execute
- write result—finish execution (WB)
  - ✓ write on CDB to all awaiting FUs & ROB; mark RS available.
- commit—update register w/ reorder result
  - ✓ when instr. at head of ROB & result present, update register with result (or store to memory) and remove instr from ROB.

Tomasulo – cycle 0

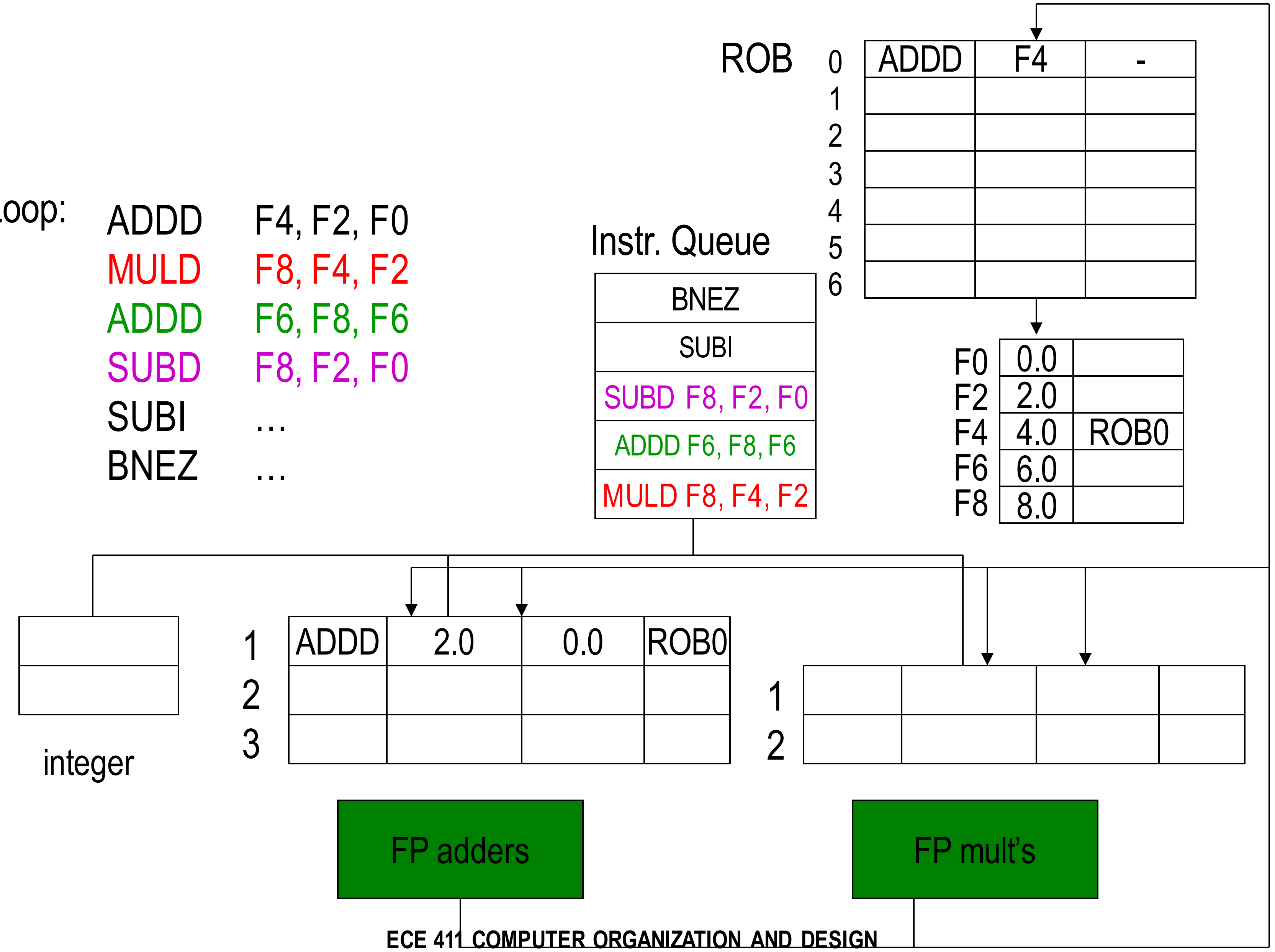
multiply takes 10 clocks, add/sub take 4

Loop:	ADDD	F4, F2, F0
	MULD	F8, F4, F2
	ADDD	F6, F8, F6
	SUBD	F8, F2, F0
	SUBI	…
	BNEZ	…, Loop



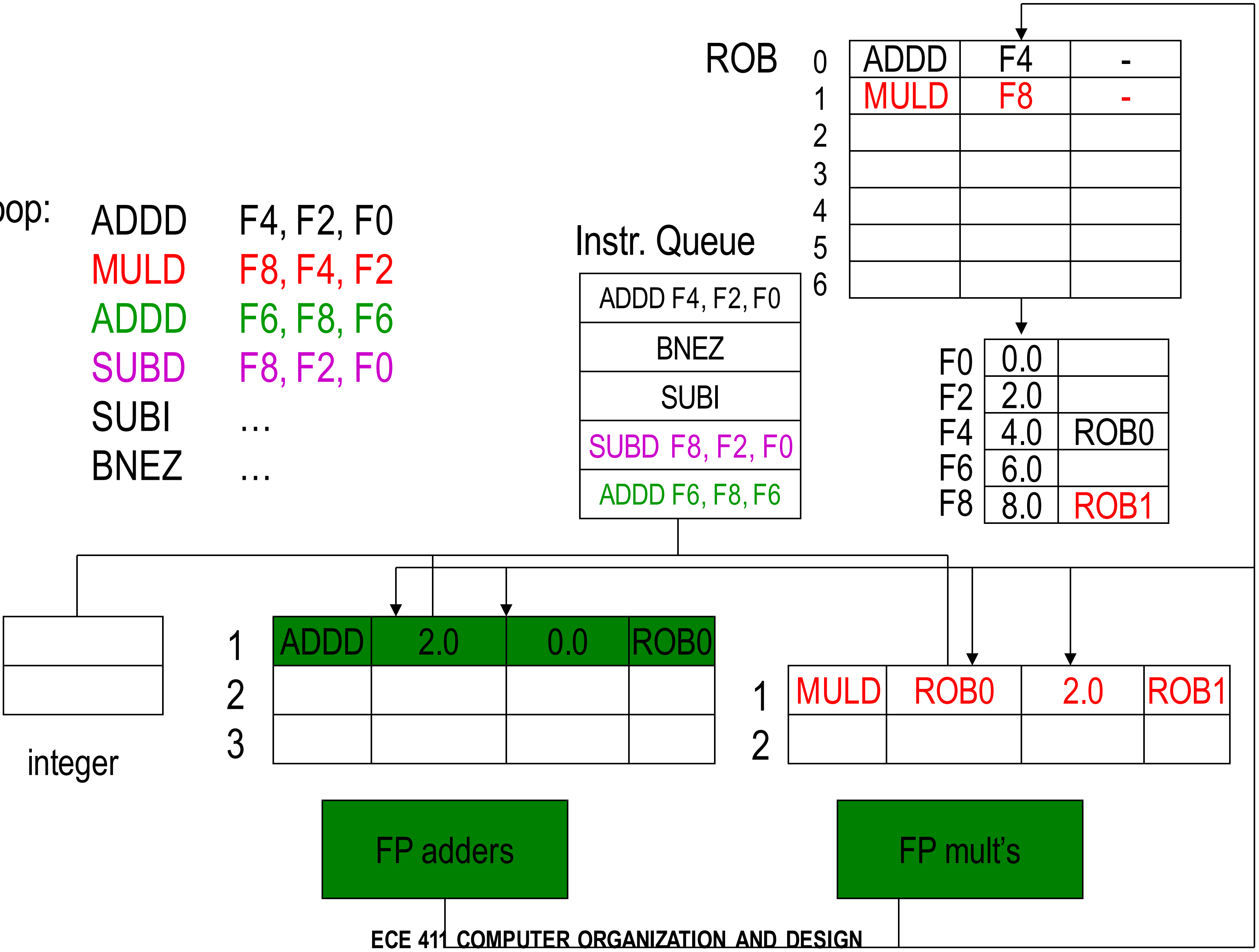
Tomasulo – cycle 1

Loop:	ADDD	F4, F2, F0
	MULD	F8, F4, F2
	ADDD	F6, F8, F6
	SUBD	F8, F2, F0
	SUBI	…
	BNEZ	…



Tomasulo – cycle 2

Loop:		
	ADDD	F4, F2, F0
	MULD	F8, F4, F2
	ADDD	F6, F8, F6
	SUBD	F8, F2, F0
	SUBI	…
	BNEZ	…



ROB

0	ADDD	F4	-
1	MULD	F8	-
2			
3			
4			
5			
6			

Instr. Queue

ADDD F4, F2, F0
BNEZ
SUBI
SUBD F8, F2, F0
ADDD F6, F8, F6

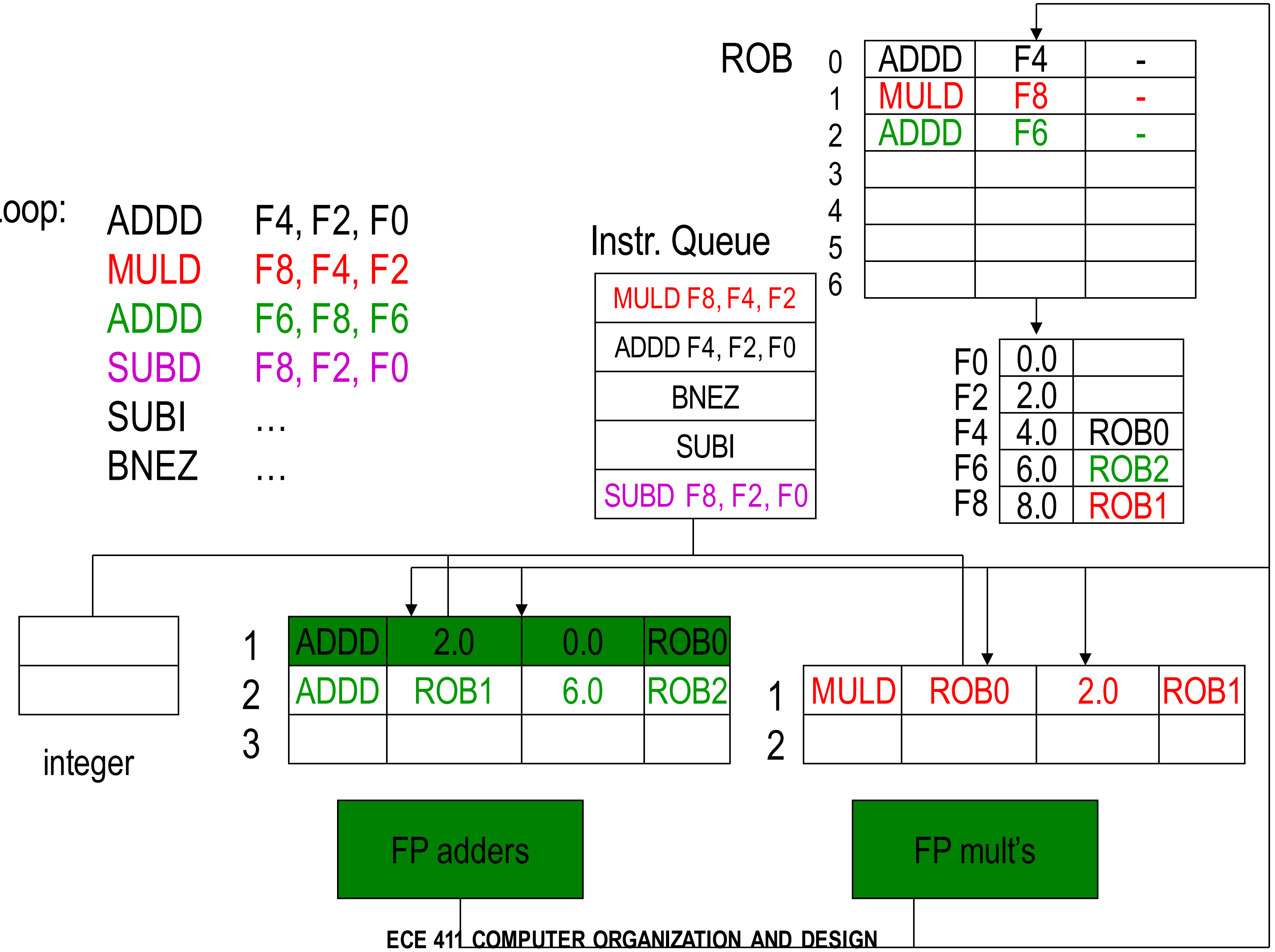
F0	0.0	
F2	2.0	
F4	4.0	ROB0
F6	6.0	
F8	8.0	ROB1

1	ADDD	2.0	0.0	ROB0
2				
3				

1	MULD	ROB0	2.0	ROB1
2				

Tomasulo – cycle 3

```
Loop:  ADDD   F4, F2, F0
       MULD   F8, F4, F2
       ADDD   F6, F8, F6
       SUBD   F8, F2, F0
       SUBI   …
       BNEZ   …
```



ROB

0	ADDD	F4	-
1	MULD	F8	-
2	ADDD	F6	-
3			
4			
5			
6			

Instr. Queue

MULD F8, F4, F2
ADDD F4, F2, F0
BNEZ
SUBI
SUBD F8, F2, F0

F0	0.0	
F2	2.0	
F4	4.0	ROB0
F6	6.0	ROB2
F8	8.0	ROB1

FP adders reservation stations

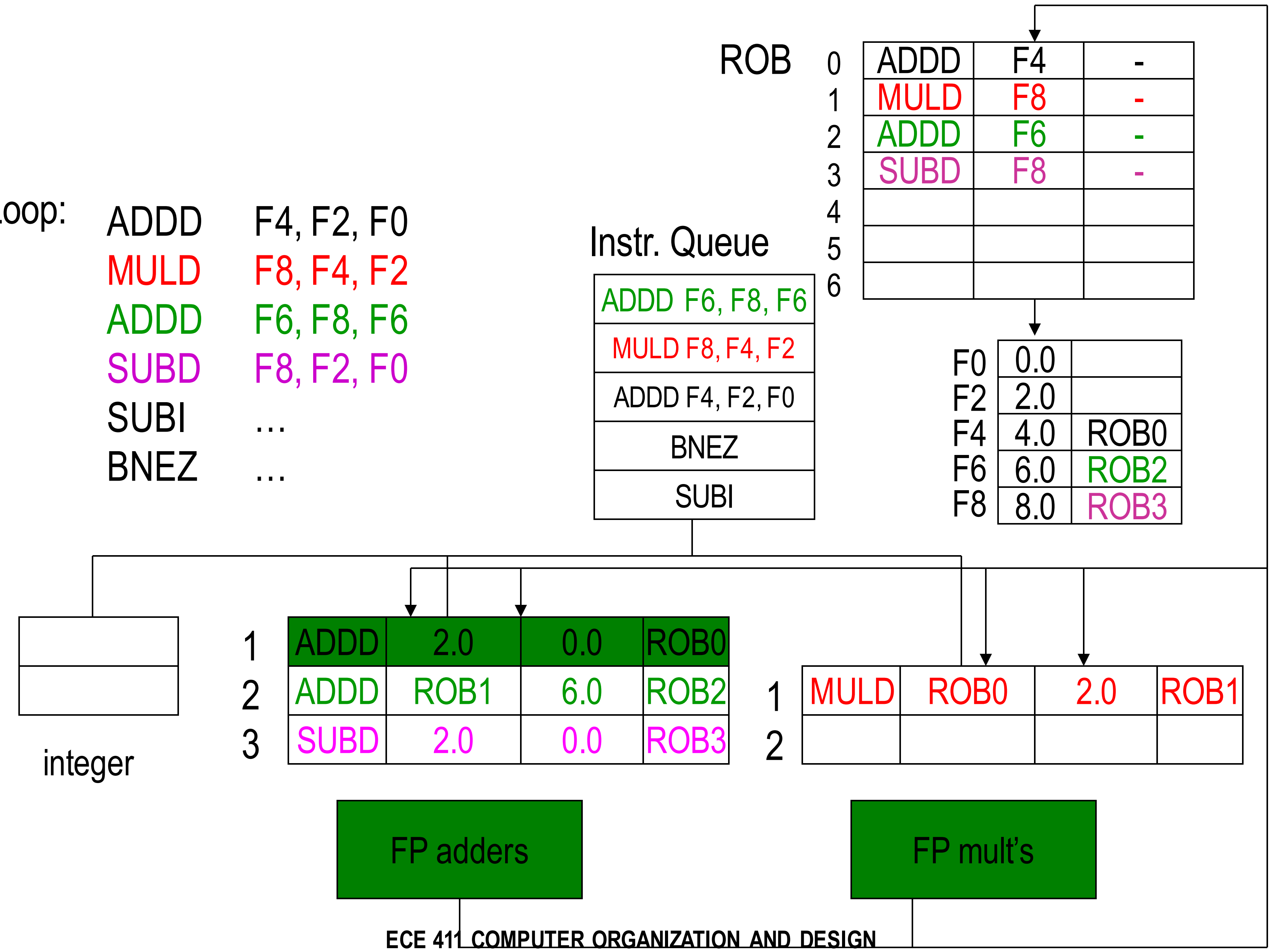
1	ADDD	2.0	0.0	ROB0
2	ADDD	ROB1	6.0	ROB2
3				

FP mult's reservation stations

1	MULD	ROB0	2.0	ROB1
2				

Tomasulo – cycle 4

```
Loop:   ADDD    F4, F2, F0
        MULD    F8, F4, F2
        ADDD    F6, F8, F6
        SUBD    F8, F2, F0
        SUBI    …
        BNEZ    …
```



ROB

0	ADDD	F4	-
1	MULD	F8	-
2	ADDD	F6	-
3	SUBD	F8	-
4			
5			
6			

Instr. Queue

ADDD F6, F8, F6
MULD F8, F4, F2
ADDD F4, F2, F0
BNEZ
SUBI

F0	0.0	
F2	2.0	
F4	4.0	ROB0
F6	6.0	ROB2
F8	8.0	ROB3

integer

FP adders

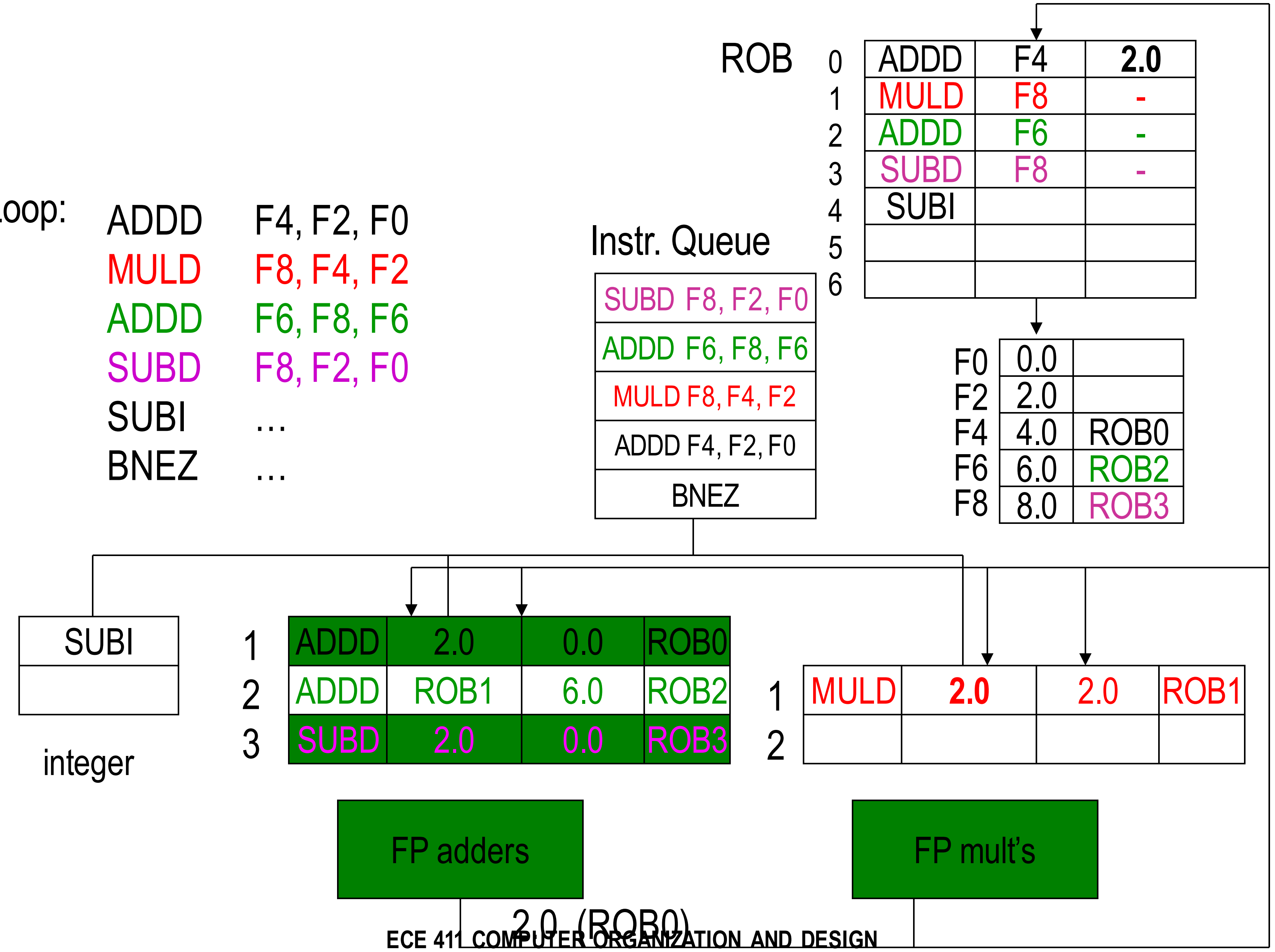
1	ADDD	2.0	0.0	ROB0
2	ADDD	ROB1	6.0	ROB2
3	SUBD	2.0	0.0	ROB3

FP mult's

1	MULD	ROB0	2.0	ROB1
2				

Tomasulo – cycle 5

```
Loop:  ADDD   F4, F2, F0
       MULD   F8, F4, F2
       ADDD   F6, F8, F6
       SUBD   F8, F2, F0
       SUBI   …
       BNEZ   …
```



ROB

0	ADDD	F4	**2.0**
1	MULD	F8	-
2	ADDD	F6	-
3	SUBD	F8	-
4	SUBI		
5			
6			

Instr. Queue

SUBD F8, F2, F0
ADDD F6, F8, F6
MULD F8, F4, F2
ADDD F4, F2, F0
BNEZ

F0	0.0	
F2	2.0	
F4	4.0	ROB0
F6	6.0	ROB2
F8	8.0	ROB3

integer

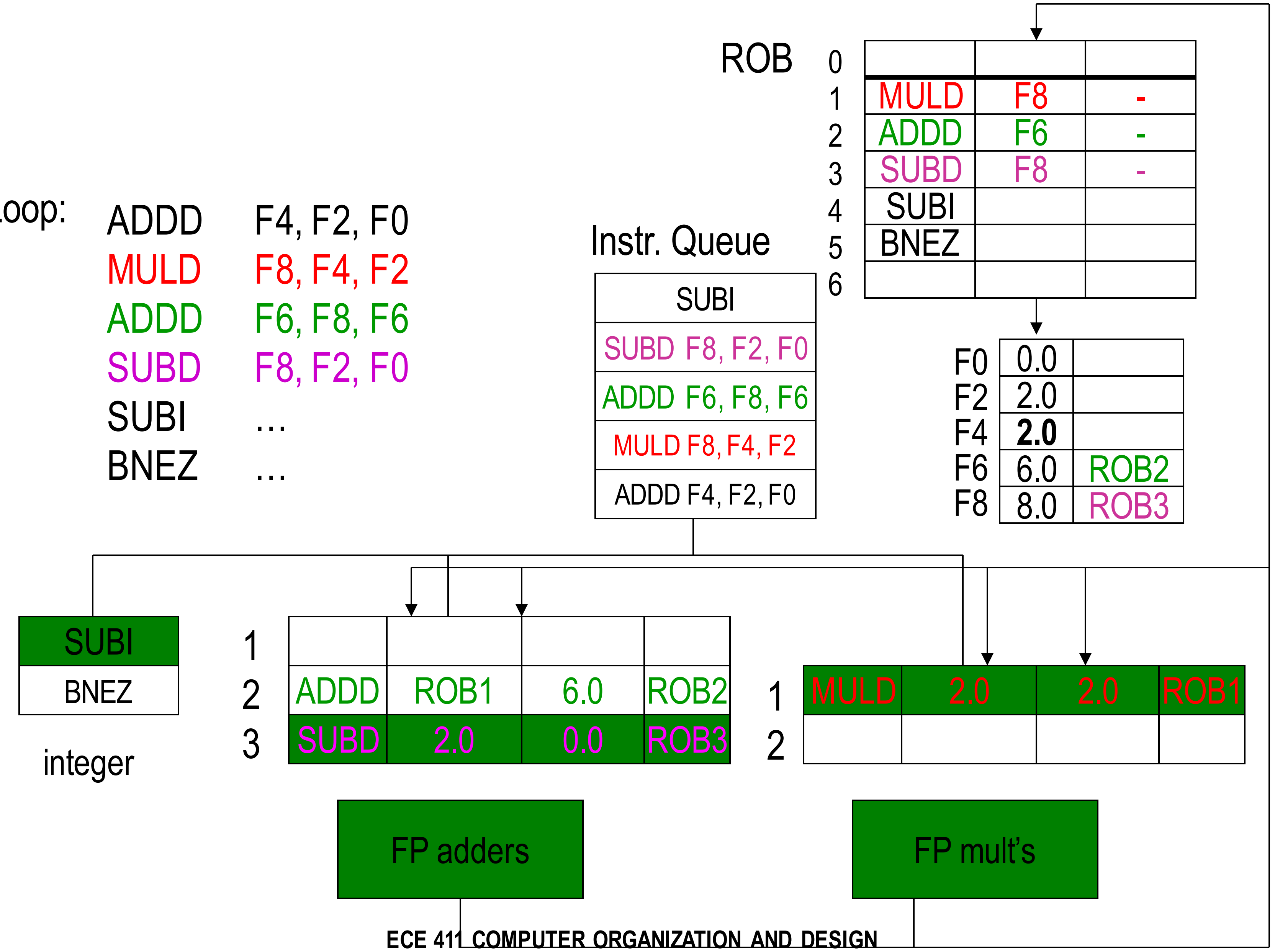
SUBI

1	ADDD	2.0	0.0	ROB0
2	ADDD	ROB1	6.0	ROB2
3	SUBD	2.0	0.0	ROB3

1	MULD	**2.0**	2.0	ROB1
2				

Tomasulo – cycle 6

Loop:		
	ADDD	F4, F2, F0
	MULD	F8, F4, F2
	ADDD	F6, F8, F6
	SUBD	F8, F2, F0
	SUBI	…
	BNEZ	…



ROB

0			
1	MULD	F8	-
2	ADDD	F6	-
3	SUBD	F8	-
4	SUBI		
5	BNEZ		
6			

Instr. Queue

SUBI
SUBD F8, F2, F0
ADDD F6, F8, F6
MULD F8, F4, F2
ADDD F4, F2, F0

F0	0.0	
F2	2.0	
F4	**2.0**	
F6	6.0	ROB2
F8	8.0	ROB3

integer

SUBI
BNEZ

FP adders

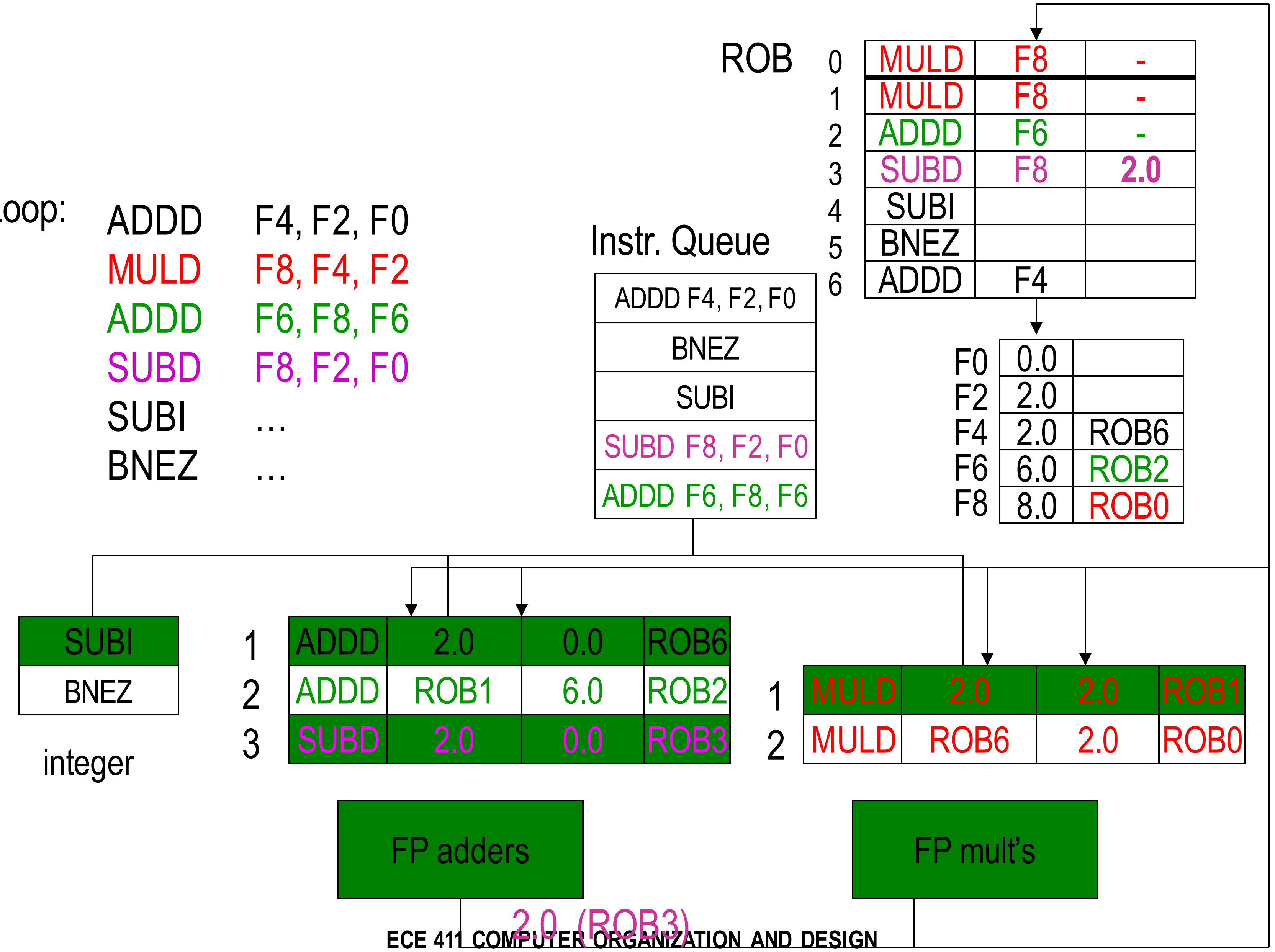
1				
2	ADDD	ROB1	6.0	ROB2
3	SUBD	2.0	0.0	ROB3

FP mult's

1	MULD	2.0	2.0	ROB1
2				

Tomasulo – cycle 8

```
Loop:  ADDD   F4, F2, F0
       MULD   F8, F4, F2
       ADDD   F6, F8, F6
       SUBD   F8, F2, F0
       SUBI   …
       BNEZ   …
```



ROB

0	MULD	F8	-
1	MULD	F8	-
2	ADDD	F6	-
3	SUBD	F8	2.0
4	SUBI		
5	BNEZ		
6	ADDD	F4	

Instr. Queue

ADDD F4, F2, F0
BNEZ
SUBI
SUBD F8, F2, F0
ADDD F6, F8, F6

F0	0.0	
F2	2.0	
F4	2.0	ROB6
F6	6.0	ROB2
F8	8.0	ROB0

integer

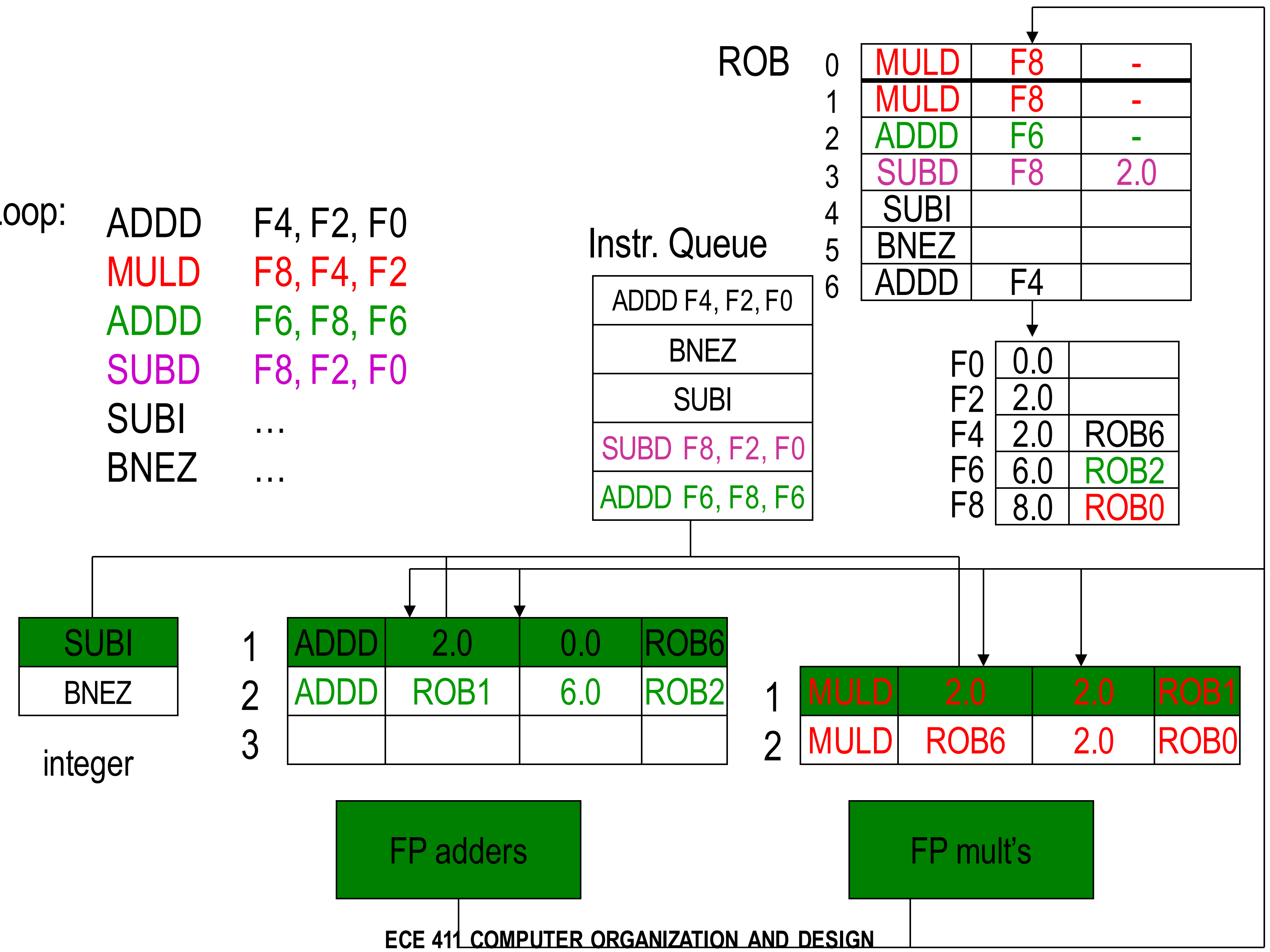
SUBI
BNEZ

1	ADDD	2.0	0.0	ROB6
2	ADDD	ROB1	6.0	ROB2
3	SUBD	2.0	0.0	ROB3

1	MULD	2.0	2.0	ROB1
2	MULD	ROB6	2.0	ROB0

Tomasulo – cycle 9

Loop:	ADDD	F4, F2, F0
	MULD	F8, F4, F2
	ADDD	F6, F8, F6
	SUBD	F8, F2, F0
	SUBI	…
	BNEZ	…



ROB

0	MULD	F8	-
1	MULD	F8	-
2	ADDD	F6	-
3	SUBD	F8	2.0
4	SUBI		
5	BNEZ		
6	ADDD	F4	

Instr. Queue

ADDD F4, F2, F0
BNEZ
SUBI
SUBD F8, F2, F0
ADDD F6, F8, F6

F0	0.0	
F2	2.0	
F4	2.0	ROB6
F6	6.0	ROB2
F8	8.0	ROB0

SUBI
BNEZ

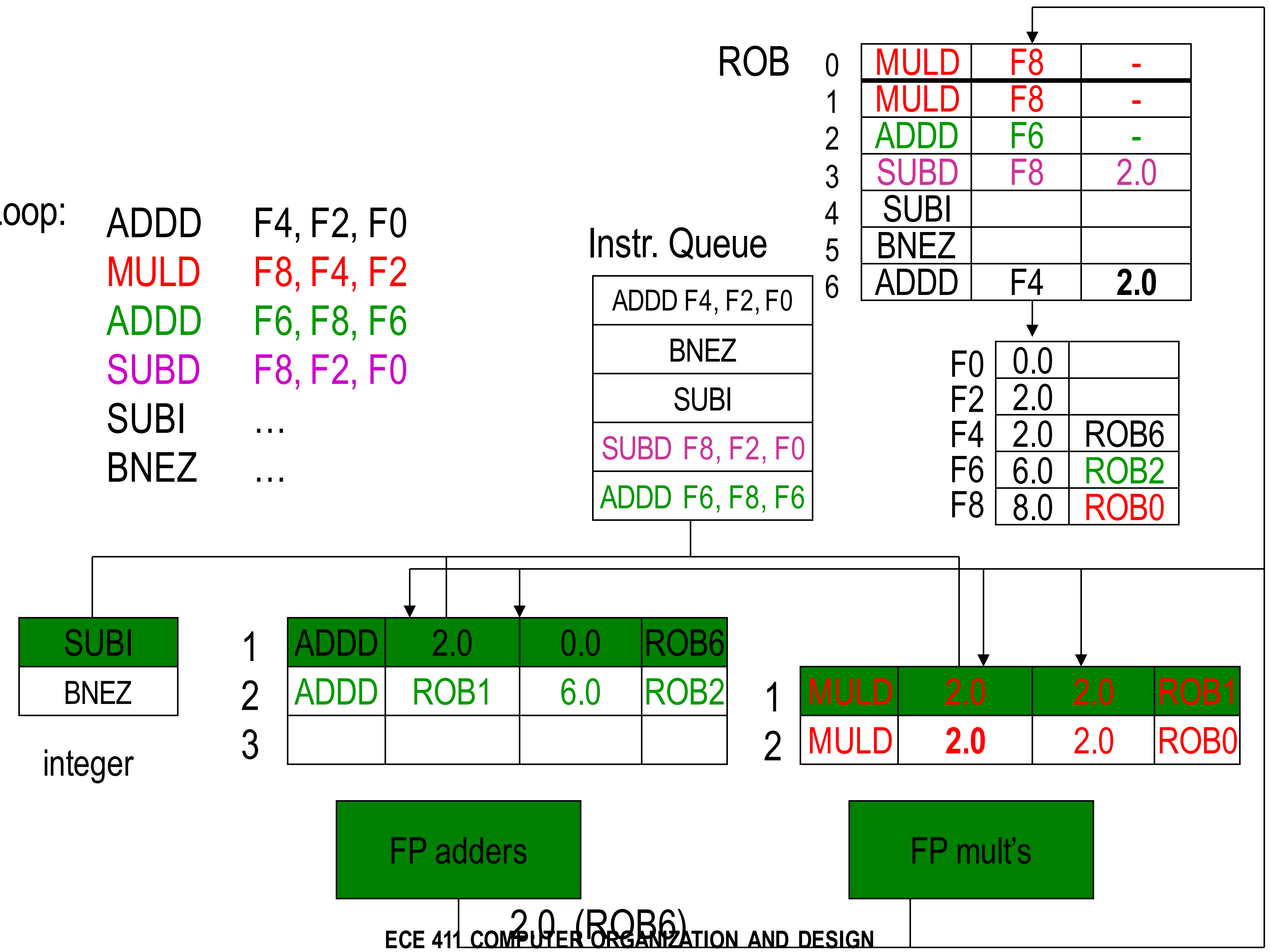
integer

1	ADDD	2.0	0.0	ROB6
2	ADDD	ROB1	6.0	ROB2
3				

1	MULD	2.0	2.0	ROB1
2	MULD	ROB6	2.0	ROB0

Tomasulo – cycle 11

Loop:		
	ADDD	F4, F2, F0
	MULD	F8, F4, F2
	ADDD	F6, F8, F6
	SUBD	F8, F2, F0
	SUBI	…
	BNEZ	…



ROB

0	MULD	F8	-
1	MULD	F8	-
2	ADDD	F6	-
3	SUBD	F8	2.0
4	SUBI		
5	BNEZ		
6	ADDD	F4	**2.0**

Instr. Queue

ADDD F4, F2, F0
BNEZ
SUBI
SUBD F8, F2, F0
ADDD F6, F8, F6

F0	0.0	
F2	2.0	
F4	2.0	ROB6
F6	6.0	ROB2
F8	8.0	ROB0

integer

SUBI
BNEZ

FP adders

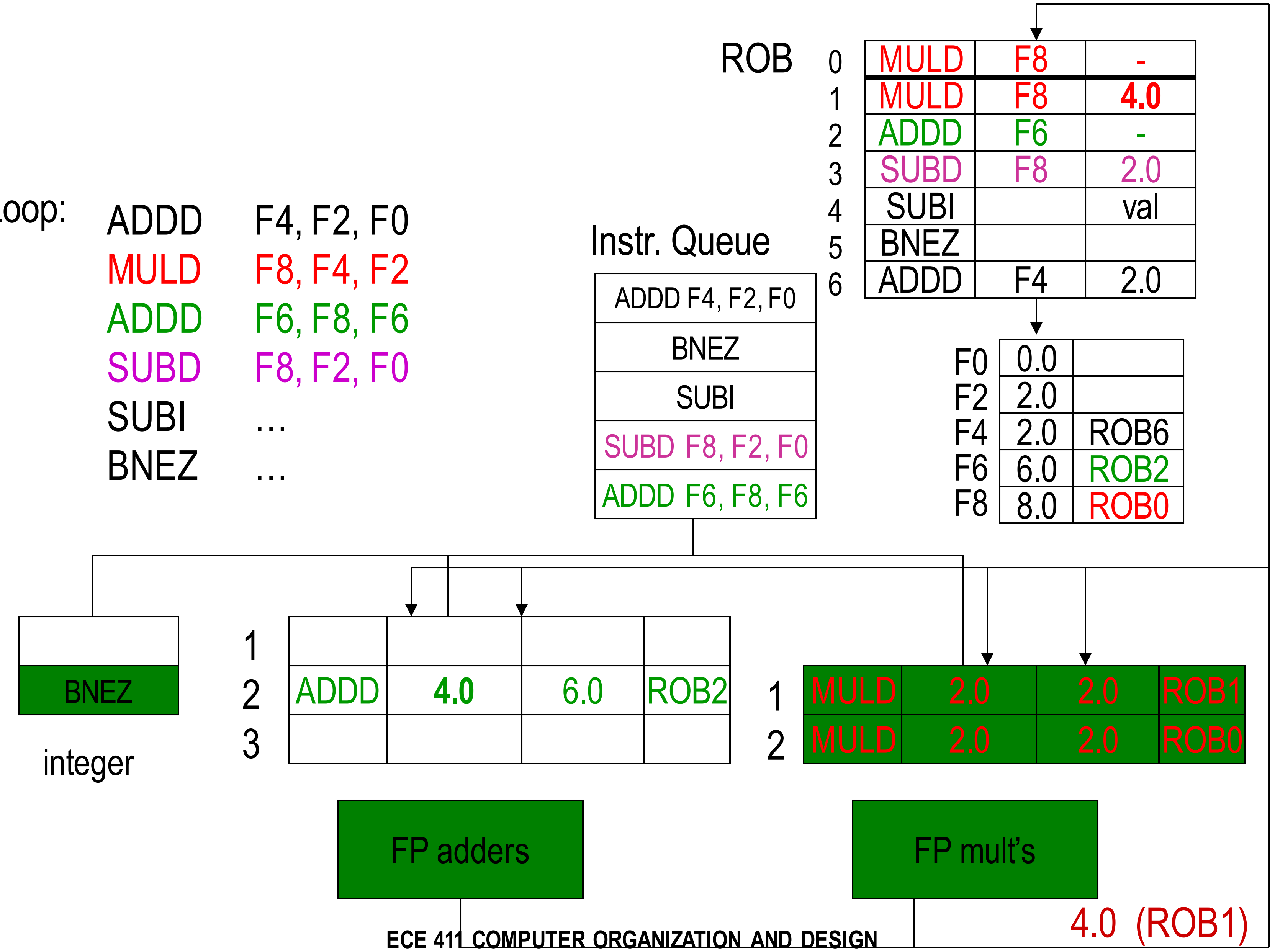
1	ADDD	2.0	0.0	ROB6
2	ADDD	ROB1	6.0	ROB2
3				

FP mult's

1	MULD	2.0	2.0	ROB1
2	MULD	**2.0**	2.0	ROB0

Tomasulo – cycle 15

```
Loop:   ADDD    F4, F2, F0
        MULD    F8, F4, F2
        ADDD    F6, F8, F6
        SUBD    F8, F2, F0
        SUBI    …
        BNEZ    …
```



ROB

0	MULD	F8	-
1	MULD	F8	**4.0**
2	ADDD	F6	-
3	SUBD	F8	2.0
4	SUBI		val
5	BNEZ		
6	ADDD	F4	2.0

Instr. Queue

ADDD F4, F2, F0
BNEZ
SUBI
SUBD F8, F2, F0
ADDD F6, F8, F6

F0	0.0	
F2	2.0	
F4	2.0	ROB6
F6	6.0	ROB2
F8	8.0	ROB0

integer

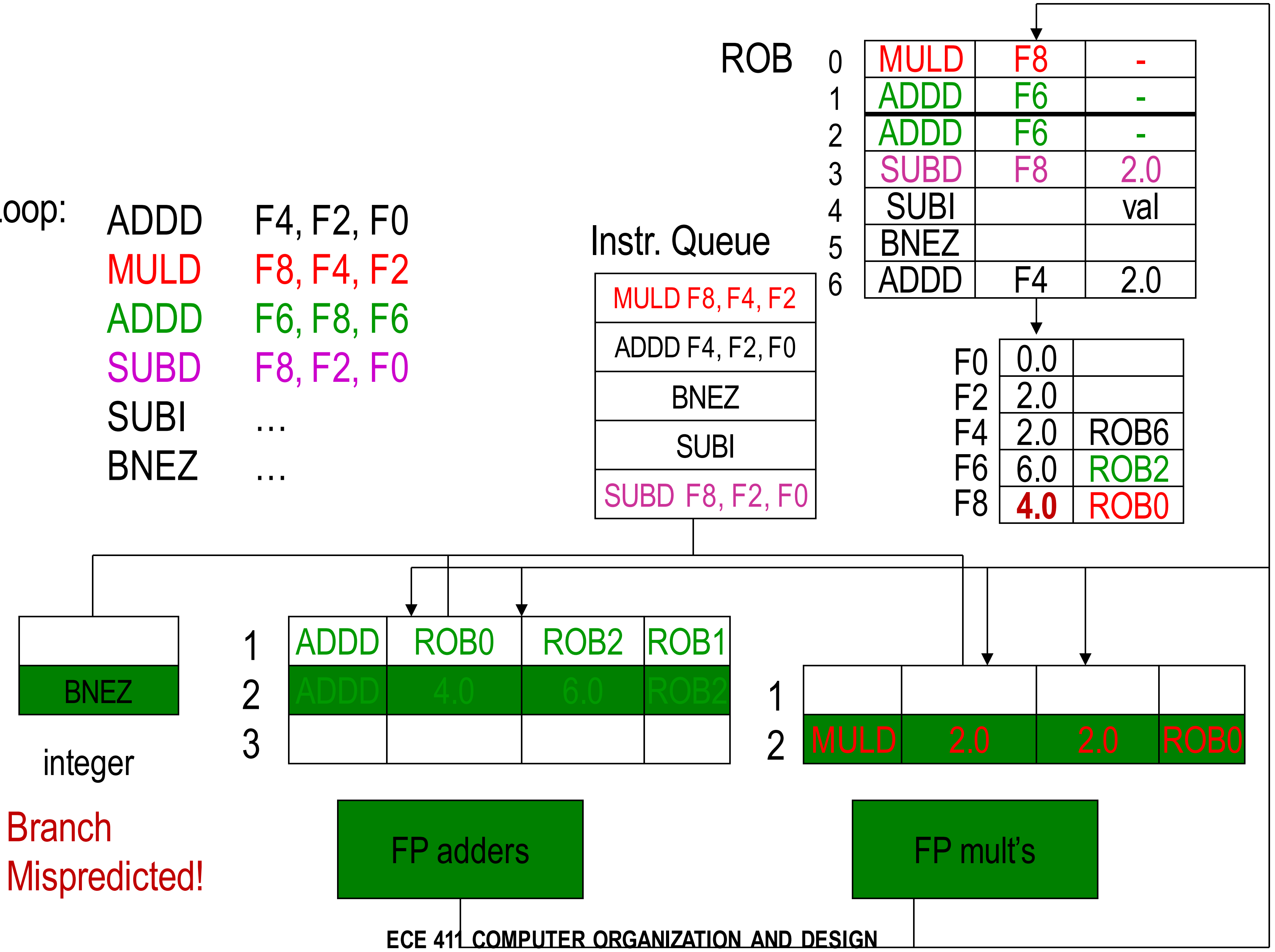
BNEZ

1				
2	ADDD	**4.0**	6.0	ROB2
3				

1	MULD	2.0	2.0	ROB1
2	MULD	2.0	2.0	ROB0

Tomasulo – cycle 16

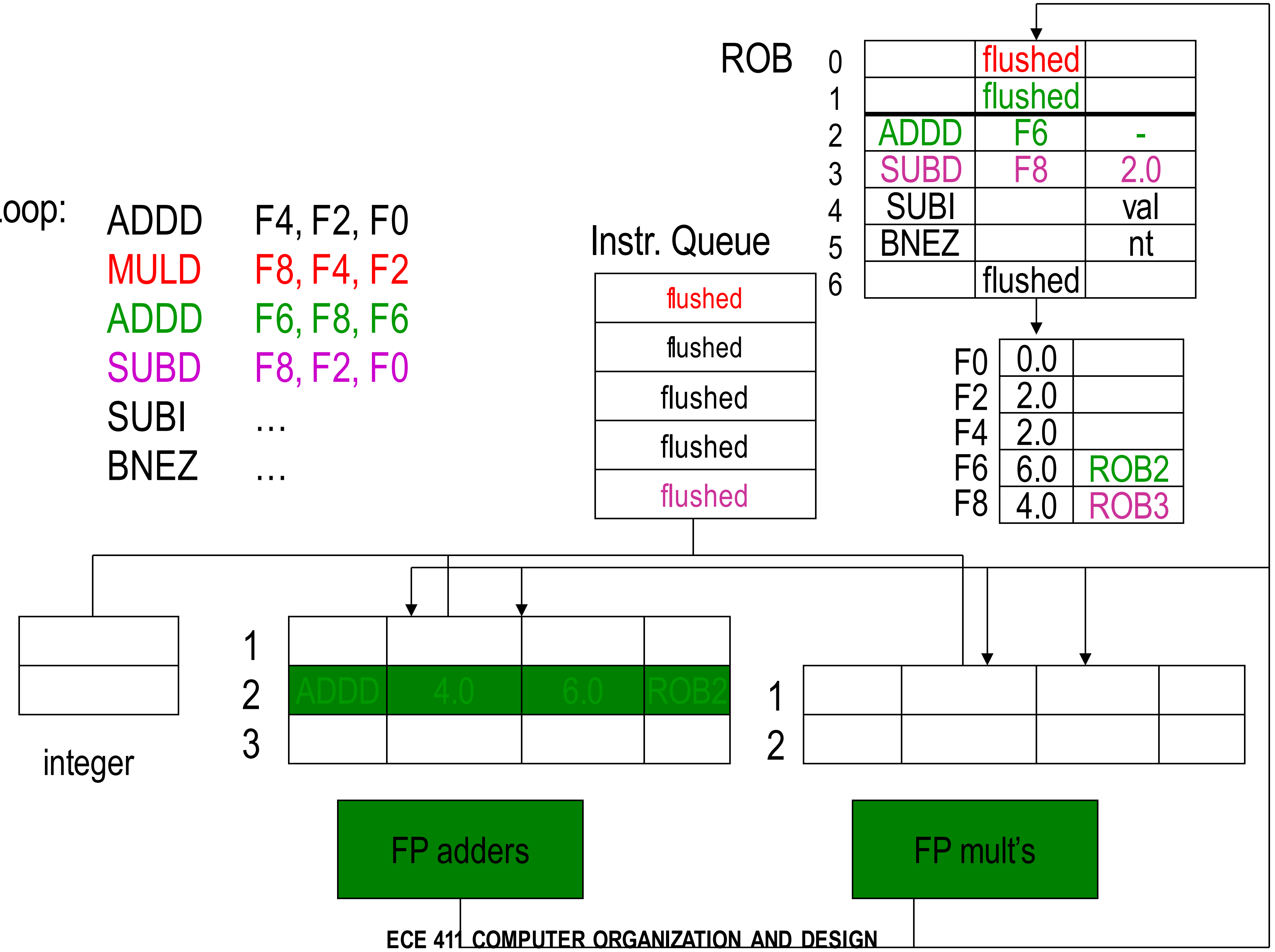
Loop:		
	ADDD	F4, F2, F0
	MULD	F8, F4, F2
	ADDD	F6, F8, F6
	SUBD	F8, F2, F0
	SUBI	…
	BNEZ	…



Branch Mispredicted!

Tomasulo – cycle 17

```
Loop:  ADDD   F4, F2, F0
       MULD   F8, F4, F2
       ADDD   F6, F8, F6
       SUBD   F8, F2, F0
       SUBI   …
       BNEZ   …
```



ROB

0		flushed	
1		flushed	
2	ADDD	F6	-
3	SUBD	F8	2.0
4	SUBI		val
5	BNEZ		nt
6		flushed	

Instr. Queue

flushed
flushed
flushed
flushed
flushed

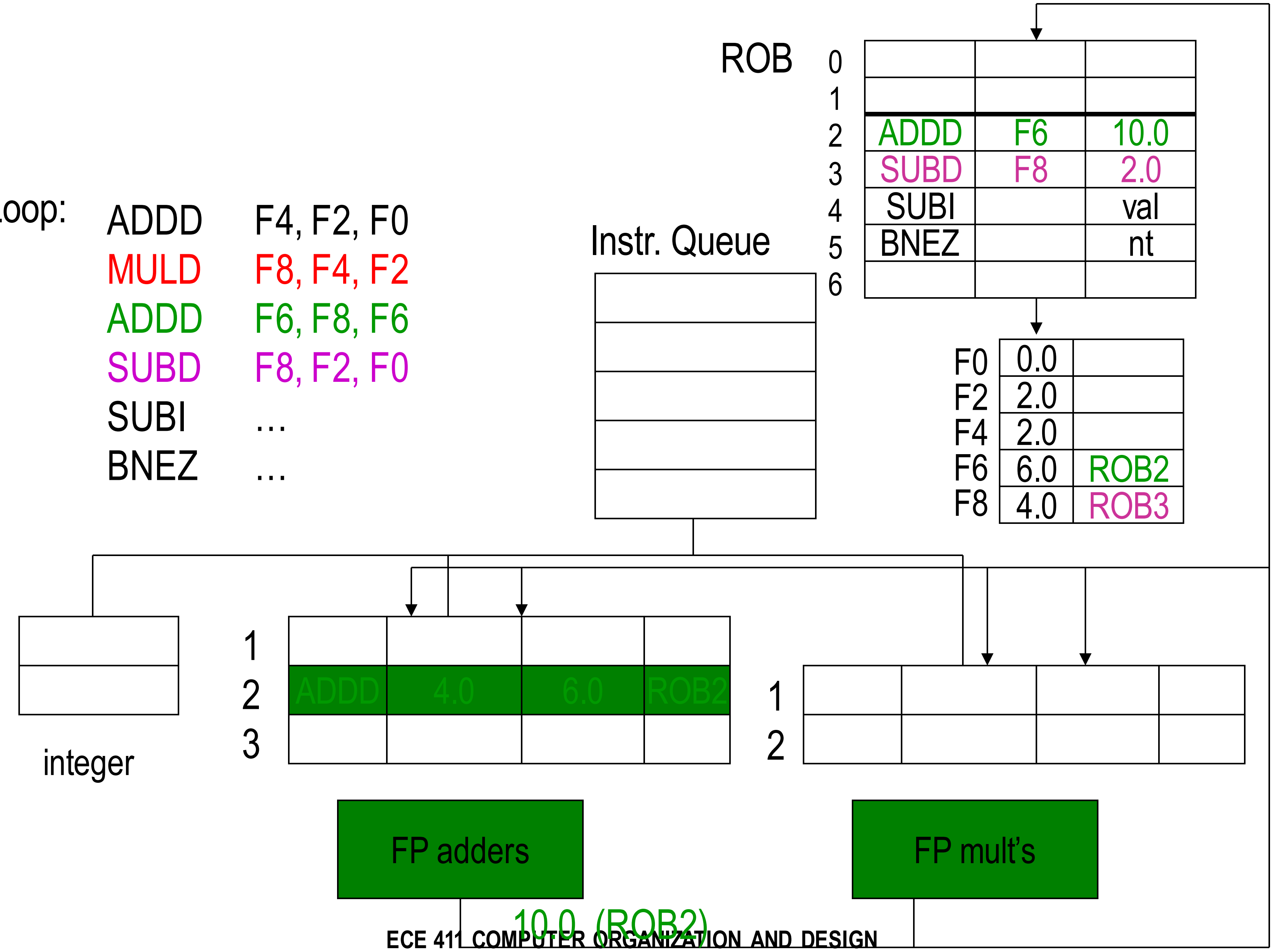
F0	0.0	
F2	2.0	
F4	2.0	
F6	6.0	ROB2
F8	4.0	ROB3

1				
2	ADDD	4.0	6.0	ROB2
3				

1				
2				

Tomasulo – cycle 19

Loop:	ADDD	F4, F2, F0
	MULD	F8, F4, F2
	ADDD	F6, F8, F6
	SUBD	F8, F2, F0
	SUBI	…
	BNEZ	…



ROB

0			
1			
2	ADDD	F6	10.0
3	SUBD	F8	2.0
4	SUBI		val
5	BNEZ		nt
6			

F0	0.0	
F2	2.0	
F4	2.0	
F6	6.0	ROB2
F8	4.0	ROB3

FP adders reservation stations

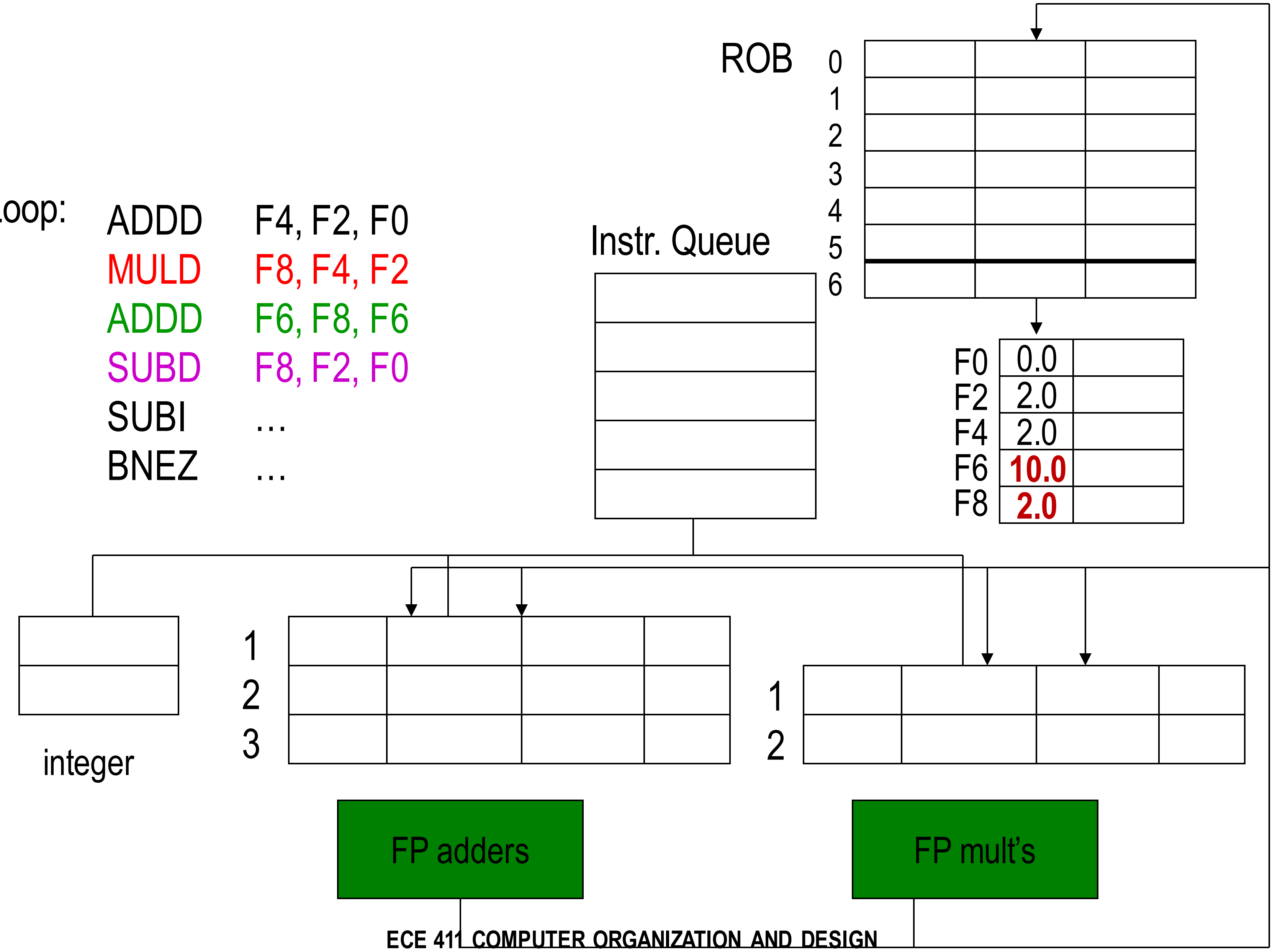
1				
2	ADDD	4.0	6.0	ROB2
3				

FP mult's reservation stations

1				
2				

Tomasulo – cycle 20

Loop:	ADDD	F4, F2, F0
	MULD	F8, F4, F2
	ADDD	F6, F8, F6
	SUBD	F8, F2, F0
	SUBI	…
	BNEZ	…



Speculative Execution

- ROB and in-order commit allow us to flush the speculative instructions from the machine when a misprediction is discovered.
- ROB is another possible source of operands
- ROB can provide __inorder commit__ in an out-of-order machine
- ROB allows us to _precisely handle__ exceptions on speculative code

Issues With Out-of-Order Execution

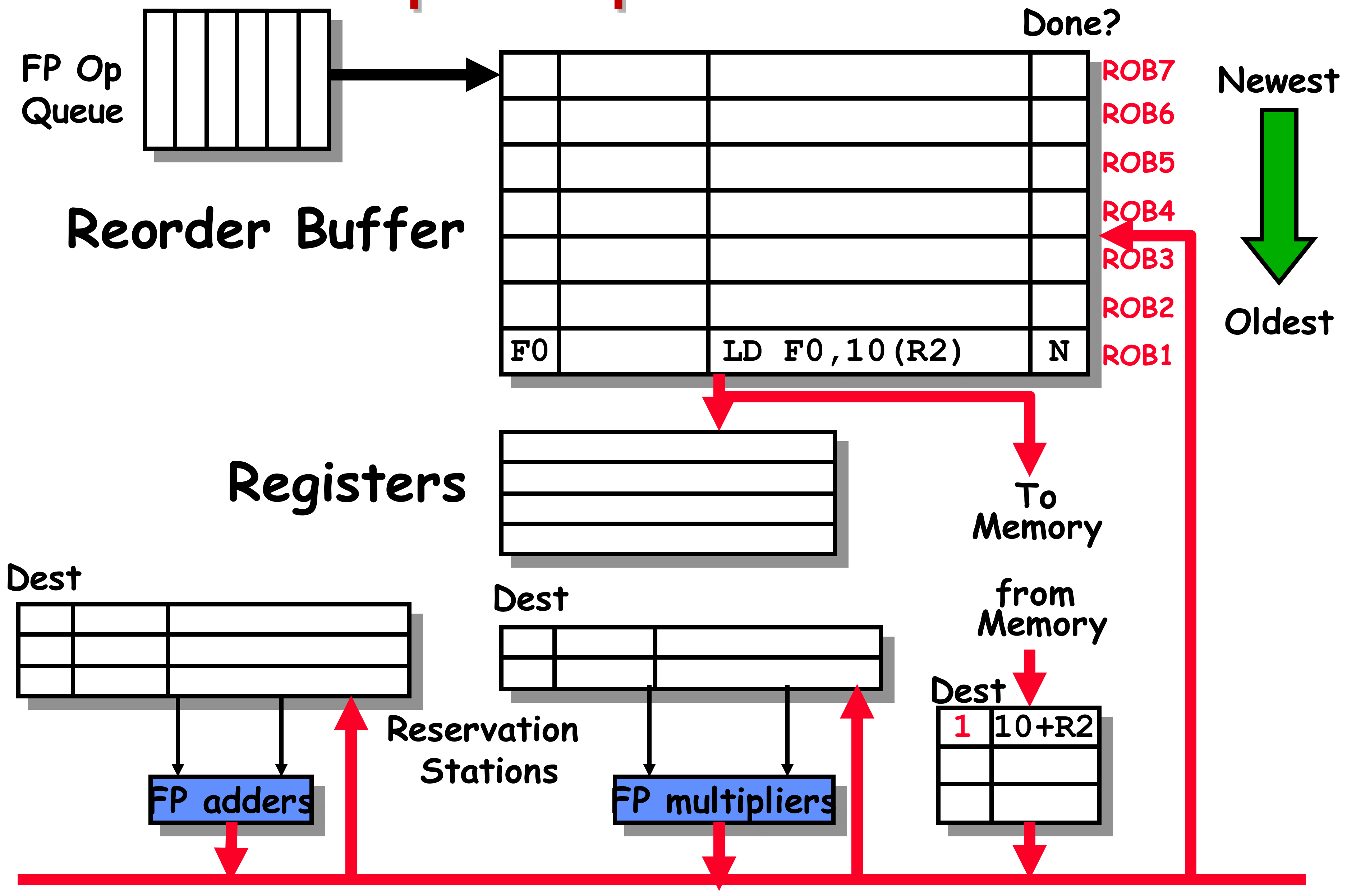
- size of reservation stations and reorder buffer grows with number of instructions in flight at any time
- processors can only examine a limited window of instructions each cycle to decide which to issue

Another Example of Speculative Tomasulo

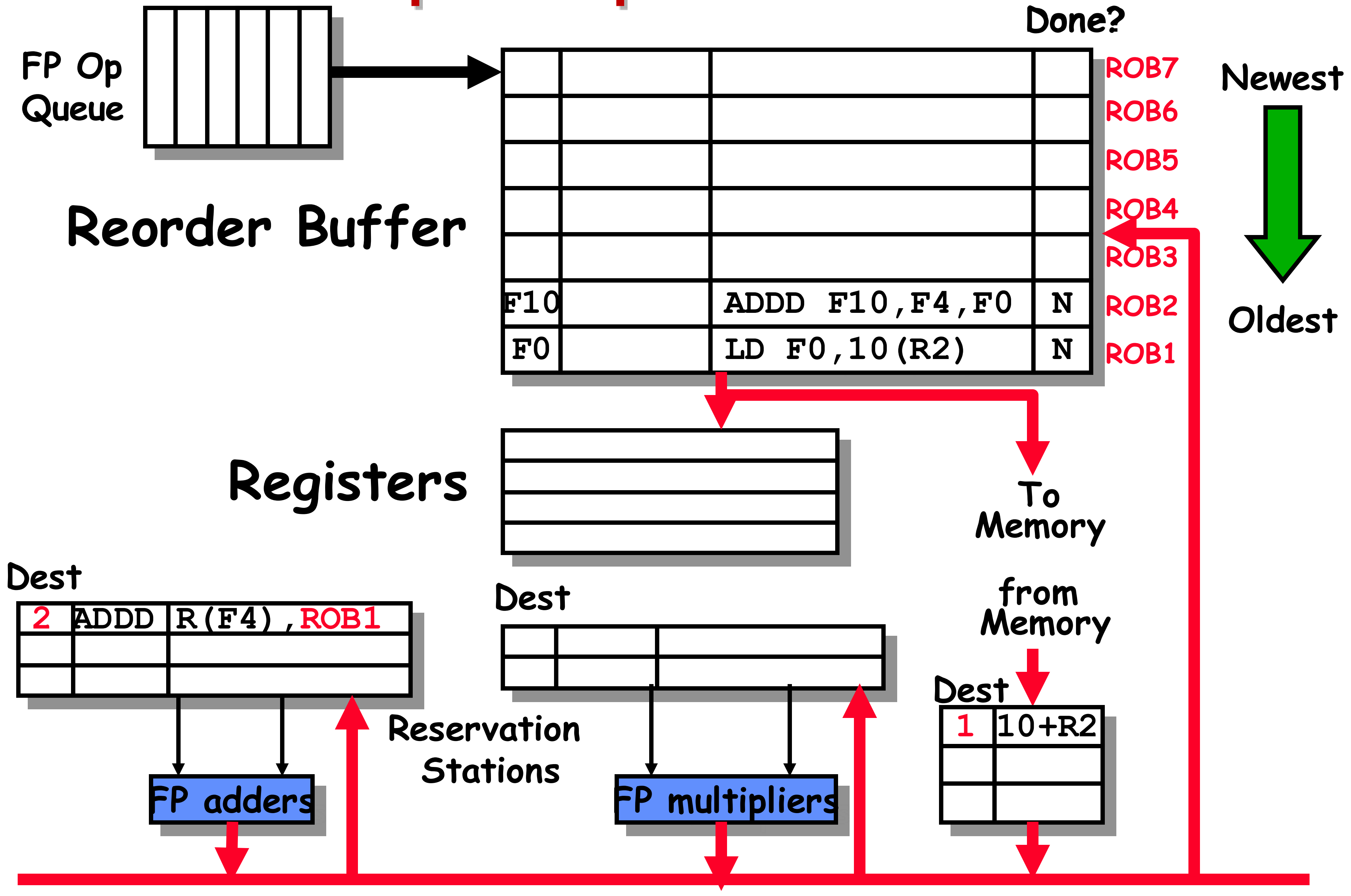
```
          LD          F0    10    R2
          ADDD        F10   F4    F0
          DIVD        F2    F10   F6
          BNEZ        F2    Exit

          LD          F4    0     R3
          ADDD        F0    F4    F9
          SD          F4    0     R3
          …
     Exit:
```

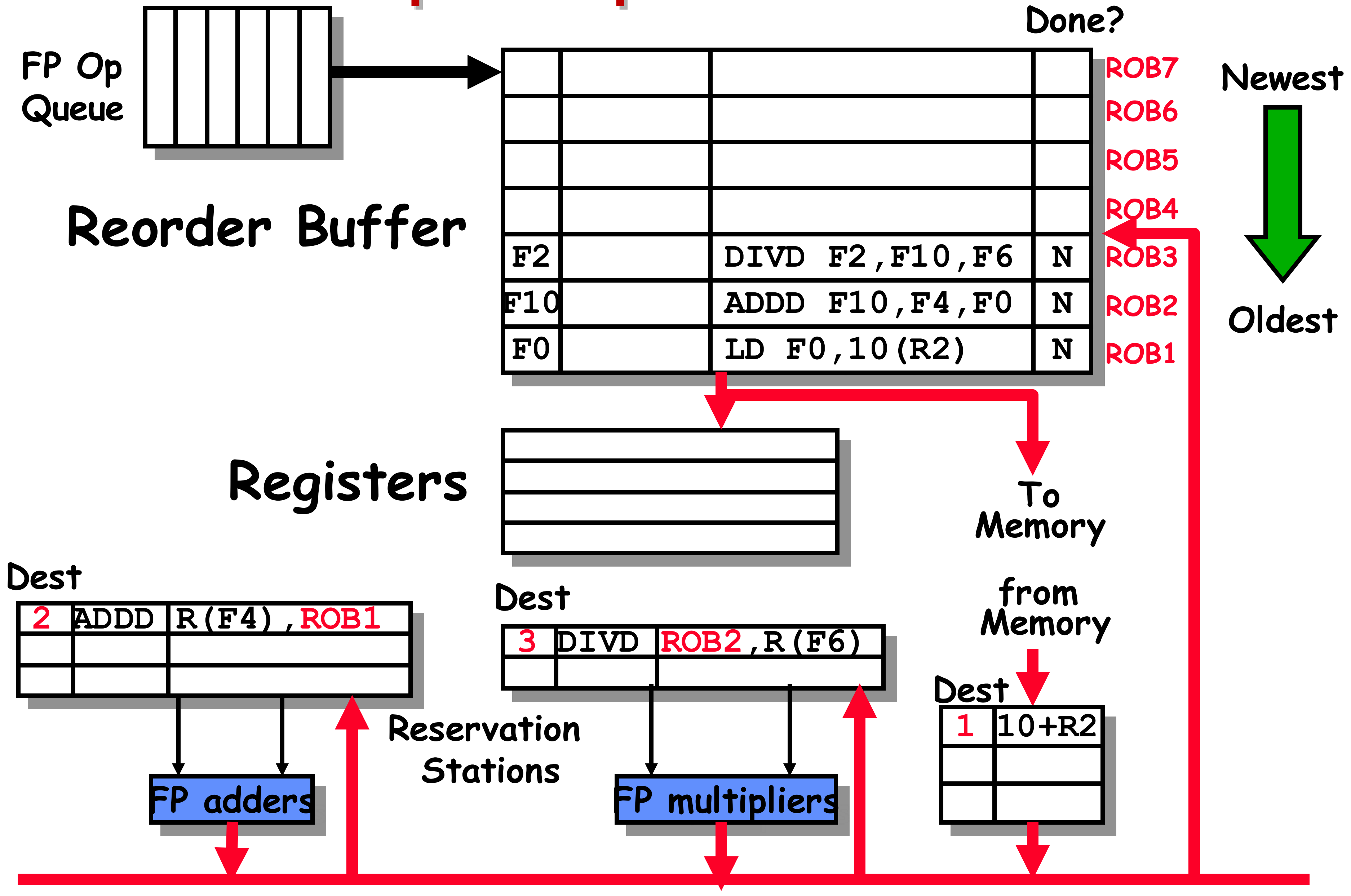
Another Example of Speculative Tomasulo



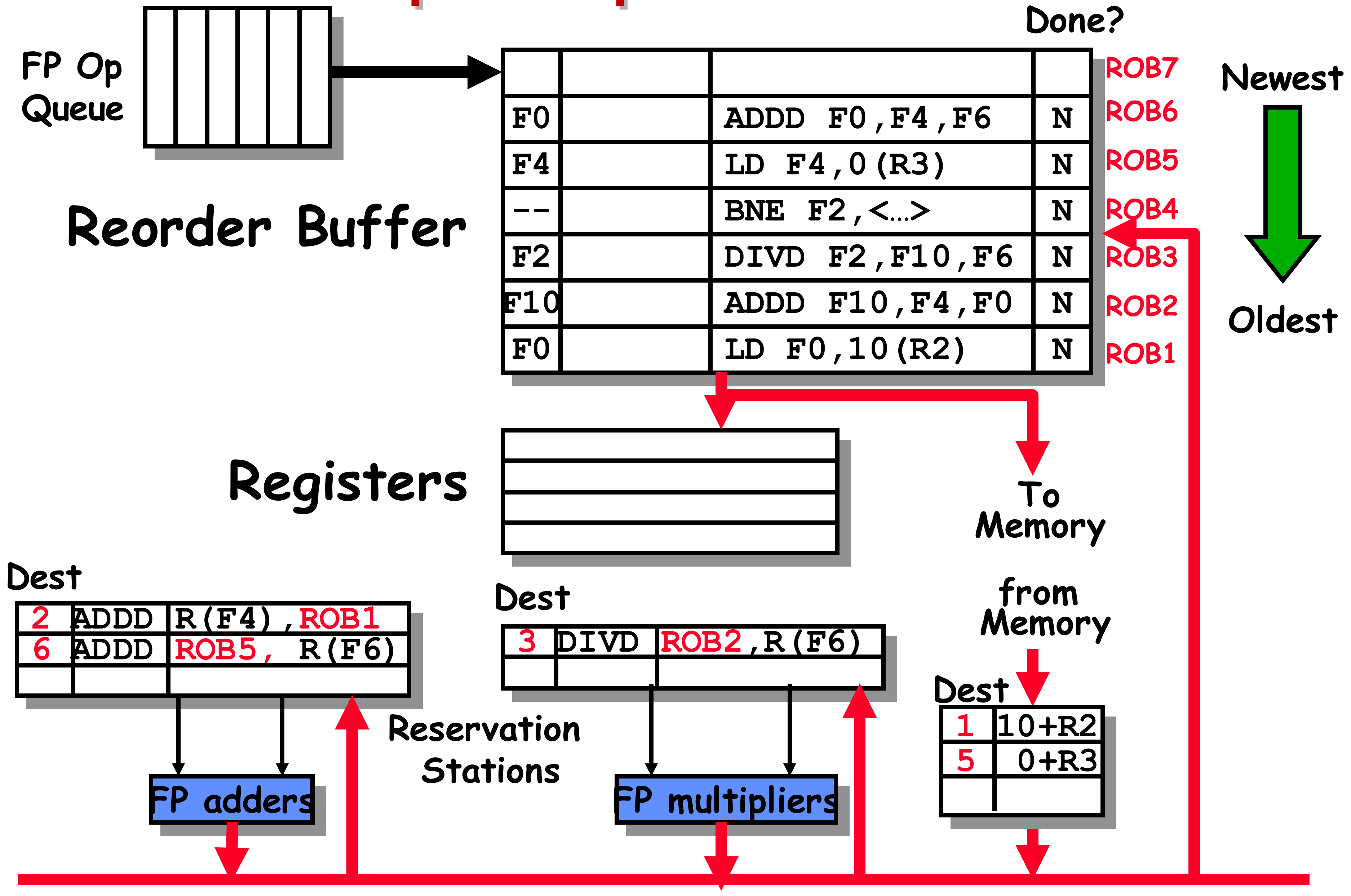
Another Example of Speculative Tomasulo



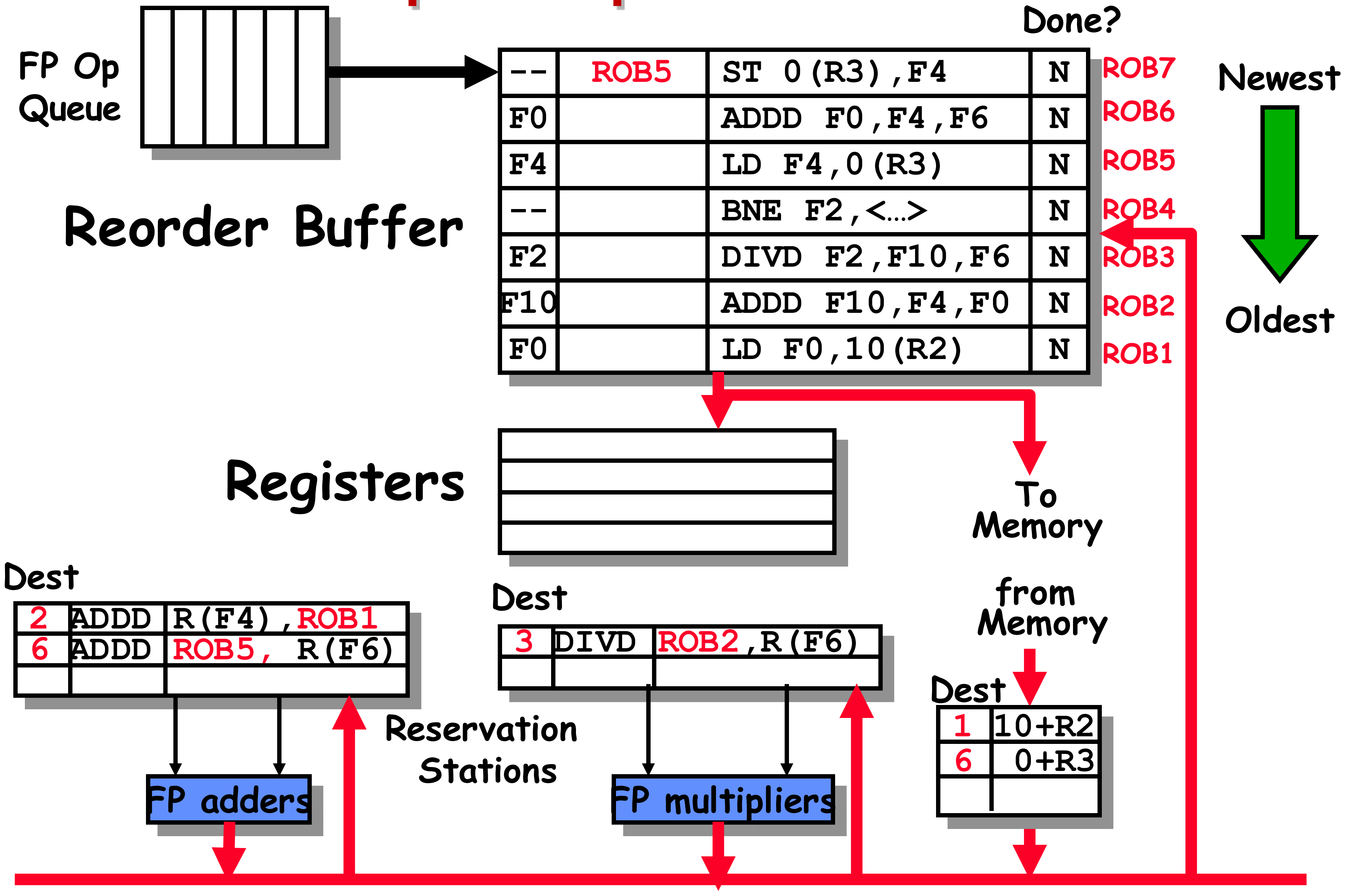
Another Example of Speculative Tomasulo



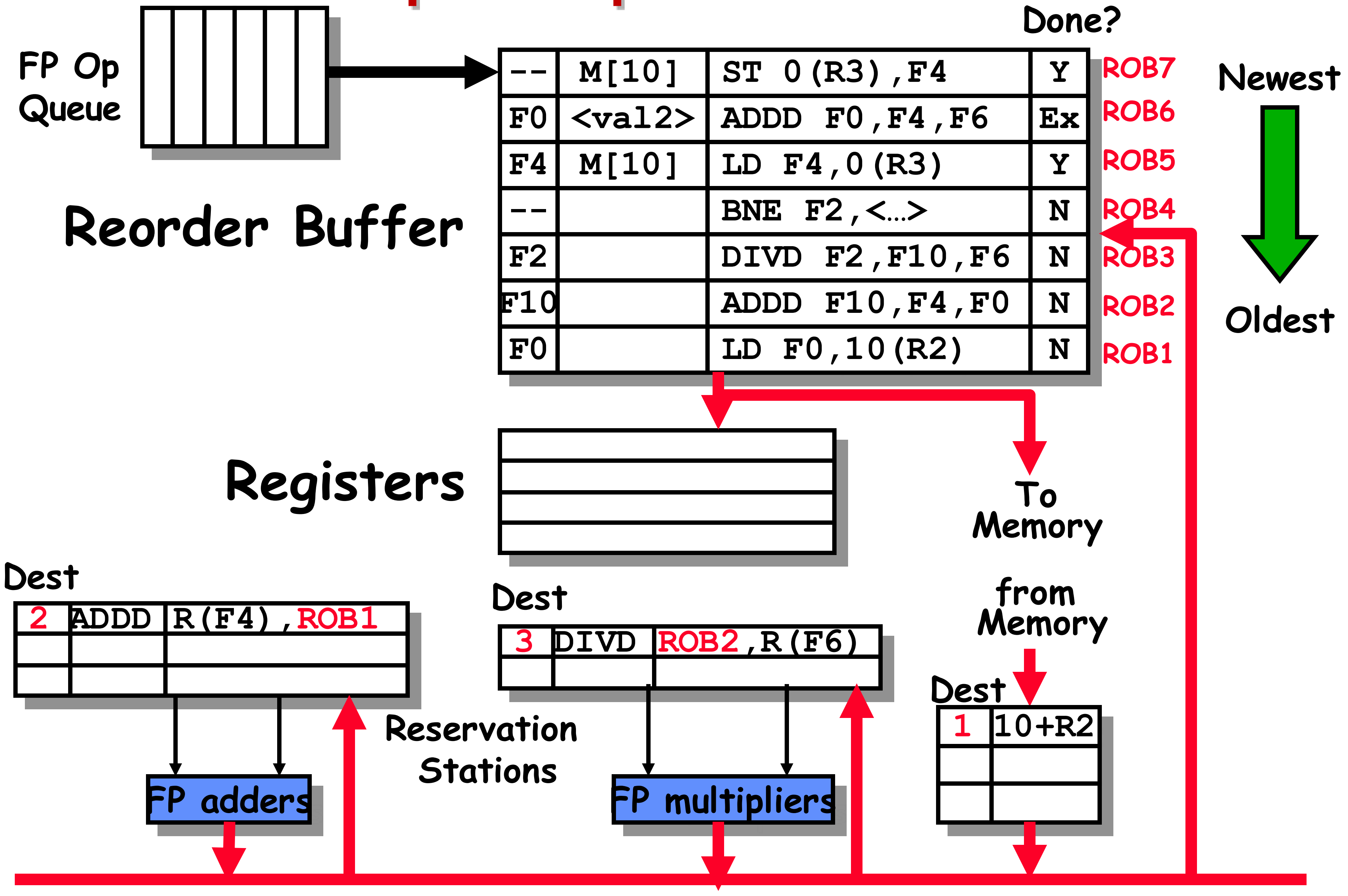
Another Example of Speculative Tomasulo



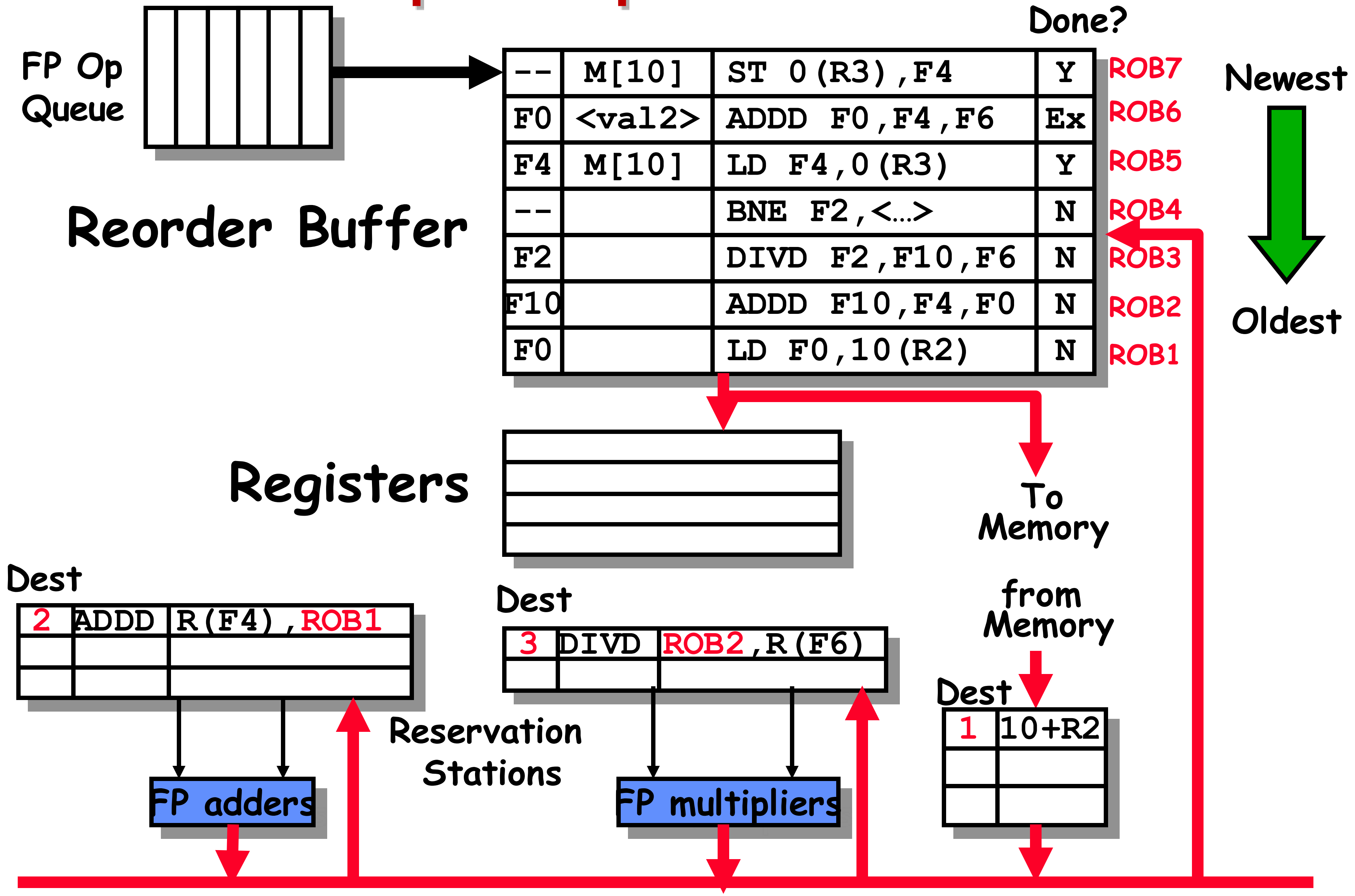
Another Example of Speculative Tomasulo


FP Op Queue
Reorder Buffer
Done?
-- | ROB5 | ST 0(R3),F4 | N | ROB7
F0 | | ADDD F0,F4,F6 | N | ROB6
F4 | | LD F4,0(R3) | N | ROB5
-- | | BNE F2,<…> | N | ROB4
F2 | | DIVD F2,F10,F6 | N | ROB3
F10 | | ADDD F10,F4,F0 | N | ROB2
F0 | | LD F0,10(R2) | N | ROB1
Newest
Oldest
Registers
To Memory
from Memory
Dest
2 ADDD R(F4),ROB1
6 ADDD ROB5, R(F6)
Dest
3 DIVD ROB2,R(F6)
Dest
1 10+R2
6 0+R3
Reservation Stations
FP adders
FP multipliers

Another Example of Speculative Tomasulo


FP Op Queue
Done?
-- | M[10] | ST 0(R3),F4 | Y | ROB7
F0 | <val2> | ADDD F0,F4,F6 | Ex | ROB6
F4 | M[10] | LD F4,0(R3) | Y | ROB5
-- | | BNE F2,<…> | N | ROB4
F2 | | DIVD F2,F10,F6 | N | ROB3
F10 | | ADDD F10,F4,F0 | N | ROB2
F0 | | LD F0,10(R2) | N | ROB1
Newest
Oldest
Reorder Buffer
Registers
To Memory
from Memory
Dest
2 ADDD R(F4),ROB1
Dest
3 DIVD ROB2,R(F6)
Dest
1 10+R2
Reservation Stations
FP adders
FP multipliers

Another Example of Speculative Tomasulo



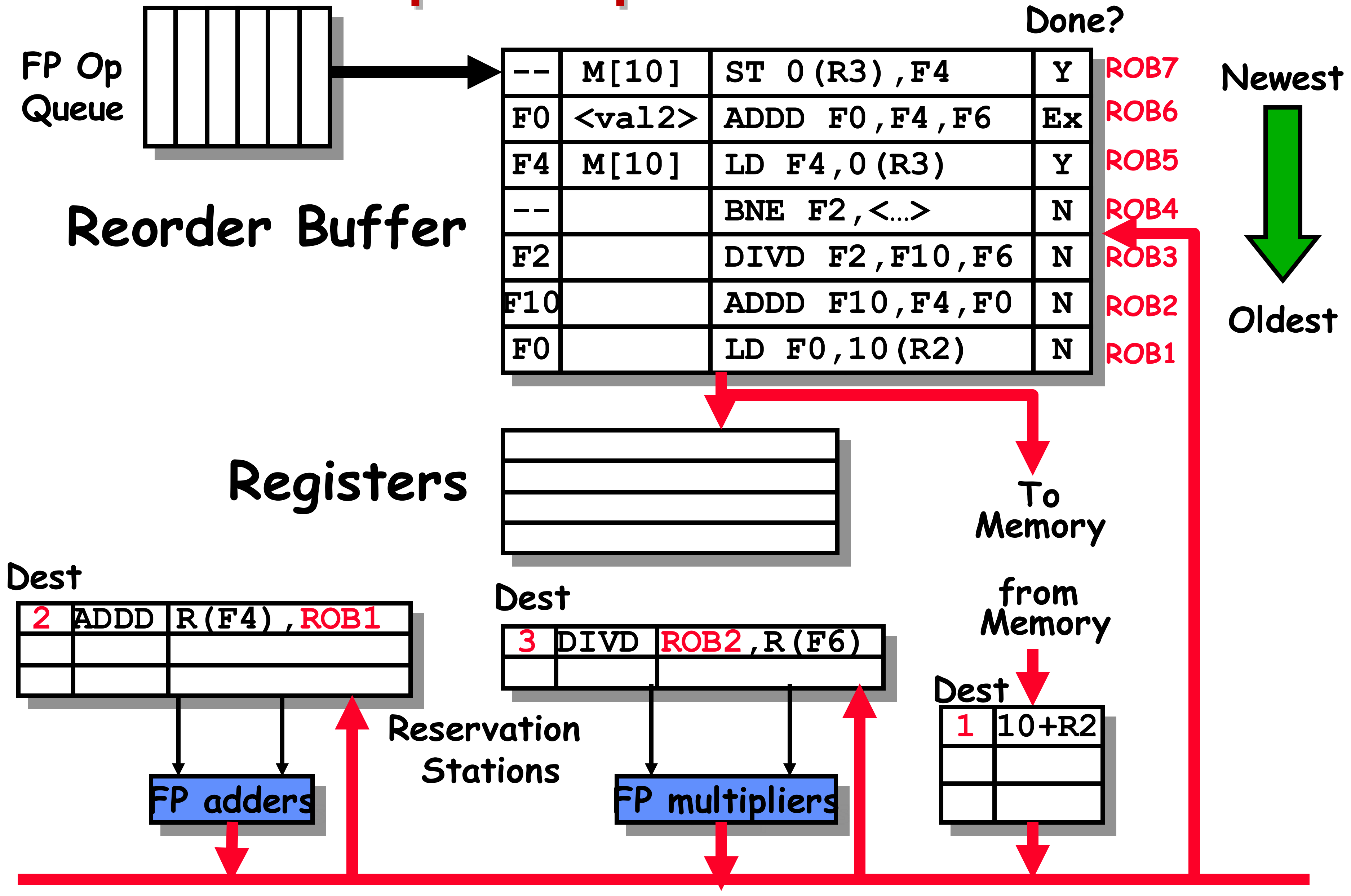
--	M[10]	ST 0(R3),F4	Y	ROB7
F0	<val2>	ADDD F0,F4,F6	Ex	ROB6
F4	M[10]	LD F4,0(R3)	Y	ROB5
--		BNE F2,<…>	N	ROB4
F2		DIVD F2,F10,F6	N	ROB3
F10		ADDD F10,F4,F0	N	ROB2
F0		LD F0,10(R2)	N	ROB1

Dest		
2	ADDD	R(F4),ROB1

Dest		
3	DIVD	ROB2,R(F6)

Dest	
1	10+R2

Another Example of Speculative Tomasulo



			Done?	
--	M[10]	ST 0(R3),F4	Y	ROB7
F0	<val2>	ADDD F0,F4,F6	Ex	ROB6
F4	M[10]	LD F4,0(R3)	Y	ROB5
--		BNE F2,<…>	N	ROB4
F2		DIVD F2,F10,F6	N	ROB3
F10		ADDD F10,F4,F0	N	ROB2
F0		LD F0,10(R2)	N	ROB1

FP adders reservation stations:

Dest		
2	ADDD	R(F4),ROB1

FP multipliers reservation stations:

Dest		
3	DIVD	ROB2,R(F6)

Load buffer:

Dest	
1	10+R2

Announcement

- next lecture: multi-core and multi-threading
  - ✓ Ch. 6.4 – 6.5 (HP1)
- MP assignment
  - ✓ MP3 checkpoint 2 due on 3/18 5pm