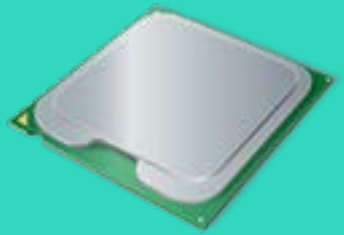
Department of Electrical & Computer Engineering
COE608 Computer Organization and Architecture

Ryerson University

Final Exam Tutorial



By: Muhammad Obaidullah

Overview

Components of a CPU

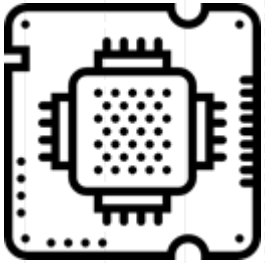
MIPS ISA

Converting C Code to MIPS Assembly and Vice Versa

Pipelining

Pipelining Hazards and Solutions

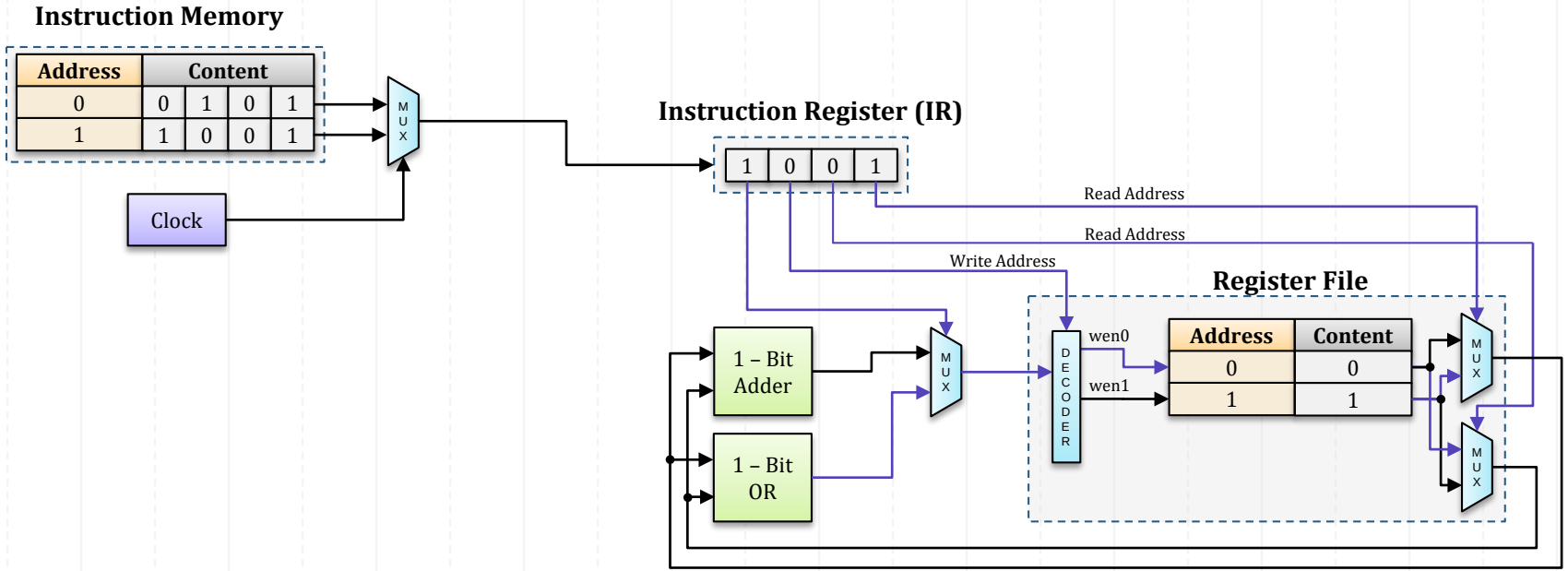
Final Exam Solution



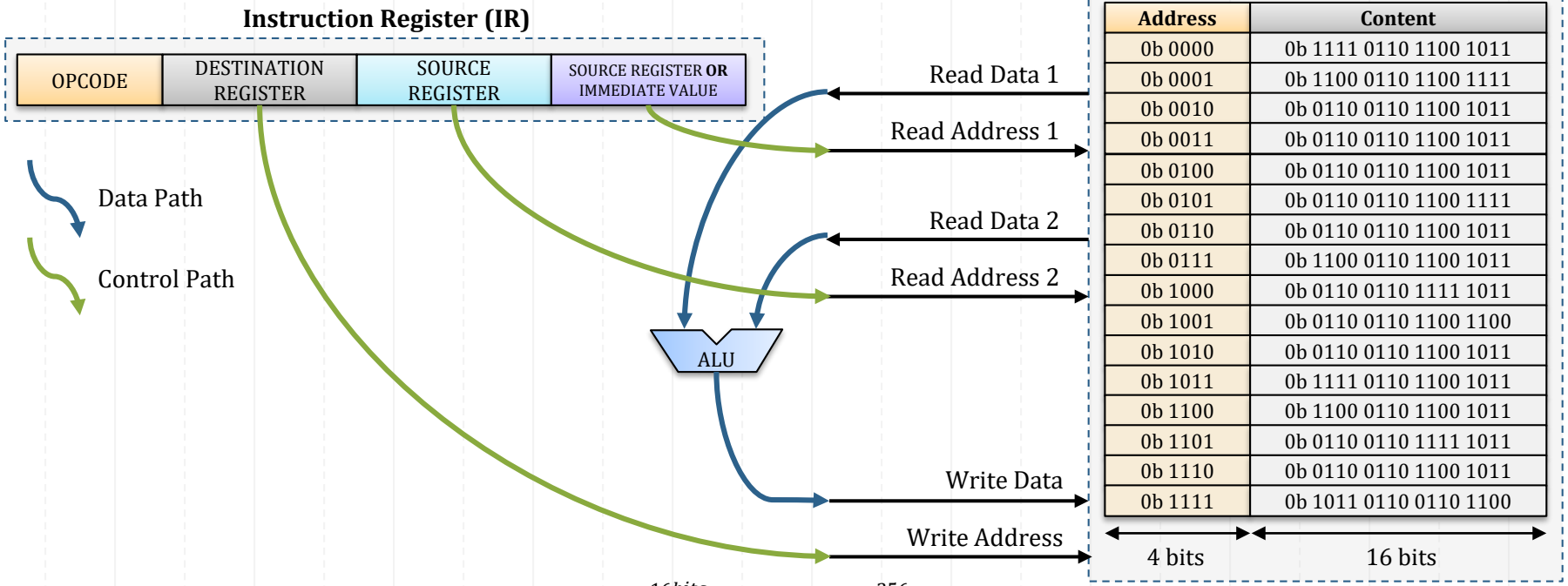
Components of a CPU

1

A Simple 1-bit CPU



Register File

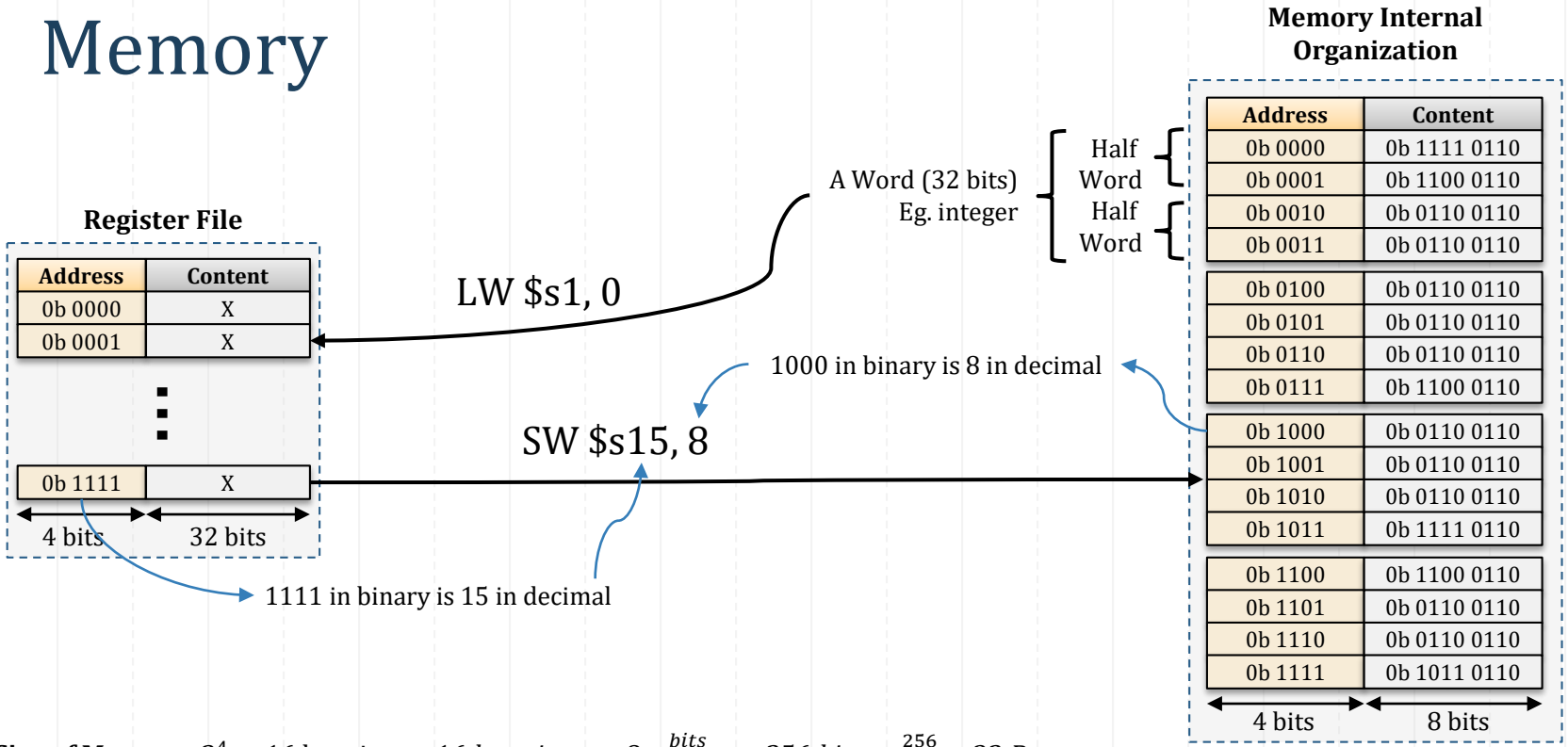


Address	Content
0b 0000	0b 1111 0110 1100 1011
0b 0001	0b 1100 0110 1100 1111
0b 0010	0b 0110 0110 1100 1011
0b 0011	0b 0110 0110 1100 1011
0b 0100	0b 0110 0110 1100 1011
0b 0101	0b 0110 0110 1100 1111
0b 0110	0b 0110 0110 1100 1011
0b 0111	0b 1100 0110 1100 1011
0b 1000	0b 0110 0110 1111 1011
0b 1001	0b 0110 0110 1100 1100
0b 1010	0b 0110 0110 1100 1011
0b 1011	0b 1111 0110 1100 1011
0b 1100	0b 1100 0110 1100 1011
0b 1101	0b 0110 0110 1111 1011
0b 1110	0b 0110 0110 1100 1011
0b 1111	0b 1011 0110 0110 1100

Size of Register File: $2^4 = 16$ location = 16 $locations \times \frac{16bits}{location} = 256\ bits = \frac{256}{8} = 32\ bytes$

Size of operands: 16 bits. This is a 16-bit CPU.

Memory



Size of Memory: $2^4 = 16$ locations = $16\ locations \times 8 \frac{bits}{location} = 256\ bits = \frac{256}{8} = 32\ Bytes$

MIPS ISA 2

Instruction Types

1. Register Type (R-Type)

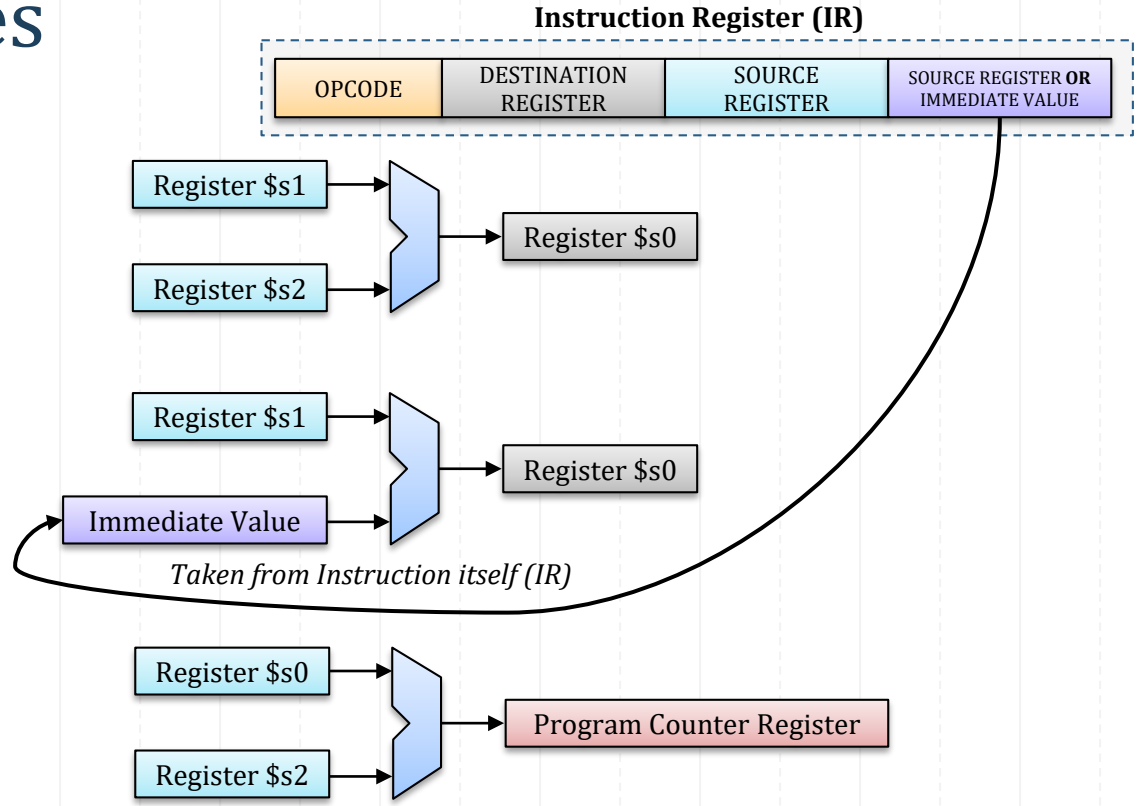
ADD	$s0	$s1	$s2

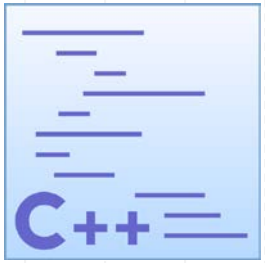
2. Immediate Type (I-Type)

ADDI	$s0	$s1	I

3. Jump Type (J-Type)

BNE	$s0	$s1	JUMP

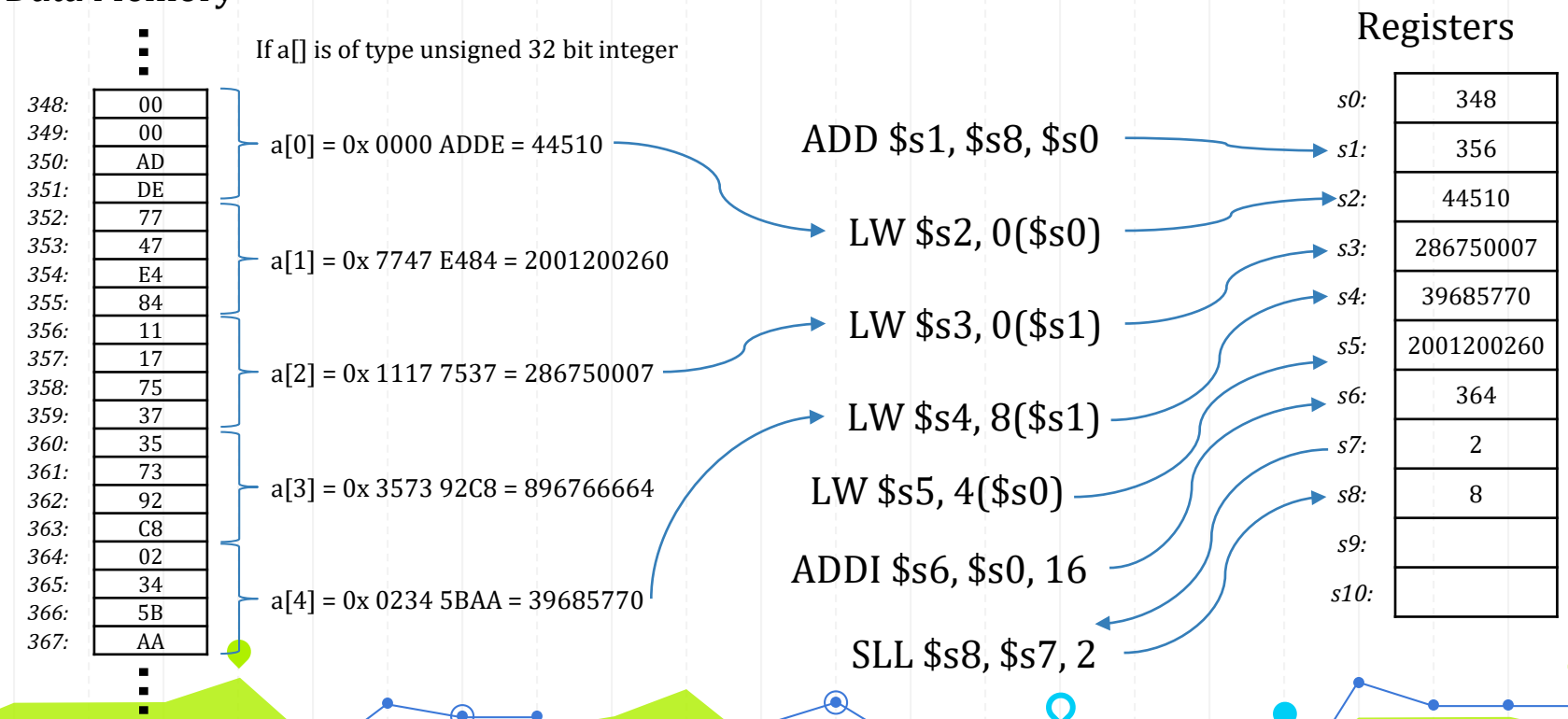




Converting C Code to MIPS Assembly and Vice Versa

3

Storing an Array in Memory



C to MIPS

C Code

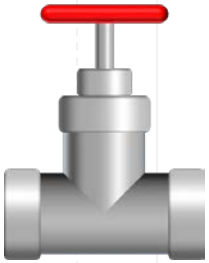
1:	for (int i = 0; I < 50; i++) {
2:	a[i] = a[i] + b [i];
3:	}

MIPS Assembly Code

	1:	ADDI	$s2	0	0
LOOP:	2:	SLL	$s3	$s2	2
	3:	ADD	$s4	$s0	$s3
	4:	ADD	$s5	$s1	$s3
	5:	LW	$s6	0	($s4)
	6:	LW	$s7	0	($s5)
	7:	ADD	$s8	$s7	$s6
	8:	SW	$s8	0	($s4)
	9:	ADDI	$s2	$s2	1
	10:	BEQ	$s2	$s9	EXIT
	11:	JUMP	LOOP	$s	$s
EXIT:	12:				

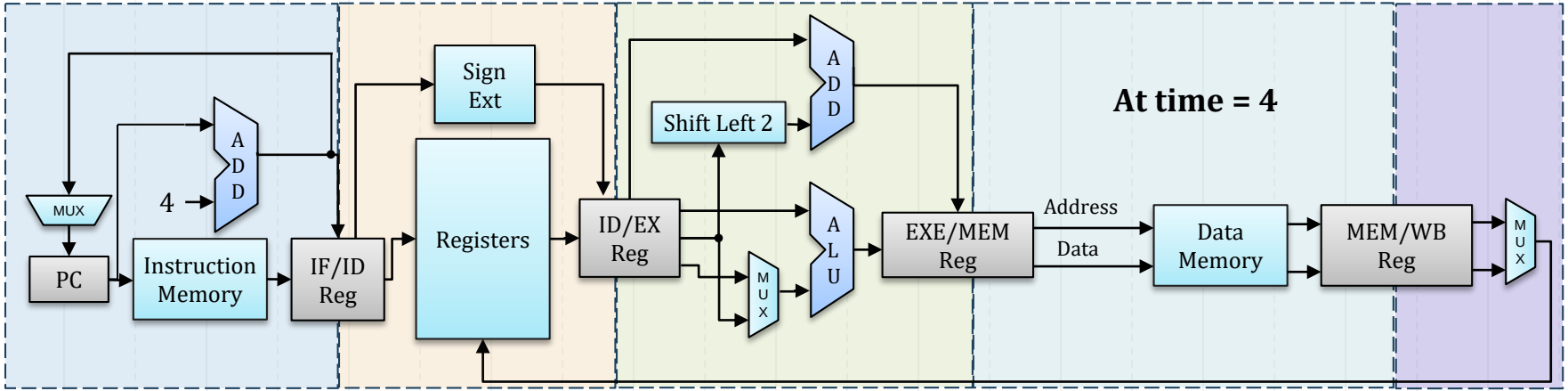
Registers

s0:	&a[0]
s1:	&b[0]
s2:	i
s3:	4i
s4:	&a[i]
s5:	&b[i]
s6:	a[i]
s7:	b[i]
s8:	a[i] + b[i]
s9:	50
s10:	



Pipelining 4

Pipelined CPU



99:	ADD	$s3	$s0	$s4
100:	ADD	$s5	$s6	$s7
101:	SUB	$s3	$s0	$s4
102:	MUL	$s5	$s6	$s7

time = 1	*time = 2*	*time = 3*	*time = 4*	*time = 5*	*time = 6*
IF	ID	EXE	MEM	WB	
	IF	ID	EXE	MEM	WB
		IF	ID	EXE	MEM	WB
			IF	ID	EXE	MEM	WB

Pipelining Hazards and Solutions 5

Data Hazards

Problems that occur when data is not correct

1. Read After Write (RAW)

1:	**ADD**	**$s0**	**$s1**	**$s2**
2:	**ADD**	**$s3**	**$s0**	**$s4**

Problem: *Instruction 2 could read register 0 before the instruction 1 has written back*
Solution: *1) Add un-related instructions between instructions 1 and 2.*
2) Forwarding (The value output from ALU can be directly fed back to input of ALU)

2. Write after Read (WAR)

1:	**MUL**	**$s0**	**$s1**	**$s2**
2:	**ADD**	**$s1**	**$s3**	**$s4**

Problem: *Instruction 2 could write register 1 before the instruction 0 has read*
Solution: *Register Renaming*

3. Write after Write (WAW)

1:	**ADD**	**$s0**	**$s1**	**$s2**
2:	**ADD**	**$s0**	**$s3**	**$s4**

Problem: *Register 0 is written again without the value in it being used*
Solution: *Register Renaming*

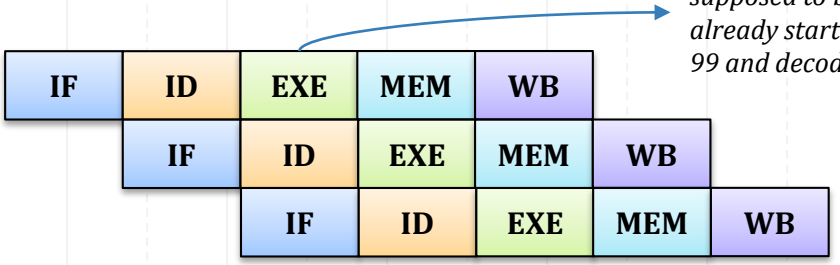
Control Hazards

Problems that occur on the instruction flow

Conditional Branch

Problem: *Don't know which instruction should be executed after instruction 98 (Instruction 99 or 156 ?)*

97:	SUB	$s8	$s9	$s10
98:	BNE	$s0	$s1	156
99:	ADD	$s3	$s0	$s4
100:	ADD	$s5	$s6	$s7
	⋮			
156:	ADD	$s0	$s1	$s2



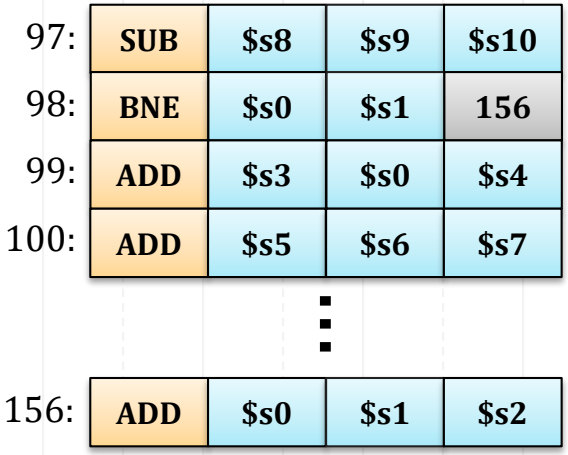
Found out that next instruction is supposed to be 156 but you have already started executing instruction 99 and decoding instruction 100

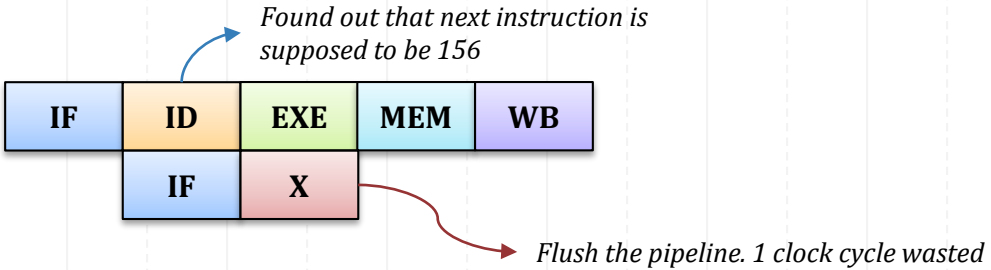
Control Hazards

Problems that occur on the instruction flow

Solution 1: Fast Branches

Change hardware so that next branch to take is calculated in ID pipeline stage rather than EXE



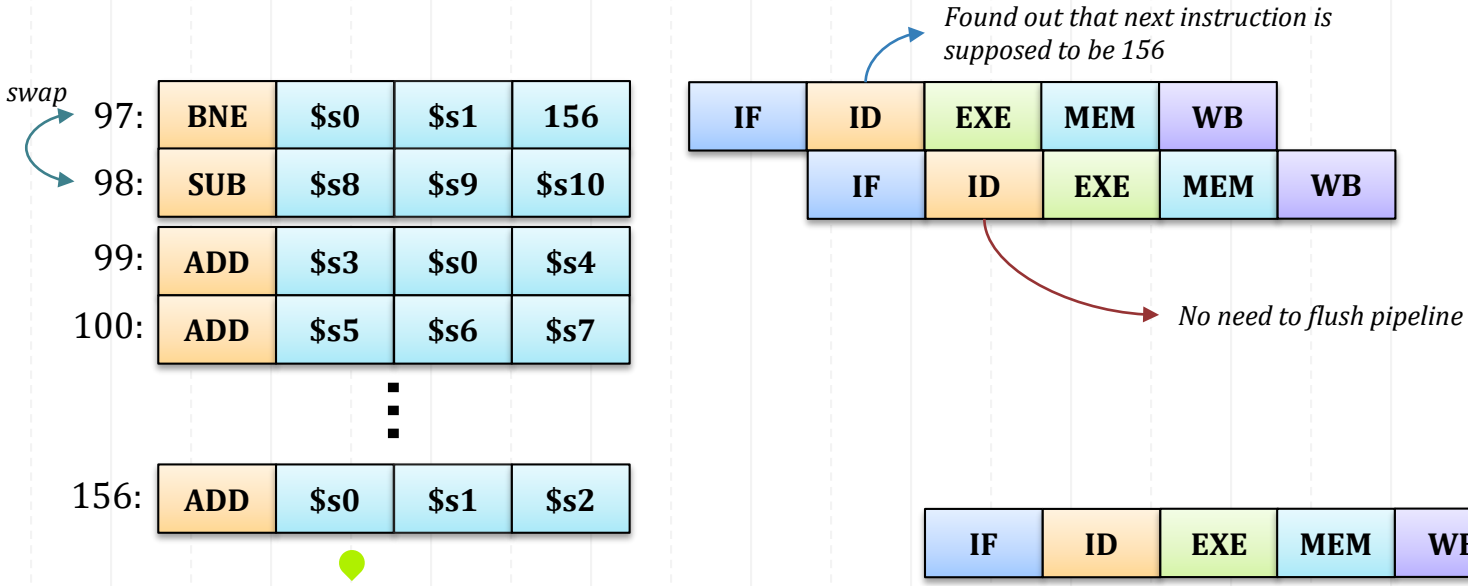


Control Hazards

Problems that occur on the instruction flow

Solution 2: Delayed Branches

Put an un-related instruction after the branch instruction. This instruction gets executed weather the branch is taken or not

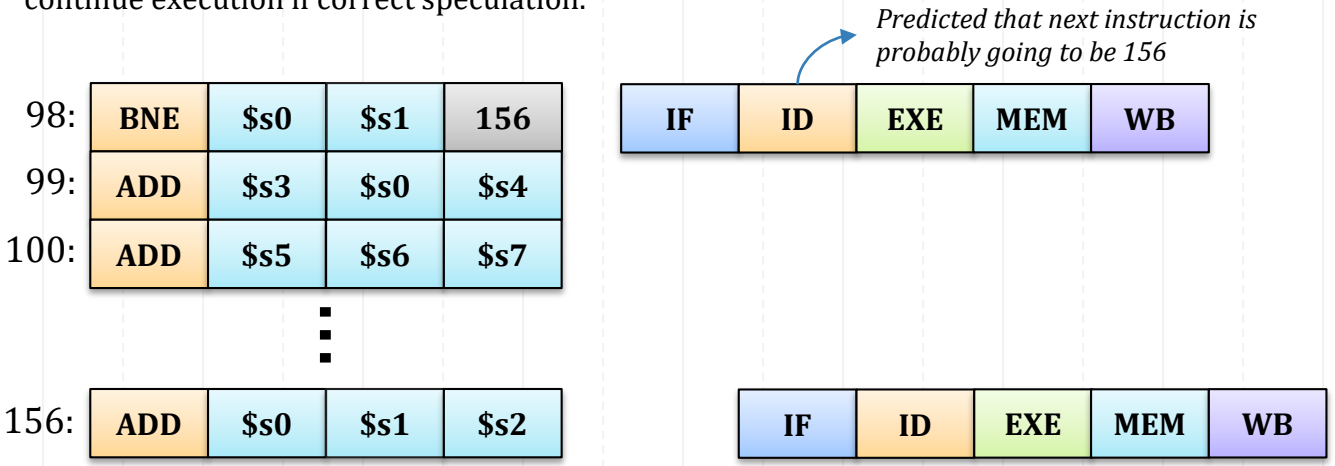


Control Hazards

Problems that occur on the instruction flow

Solution 3: Speculative Execution

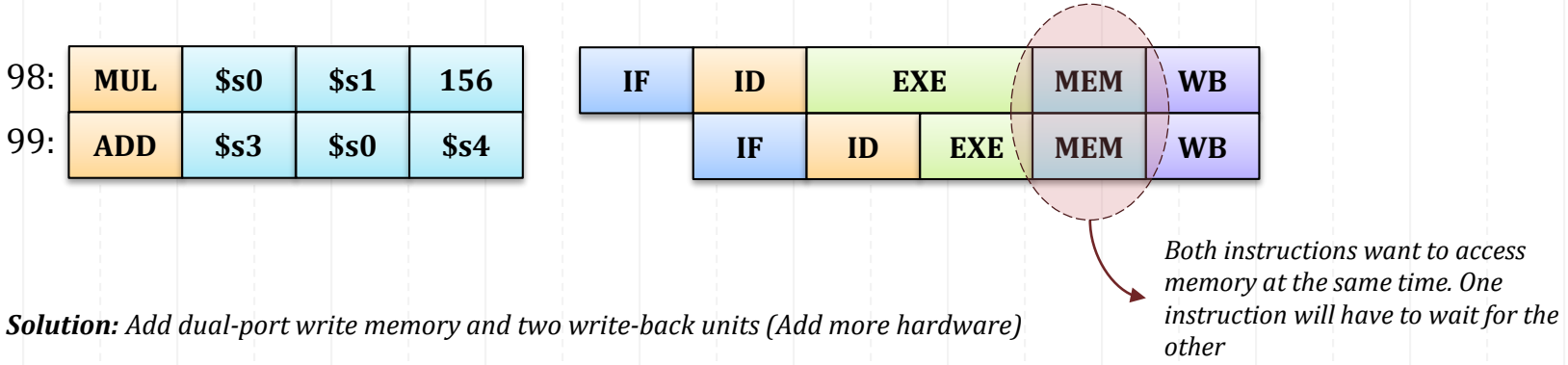
Predict weather the branch is going to be taken or not (usually based on probability). Flush pipeline if wrong speculation and continue execution if correct speculation.



Structural Hazards

Problems that occur because of hardware

Problem: *If Multiply takes 2 clock cycles to execute while addition takes 1 clock cycle to execute, then addition instruction after multiply instruction will cause both instructions wanting to access memory at the same time!*



Solution: *Add dual-port write memory and two write-back units (Add more hardware)*

Final Exam Solution 6

THANKS!

Any questions?

You can find me at

ENG 402 / mobaidullah@ryerson.ca