

Exercise Session 9

Tomasulo + ROB, Scoreboard + Cache Miss, Cache Coerency

Advanced Computer Architectures

Politecnico di Milano

May 28th, 2025

Alessandro Verosimile <alessandro.verosimile@polimi.it>



POLITECNICO
MILANO 1863

POLITECNICO MILANO 1863
NECST
laboratory

Recall: Separating Completion from Commit

Re-order buffer holds register results from completion until commit

Entries allocated in program order during decode

Buffers completed values and exception state until in-order commit point

Completed values can be used by dependents before committed (bypassing)

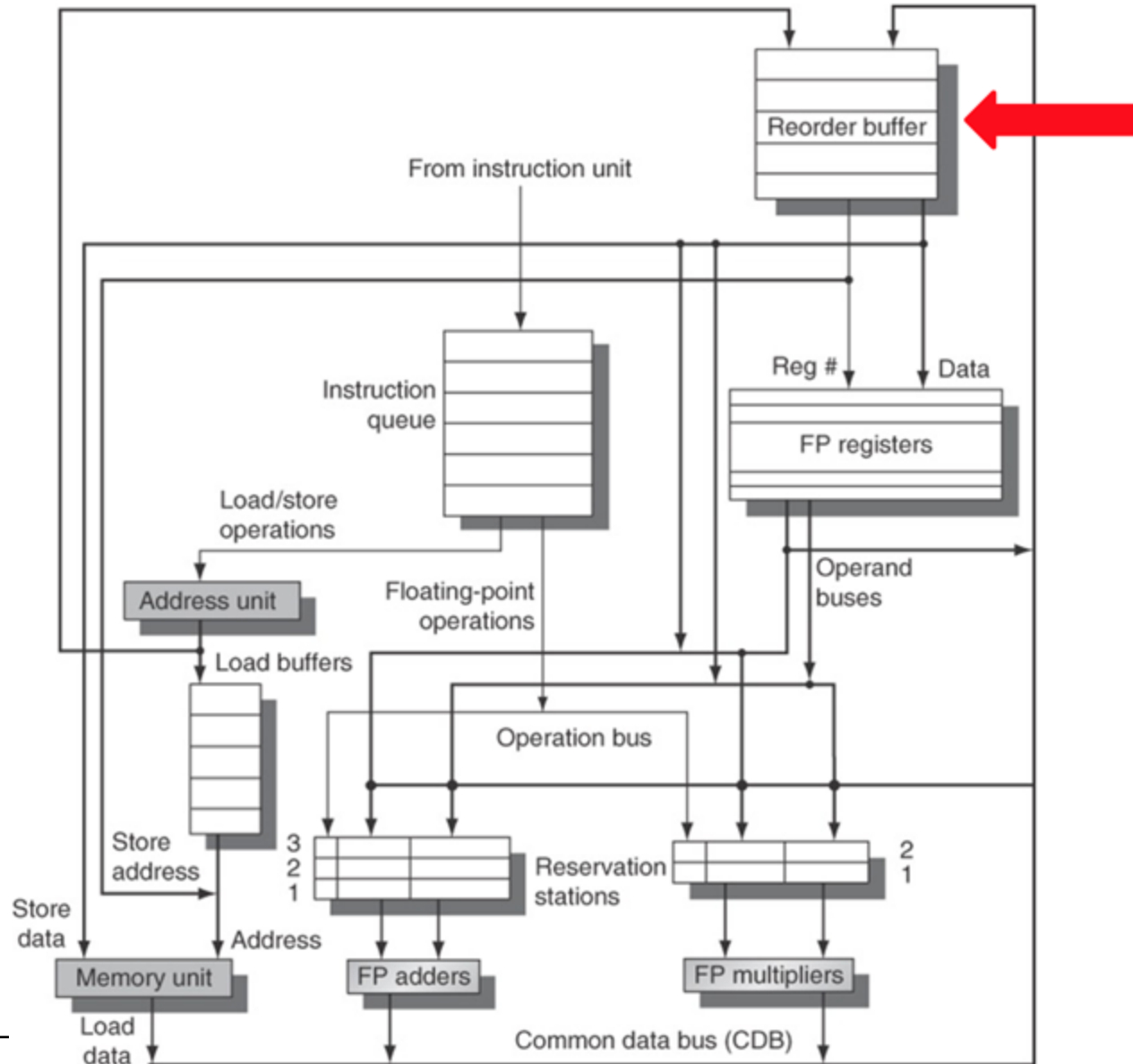
Each entry holds program counter, instruction type, destination register specifier and value if any, and exception status (info often compressed to save hardware)

Memory reordering needs special data structures

Speculative store address and data buffers

Speculative load address and data buffers

Exe Tomasulo with Reorder Buffer



Recall: the Tomasulo pipeline with ROB

ISSUE	EXECUTION	WRITE	COMMIT
Get Instruction from Queue and Rename Registers Add Instruction to ROB	Execute and Watch CDB;	Write on CDB; Write on ROB	Update register with result (or store to memory); remove Instr from ROB
Structural RSs check; Structural ROB check; WAW and WAR solved by Renaming (!!!in-order-issue!!!);	Check for Struct on FUs; RAW delaying; Struct check on CDB;	(FUs will hold results unless CDB free) RSs/FUs marked free	In-order commit; Mispredicted branch flushes ROB

Exe 4 Tomasulo with ROB: the Code

```
LOOP:I1: MULTD F2, F6, F8  
      I2: ADDD F0, F6, F4  
      I3: MULTD F10, F0, F2  
      I4: ADDD F0, F12, F14  
      I5: SUBI R0, R0, 4  
      I6: BNEZ R0, LOOP
```

Exe 4 Tomasulo with ROB: the Conflicts

```
LOOP:I1: MULTD F2, F6, F8  
      I2: ADDD F0, F6, F4  
      I3: MULTD F10, F0, F2  
      I4: ADDD F0, F12, F14  
      I5: SUBI R0, R0, 4  
      I6: BNEZ R0, LOOP
```

Exe 4 Tomasulo with ROB: the Conflicts

LOOP:I1: MULTD **F2**, F6, F8
I2: ADDD F0, F6, F4
I3: MULTD F10, F0, **F2**
I4: ADDD F0, F12, F14
I5: SUBI R0, R0, 4
I6: BNEZ R0, LOOP

RAW **F2** I1-I3

Exe 4 Tomasulo with ROB: the Conflicts

LOOP:I1: MULTD **F2**, F6, F8
I2: ADDD **F0**, F6, F4
I3: MULTD F10, **F0**, **F2**
I4: ADDD F0, F12, F14
I5: SUBI R0, R0, 4
I6: BNEZ R0, LOOP

RAW **F2** I1-I3

RAW **F0** I2-I3

Exe 4 Tomasulo with ROB: the Conflicts

LOOP:I1: MULTD **F2**, F6, F8
I2: ADDD **F0**, F6, F4
I3: MULTD F10, **F0**, **F2**
I4: ADDD F0, F12, F14
I5: SUBI **R0**, R0, 4
I6: BNEZ **R0**, LOOP

RAW **F2** I1-I3

RAW **F0** I2-I3

RAW **R0** I5-I6

Exe 4 Tomasulo with ROB: the Conflicts

LOOP:I1: MULTD **F2**, F6, F8
I2: ADDD **F0**, F6, F4
I3: MULTD F10, **F0**, **F2**
I4: ADDD **F0**, F12, F14
I5: SUBI **R0**, R0, 4
I6: BNEZ **R0**, LOOP

RAW **F2** I1-I3

RAW **F0** I2-I3

RAW **R0** I5-I6

WAW **F0** I2-I4

Exe 4 Tomasulo with ROB: the Conflicts

LOOP:I1: MULTD **F2**, F6, F8
I2: ADDD **F0**, F6, F4
I3: MULTD F10, **F0**, **F2**
I4: ADDD **F0**, F12, F14
I5: SUBI **R0**, R0, 4
I6: BNEZ **R0**, LOOP

RAW **F2** I1-I3
RAW **F0** I2-I3
RAW **R0** I5-I6
WAW **F0** I2-I4
WAR **F0** I3-I4

Exe.4 Tomasulo with ROB

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB
- Enough RS and FUs for integer operations, and separated CDB

To be clear, will show when the SUBI and BNEZ are issued

In the case of hazard on CDB, the oldest instruction has priority

Let's assume that we discover the BNEZ misprediction 5 clock cycles after the issue

Exe Tomasulo with ROB CC0

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8								
It. 1: ADDD F0, F6, F4								
It. 1: MULTD F10, F0, F2								
It. 1: ADDD F0, F12, F14								
It. 2: MULTD F2, F6, F8								
It. 2: ADDD F0, F6, F4								
It. 2: MULTD F10, F0, F2								
It. 2: ADDD F0, F12, F14								

RAW **F2** I1-I3 RAW **F0** I2-I3

Exe Tomasulo with ROB CC1

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1,MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4,RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1				0		RS1	
It. 1: ADDD F0, F6, F4								
It. 1: MULTD F10, F0, F2								
It. 1: ADDD F0, F12, F14								
It. 2: MULTD F2, F6, F8								
It. 2: ADDD F0, F6, F4								
It. 2: MULTD F10, F0, F2								
It. 2: ADDD F0, F12, F14								

RAW **F2** I1-I3 RAW **F0** I2-I3

Exe Tomasulo with ROB CC2

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2			0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2				1		RS4	
It. 1: MULTD F10, F0, F2								
It. 1: ADDD F0, F12, F14								
It. 2: MULTD F2, F6, F8								
It. 2: ADDD F0, F6, F4								
It. 2: MULTD F10, F0, F2								
It. 2: ADDD F0, F12, F14								

RAW **F2** I1-I3 RAW **F0** I2-I3

Exe Tomasulo with ROB CC3

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2			0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3			1		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3				2		RS2	
It. 1: ADDD F0, F12, F14								
It. 2: MULTD F2, F6, F8								
It. 2: ADDD F0, F6, F4								
It. 2: MULTD F10, F0, F2								
It. 2: ADDD F0, F12, F14								

RAW **F2** I1-I3 RAW **F0** I2-I3

Exe Tomasulo with ROB CC4

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2			0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3			1		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3				2	RAW F2, F0	RS2	
It. 1: ADDD F0, F12, F14	4				3		RS5	
It. 2: MULTD F2, F6, F8								
It. 2: ADDD F0, F6, F4								
It. 2: MULTD F10, F0, F2								
It. 2: ADDD F0, F12, F14								

RAW **F2** I1-I3 RAW **F0** I2-I3

Exe Tomasulo with ROB CC5

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2			0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3	5		1		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3				2	RAW F2, F0	RS2	
It. 1: ADDD F0, F12, F14	4	5			3		RS5	ADDD2
It. 2: MULTD F2, F6, F8								
It. 2: ADDD F0, F6, F4								
It. 2: MULTD F10, F0, F2								
It. 2: ADDD F0, F12, F14								

SUBI issued, ROB idx = 4

RAW **F2** I1-I3 RAW **F0** I2-I3

Exe Tomasulo with ROB CC6

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It.1: MULTD F2, F6, F8	1	2	6		0		RS1	MULT1
It.1: ADDD F0, F6, F4	2	3	5		1		RS4	ADDD1
It.1: MULTD F10, F0, F2	3				2	RAW F2, F0	RS2	
It.1: ADDD F0, F12, F14	4	5			3		RS5	ADDD2
It.2: MULTD F2, F6, F8								
It.2: ADDD F0, F6, F4								
It.2: MULTD F10, F0, F2								
It.2: ADDD F0, F12, F14								

BNEZ issued, ROB idx = 5

RAW **F2** I1-I3 RAW **F0** I2-I3

0 CC elapsed to misprediction

Exe Tomasulo with ROB CC7

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2	6	7	0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3	5		1		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3	7			2	RAW F2, F0	RS2	MULT2
It. 1: ADDD F0, F12, F14	4	5	7		3		RS5	ADDD2
It. 2: MULTD F2, F6, F8	7				6		RS1	
It. 2: ADDD F0, F6, F4								
It. 2: MULTD F10, F0, F2								
It. 2: ADDD F0, F12, F14								

RAW **F2** I1-I3 RAW **F0** I2-I3

1 CC elapsed to misprediction

Exe Tomasulo with ROB CC8

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2	6	7	0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3	5	8	1		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3	7			2	RAW F2, F0	RS2	MULT2
It. 1: ADDD F0, F12, F14	4	5	7		3		RS5	ADDD2
It. 2: MULTD F2, F6, F8	7	8			6		RS1	MULT1
It. 2: ADDD F0, F6, F4	8				0		RS4	
It. 2: MULTD F10, F0, F2								
It. 2: ADDD F0, F12, F14								

RAW **F2** I1-I3 RAW **F0** I2-I3

2 CCs elapsed to misprediction

Exe Tomasulo with ROB CC9

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2	6	7	0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3	5	8	4		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3	7			2	RAW F2, F0	RS2	MULT2
It. 1: ADDD F0, F12, F14	4	5	7		3		RS5	ADDD2
It. 2: MULTD F2, F6, F8	7	8			6		RS1	MULT1
It. 2: ADDD F0, F6, F4	8	9			0		RS4	ADDD1
It. 2: MULTD F10, F0, F2	9				1		RS3	
It. 2: ADDD F0, F12, F14								

RAW **F2** I1-I3 RAW **F0** I2-I3

3 CCs elapsed to misprediction

Exe Tomasulo with ROB CC10

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2	6	7	0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3	5	8	4		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3	7			2	RAW F2, F0	RS2	MULT2
It. 1: ADDD F0, F12, F14	4	5	7		3		RS5	ADDD2
It. 2: MULTD F2, F6, F8	7	8			6		RS1	MULT1
It. 2: ADDD F0, F6, F4	8	9			0		RS4	ADDD1
It. 2: MULTD F10, F0, F2	9				1	RAW F2, F0	RS3	
It. 2: ADDD F0, F12, F14						Struct ROB		

No ROB slots available for
ADDD F0, F12, F14

RAW **F2** I1-I3 RAW **F0** I2-I3

4 CCs elapsed to misprediction

Exe Tomasulo with ROB CC11 (part 1)

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2	6	7	0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3	5	8	4		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3	7	11		2	RAW F2, F0	RS2	MULT2
It. 1: ADDD F0, F12, F14	4	5	7		3		RS5	ADDD2
It. 2: MULTD F2, F6, F8	7	8			6		RS1	MULT1
It. 2: ADDD F0, F6, F4	8	9			0	Struct CDB	RS4	ADDD1
It. 2: MULTD F10, F0, F2	9				1	RAW F2, F0	RS3	
It. 2: ADDD F0, F12, F14						Struct ROB		

RAW **F2** I1-I3 RAW **F0** I2-I3

5 CCs elapsed to misprediction

Exe Tomasulo with ROB CC11 (part 2)

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2	6	7	0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3	5	8	4		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3	7	11		2	RAW F2, F0	RS2	MULT2
It. 1: ADDD F0, F12, F14	4	5	7		3		RS5	ADDD2
It. 2: MULTD F2, F6, F8	7	8			6		RS1	MULT1
It. 2: ADDD F0, F6, F4	8	9			0	Struct CDB	RS4	ADDD1
It. 2: MULTD F10, F0, F2	9				1	RAW F2, F0	RS3	
It. 2: ADDD F0, F12, F14						Struct ROB		

Branch mispredicted!! Need to flush ROB
instructions after the branch

RAW **F2** I1-I3 RAW **F0** I2-I3

5 CCs elapsed to misprediction

Exe Tomasulo with ROB CC12

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2	6	7	0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3	5	8	4		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3	7	11	12	2	RAW F2, F0	RS2	MULT2
It. 1: ADDD F0, F12, F14	4	5	7		3		RS5	ADDD2
It. 2: MULTD F2, F6, F8	7	8			6		RS1	MULT1
It. 2: ADDD F0, F6, F4	8	9			0	Struct CDB	RS4	ADDD1
It. 2: MULTD F10, F0, F2	9				4	RAW F2, F0	RS3	
It. 2: ADDD F0, F12, F14						Struct ROB		

Branch mispredicted!! Need to flush ROB
instructions after the branch

RAW **F2** I1-I3 RAW **F0** I2-I3

Exe Tomasulo with ROB CC13

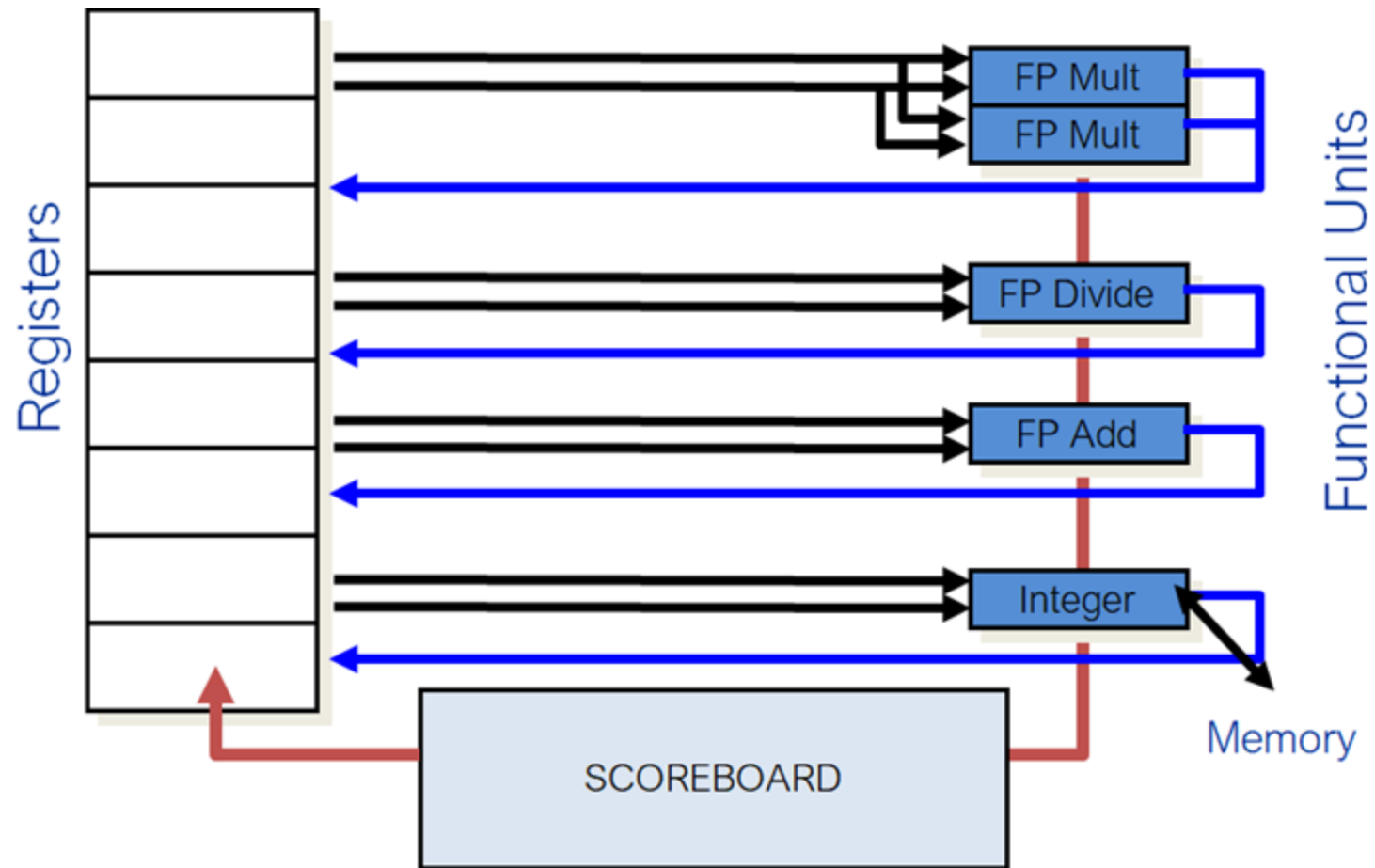
- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 MULT unit (MULT1, MULT2, MULT3) with latency 4
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 ADDD/SUBD (ADDD1, ADDD2, ADDD3) with latency 2
- 7-slot ROB

Instruction	ISSUE	START EXE	WB	Commit	ROB idx	Hazards Type	RSi	Unit
It. 1: MULTD F2, F6, F8	1	2	6	7	0		RS1	MULT1
It. 1: ADDD F0, F6, F4	2	3	5	8	4		RS4	ADDD1
It. 1: MULTD F10, F0, F2	3	7	11	12	2	RAW F2, F0	RS2	MULT2
It. 1: ADDD F0, F12, F14	4	5	7	13	3		RS5	ADDD2
It. 2: MULTD F2, F6, F8	7	8			6		RS1	MULT1
It. 2: ADDD F0, F6, F4	8	9			0	Struct CDB	RS4	ADDD1
It. 2: MULTD F10, F0, F2	9				4	RAW F2, F0	RS3	
It. 2: ADDD F0, F12, F14						Struct ROB		

RAW **F2** I1-I3 RAW **F0** I2-I3



Exe Scoreboard



[Parallel operation in the control data 6600](#)

Recall: the Scoreboard pipeline

ISSUE	READ OPERAND	EXE COMPLETE	WB
Decode instruction;	Read operands;	Operate on operands;	Finish exec;
Structural FUs check; WAW checks	RAW check; WAR if need to read	Notify Scoreboard on completion;	WAR & Struct check (FUs will hold results); Can overlap issue/read&write 4 Structural Hazard;

Exe .1 Scoreboard: Code

```
I1:  lw $f1, 0($r0)
I2:  faddi $f1, $f1, C1
I3:  faddi $f2, $f1, C2
I4:  sw $f2, 0($r0)
I5:  lw $f2, 4($r0)
I6:  fadd $f2, $f2, $f2
I7:  sw $f2, 4($r0)
```


Exe .1 Scoreboard: Conflicts

I1: lw **\$f1**, 0(\$r0)
I2: faddi **\$f1**, **\$f1**, C1
I3: faddi **\$f2**, **\$f1**, C2
I4: sw **\$f2**, 0(\$r0)
I5: lw **\$f2**, 4(\$r0)
I6: fadd **\$f2**, **\$f2**, **\$f2**
I7: sw **\$f2**, 4(\$r0)

RAW **f1** I1-I2

RAW **f1** I2-I3

RAW **f2** I3-I4

RAW **f2** I5-I6

RAW **f2** I6-I7

WAW **f2** I5-I6

WAW **f2** I5-I3

WAW **f1** I2-I1

WAR **f2** I4-I5

WAR **f2** I4-I6

Exe .1 Scoreboard: Architecture

3 FPU, latency 2 cc

&&

4 LDU, latency 4 cc

```

I1:  lw $f1, 0($r0)
I2:  faddi $f1, $f1, C1
I3:  faddi $f2, $f1, C2
I4:  sw $f2, 0($r0)
I5:  lw $f2, 4($r0)
I6:  fadd $f2, $f2, $f2
I7:  sw $f2, 4($r0)
    
```

RAW **f1** I1-I2

RAW **f1** I2-I3

RAW **f2** I3-I4

RAW **f2** I5-I6

RAW **f2** I6-I7

WAW **f2** I5-I6

WAW **f2** I5-I3

WAW **f1** I2-I1

WAW **f2** I6-I3

WAR **f2** I4-I5

WAR **f2** I4-I6

Exe .1 Scoreboard: the Cache Misses

3 FPU, latency 2 cc

&&

4 LDU, latency 4 cc

I1: lw **\$f1**, 0(\$r0)
I2: faddi **\$f1**, **\$f1**, C1
I3: faddi **\$f2**, **\$f1**, C2
I4: sw **\$f2**, 0(\$r0)
I5: lw **\$f2**, 4(\$r0)
I6: fadd **\$f2**, **\$f2**, **\$f2**
I7: sw **\$f2**, 4(\$r0)

RAW **f1** I1-I2

RAW **f1** I2-I3

RAW **f2** I3-I4

RAW **f2** I5-I6

RAW **f2** I6-I7

WAW **f2** I5-I6

WAW **f2** I5-I3

WAW **f1** I2-I1

WAW **f2** I6-I3

WAR **f2** I4-I5

WAR **f2** I4-I6

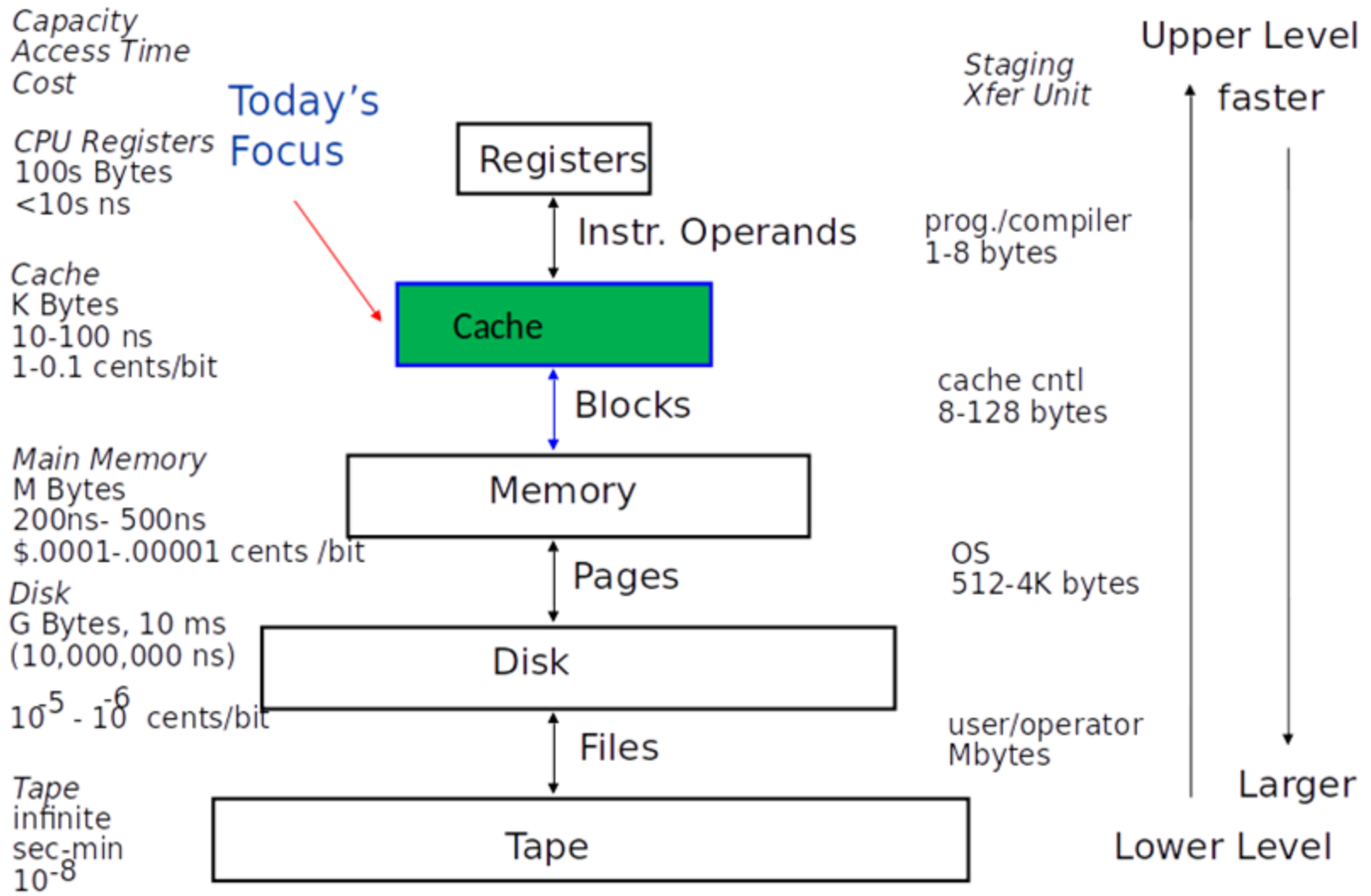
I1 cache miss

→ Penalty of 2 cc

I5 cache miss

→ Penalty of 5 cc

Recall: Memory hierarchy



Recall: Locality Principles

- Programs access a small proportion of their address space at any time
- **Temporal** locality
 - Items accessed recently are likely to be accessed again soon
 - e.g., instructions in a loop, induction variables
- **Spatial** locality
 - Items near those accessed recently are likely to be accessed soon
 - E.g., sequential instruction access, array data

Recall: Cache Algorithm reads

Look at Processor Address, search cache tags to find match. Then either

HIT - Found in Cache

Return copy
of data from
cache

Hit Rate = fraction of accesses found in cache

Miss Rate = $1 - \text{Hit rate}$

Hit Time = RAM access time +
time to determine HIT/MISS

Miss Time = time to replace block in cache +
time to deliver block to processor

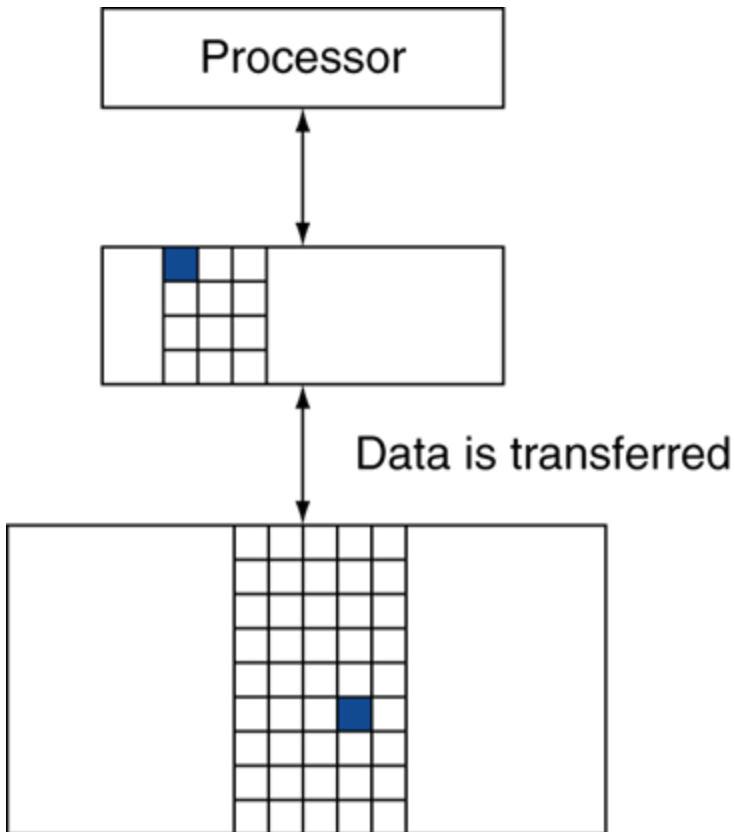
MISS - Not in cache

Read block of data
from Main Memory

Wait ...

Return data to
processor
and update cache

Recall: Hierarchy Principles



- Block (aka line): unit of copying
 - May be multiple words
- If accessed data is present in upper level
 - Hit: access satisfied by upper level
 - Hit ratio: hits/accesses
- If accessed data is absent
 - Miss: block copied from lower level
 - Time taken: miss penalty
 - Miss ratio: misses/accesses = 1 – hit ratio
 - Then accessed data supplied from upper level

$$\frac{\text{Misses}}{\text{Instruction}} = \frac{\text{Miss rate} \times \text{Memory accesses}}{\text{Instruction count}} = \text{Miss rate} \times \frac{\text{Memory accesses}}{\text{Instruction}}$$

$$\text{Average memory access time} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

Exe Scoreboard

3 FPU, latency 2 cc
4 LDU, latency 4 cc

Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Unit
I1:lw \$f1, 0(\$r0)					
I2:faddi \$f1, \$f1, c1					
I3:faddi \$f2, \$f1, c2					
I4:sw \$f2, 0(\$r0)					
I5:lw \$f2, 4(\$r0)					
I6:fadd \$f2, \$f2, \$f2					
I7:sw \$f2, 4(\$r0)					

I1 cache miss

→ Penalty of 2 cc

I5 cache miss

→ Penalty 5 cc

RAW **f1** I1-I2

RAW **f1** I2-I3

RAW **f2** I3-I4

RAW **f2** I5-I6

RAW **f2** I6-I7

WAW **f2** I5-I6

WAW **f2** I5-I3

WAW **f1** I2-I1

WAR **f2** I4-I5

WAR **f2** I4-I6

Exe Scoreboard

3 FPU, latency 2 cc
4 LDU, latency 4 cc

Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Unit
I1:lw \$f1, 0(\$r0)	1	2	8	9	LDU1
I2:faddi \$f1, \$f1, c1					
I3:faddi \$f2, \$f1, c2					
I4:sw \$f2, 0(\$r0)					
I5:lw \$f2, 4(\$r0)					
I6:fadd \$f2, \$f2, \$f2					
I7:sw \$f2, 4(\$r0)					

I1 cache miss
 → Penalty of 2 cc
 I5 cache miss
 → Penalty 5 cc

RAW f1 I1-I2
 RAW f1 I2-I3
 RAW f2 I3-I4
 RAW f2 I5-I6

RAW f2 I6-I7
 WAW f2 I5-I6
 WAW f2 I5-I3
 WAW f1 I2-I1

WAR f2 I4-I5
 WAR f2 I4-I6

Exe Scoreboard

3 FPU, latency 2 cc
4 LDU, latency 4 cc

Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Unit
I1:lw \$f1, 0(\$r0)	1	2	8	9	LDU1
I2:faddi \$f1, \$f1, c1	10	11	13	14	FPU1
I3:faddi \$f2, \$f1, c2					
I4:sw \$f2, 0(\$r0)					
I5:lw \$f2, 4(\$r0)					
I6:fadd \$f2, \$f2, \$f2					
I7:sw \$f2, 4(\$r0)					

~~I1 cache miss~~

~~→ Penalty of 2 cc~~

I5 cache miss

→ Penalty 5 cc

~~RAW f1 I1-I2~~

RAW f1 I2-I3

RAW f2 I3-I4

RAW f2 I5-I6

RAW f2 I6-I7

WAW f2 I5-I6

WAW f2 I5-I3

WAW f1 I2-I1

WAR f2 I4-I5

WAR f2 I4-I6

Exe Scoreboard

3 FPU, latency 2 cc
4 LDU, latency 4 cc

Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Unit
I1:lw \$f1, 0(\$r0)	1	2	8	9	LDU1
I2:faddi \$f1, \$f1, c1	10	11	13	14	FPU1
I3:faddi \$f2, \$f1, c2	11	15	17	18	FPU2
I4:sw \$f2, 0(\$r0)					
I5:lw \$f2, 4(\$r0)					
I6:fadd \$f2, \$f2, \$f2					
I7:sw \$f2, 4(\$r0)					

~~I1 cache miss~~

~~→ Penalty of 2 cc~~

I5 cache miss

→ Penalty 5 cc

~~RAW f1 I1-I2~~

RAW f1 I2-I3

RAW f2 I3-I4

RAW f2 I5-I6

RAW f2 I6-I7

WAW f2 I5-I6

WAW f2 I5-I3

~~WAW f1 I2-I1~~

WAR f2 I4-I5

WAR f2 I4-I6

Exe Scoreboard

3 FPU, latency 2 cc
4 LDU, latency 4 cc

Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Unit
I1:lw \$f1, 0(\$r0)	1	2	8	9	LDU1
I2:faddi \$f1, \$f1, c1	10	11	13	14	FPU1
I3:faddi \$f2, \$f1, c2	11	15	17	18	FPU2
I4:sw \$f2, 0(\$r0)	12	19	23	24	LDU2
I5:lw \$f2, 4(\$r0)					
I6:fadd \$f2, \$f2, \$f2					
I7:sw \$f2, 4(\$r0)					

~~I1 cache miss~~

~~→ Penalty of 2 cc~~

I5 cache miss

→ Penalty 5 cc

~~RAW f1 I1-I2~~

~~RAW f1 I2-I3~~

RAW f2 I3-I4

RAW f2 I5-I6

RAW f2 I6-I7

WAW f2 I5-I6

WAW f2 I5-I3

~~WAW f1 I2-I1~~

WAR f2 I4-I5

WAR f2 I4-I6

Exe Scoreboard

3 FPU, latency 2 cc
4 LDU, latency 4 cc

Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Unit
I1:lw \$f1, 0(\$r0)	1	2	8	9	LDU1
I2:faddi \$f1, \$f1, c1	10	11	13	14	FPU1
I3:faddi \$f2, \$f1, c2	11	15	17	18	FPU2
I4:sw \$f2, 0(\$r0)	12	19	23	24	LDU2
I5:lw \$f2, 4(\$r0)	19	20	29	30	LDU3
I6:fadd \$f2, \$f2, \$f2					
I7:sw \$f2, 4(\$r0)					

~~I1 cache miss~~

~~→ Penalty of 2 cc~~

I5 cache miss

→ Penalty 5 cc

~~RAW f1 I1-I2~~

~~RAW f1 I2-I3~~

~~RAW f2 I3-I4~~

RAW f2 I5-I6

RAW f2 I6-I7

WAW f2 I5-I6

~~WAW f2 I5-I3~~

~~WAW f1 I2-I1~~

~~WAR f2 I4-I5~~

WAR f2 I4-I6

Exe Scoreboard

3 FPU, latency 2 cc
4 LDU, latency 4 cc

Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Unit
I1:lw \$f1, 0(\$r0)	1	2	8	9	LDU1
I2:faddi \$f1, \$f1, c1	10	11	13	14	FPU1
I3:faddi \$f2, \$f1, c2	11	15	17	18	FPU2
I4:sw \$f2, 0(\$r0)	12	19	23	24	LDU2
I5:lw \$f2, 4(\$r0)	19	20	29	30	LDU3
I6:fadd \$f2, \$f2, \$f2	31	32	34	35	FPU3
I7:sw \$f2, 4(\$r0)					

~~I1 cache miss~~

~~→ Penalty of 2 cc~~

~~I5 cache miss~~

~~→ Penalty 5 cc~~

~~RAW f1 I1-I2~~

~~RAW f1 I2-I3~~

~~RAW f2 I3-I4~~

RAW f2 I5-I6

RAW f2 I6-I7

WAW f2 I5-I6

~~WAW f2 I5-I3~~

~~WAW f1 I2-I1~~

~~WAR f2 I4-I5~~

~~WAR f2 I4-I6~~

Exe Scoreboard

3 FPU, latency 2 cc
4 LDU, latency 4 cc

Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Unit
I1:lw \$f1, 0(\$r0)	1	2	8	9	LDU1
I2:faddi \$f1, \$f1, c1	10	11	13	14	FPU1
I3:faddi \$f2, \$f1, c2	11	15	17	18	FPU2
I4:sw \$f2, 0(\$r0)	12	19	23	24	LDU2
I5:lw \$f2, 4(\$r0)	19	20	29	30	LDU3
I6:fadd \$f2, \$f2, \$f2	31	32	34	35	FPU3
I7:sw \$f2, 4(\$r0)	32	36	40	41	LDU4

~~I1 cache miss~~

~~→ Penalty of 2 cc~~

~~I5 cache miss~~

~~→ Penalty 5 cc~~

~~RAW f1 I1-I2~~

~~RAW f1 I2-I3~~

~~RAW f2 I3-I4~~

~~RAW f2 I5-I6~~

~~RAW f2 I6-I7~~

~~WAW f2 I5-I6~~

~~WAW f2 I5-I3~~

~~WAW f1 I2-I1~~

~~WAR f2 I4-I5~~

~~WAR f2 I4-I6~~



Recall: MESI Protocol

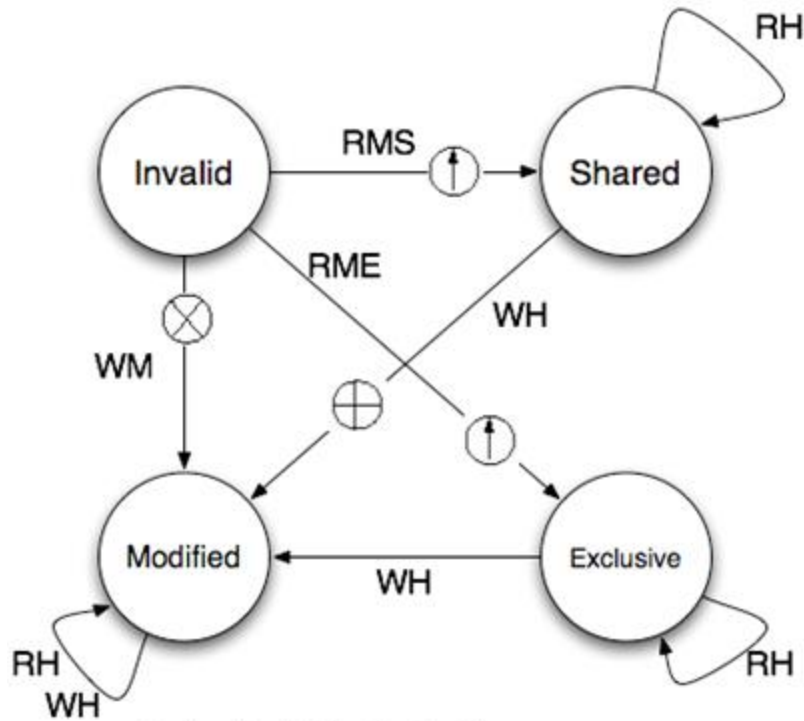
MESI Protocol: Write-Invalidate

Each cache block can be in one of **four** states:

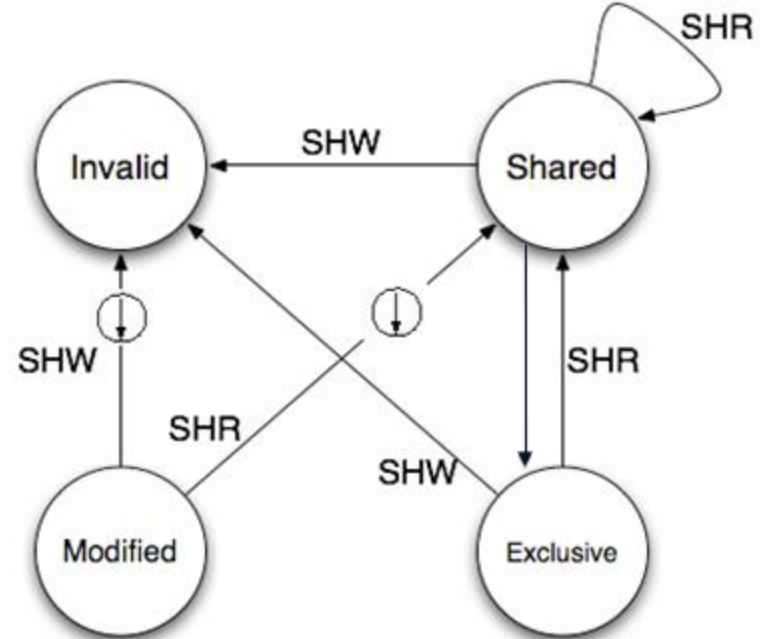
- **Modified**: the block is dirty and cannot be shared; cache has only copy, its writeable.
- **Exclusive**: the block is clean and cache has only copy;
- **Shared**: the block is clean and other copies of the block are in cache;
- **Invalid**: block contains no valid data

Add exclusive state to distinguish exclusive (writable) and owned (written)

Recall: MESI State Transition Diagram







Cache line in the "acting" processor



Transaction due to events snooped on the common BUS

BUS Transactions

RH = Read Hit
RMS = Read Miss, Shared
RME = Read Miss, Exclusive
WH = Write Hit
WM = Write Miss
SHR = Snoop Hit on a Read
SHW = Snoop Hit on a Write or
Read-with-Intent-to-Modify

-  = Snoop Push
-  = Invalidate Transaction
-  = Read-with-Intent-to-Modify
-  = Cache Block Fill

Exe1: Cache Coherency

Consider the following access pattern on a two-processor system with a direct-mapped, write-back cache with **one cache block and a two cache block memory**.

Assume the MESI protocol is used, with **write-back caches, write-allocate, and invalidation** of other caches on write (instead of updating the value in the other caches).

Exe 1: Cache Coherency

Cycle	After Operation	P0 cache block state	P1 cache block state	Memory at block 0 up to date?	Memory at block 1 up to date?
0	P0: read block 1	Exclusive (1)	Invalid	Yes	Yes
1	P1: read block 0				
2	P0: write block 1				
3	P1: write block 0				
4	P1: read block 0				
5	P1: write block 0				
6	P0: read block 0				
7	P0: read block 0				
8	P0: write block 0				
9	P1: write block 1				

Exe 1: Cache Coherency

Cycle	After Operation	P0 cache block state	P1 cache block state	Memory at block 0 up to date?	Memory at block 1 up to date?
0	P0: read block 1	Exclusive (1)	Invalid	Yes	Yes
1	P1: read block 0	Exclusive (1)	Exclusive (0)	Yes	Yes
2	P0: write block 1	Modified (1)	Exclusive (0)	Yes	No
3	P1: write block 0	Modified (1)	Modified (0)	No	No
4	P1: read block 0	Modified (1)	Modified (0)	No	No
5	P1: write block 0	Modified (1)	Modified (0)	No	No
6	P0: read block 0	Shared (0)	Shared (0)	Yes	Yes
7	P0: read block 0	Shared (0)	Shared (0)	Yes	Yes
8	P0: write block 0	Modified (0)	Invalid	No	Yes
9	P1: write block 1	Modified (0)	Modified (1)	No	No

Exe 1: Cache Coherency

Cycle	After Operation	P0 cache block state	P1 cache block state	Memory at block 0 up to date?	Memory at block 1 up to date?
0	P0: read block 1	Exclusive (1)	Invalid	Yes	Yes
1	P1: read block 0	Exclusive (1)	Exclusive (0)	Yes	Yes
2	P0: write block 1	Modified (1)	Exclusive (0)	Yes	No
3	P1: write block 0	Modified (1)	Modified (0)	No	No
4	P1: read block 0	Modified (1)	Modified (0)	No	No
5	P1: write block 0	Modified (1)	Modified (0)	No	No
6	P0: read block 0	Shared (0)	Shared (0)	Yes	Yes
7	P0: read block 0	Shared (0)	Shared (0)	Yes	Yes
8	P0: write block 0	Modified (0)	Invalid	No	Yes
9	P1: write block 1	Modified (0)	Modified (1)	No	No

What if write non allocate?



Thank you for your attention

Questions?

Alessandro Verosimile <alessandro.verosimile@polimi.it>

Acknowledgements

Davide Conficconi, E. Del Sozzo, Marco D. Santambrogio, D. Sciuto

Part of this material comes from:

- “Computer Organization and Design” and “Computer Architecture A Quantitative Approach” Patterson and Hennessy books
- News and paper cited throughout the lecture

and are **properties of their respective owners**