INTRODUCTION TO COMPUTER ORGANIZATION

CCIT4026

Chapter 9: Multiple Cycle Processor

Outline

- Single-cycle Processor
- Multi-cycle Processor
- Control Signal of Processor

Single-cycle Processor

- We have introduced how to combine the instructions into a **single datapath**, by devising ways to share some of the resources (e.g., ALU) among the different instructions instead of duplicating them.

- We also discussed how to control and select the operation of the datapath by generating appropriate control signals (e.g. read/write signals for memory/registers, selection input for muxes, ALU control inputs)

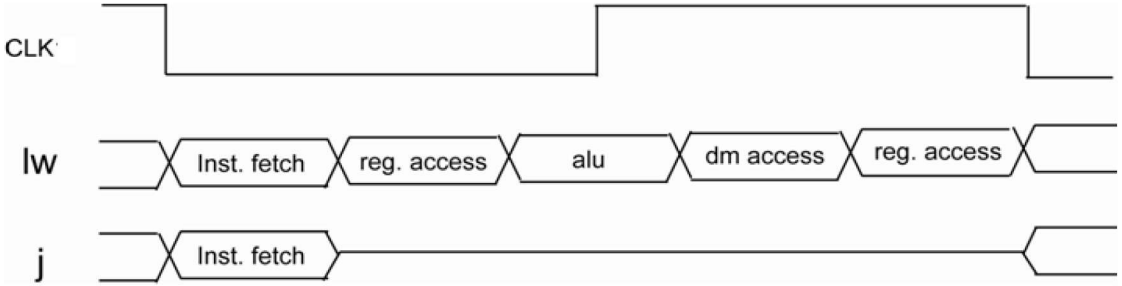
Single-cycle Processor

- Every instruction takes one clock cycle (CPI = 1).
  - The clock cycle is determined by the longest path
- CPU time = instruction count x CPI x clock period
- Even though CPI = 1
  - The number of clock cycle is expected to be huge, CPU performance is low
- **lw** uses the longest path
  - Five functional units in series

Instruction class	Functional units used by the instruction class				
R-format	Instruction fetch	Register access	ALU	Register access	
Load word	Instruction fetch	Register access	ALU	Memory access	Register access
Store word	Instruction fetch	Register access	ALU	Memory access	
Branch	Instruction fetch	Register access	ALU		
Jump	Instruction fetch				

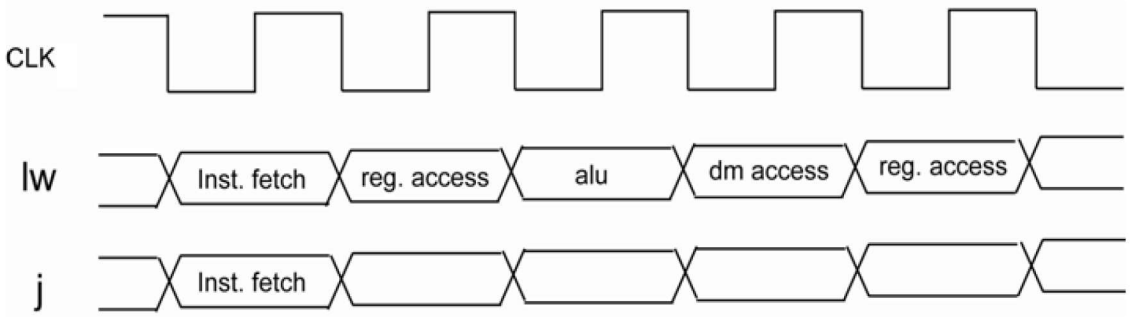
Single-cycle vs Multi-cycle

Single-cycle implementation



- Cycle time depends on the most consuming instruction.

Multi-cycle implementation



- Faster clock
- Adopt different number of clock cycles per instruction
- Reduce physical resources

Example

Instruction class	Functional units used by the instruction class				
R-format	Instruction fetch	Register access	ALU	Register access	
Load word	Instruction fetch	Register access	ALU	Memory access	Register access
Store word	Instruction fetch	Register access	ALU	Memory access	
Branch	Instruction fetch	Register access	ALU		
Jump	Instruction fetch				

Instruction class	Instruction memory	Register read	ALU operation	Data memory	Register write	Total
R-format	2	1	2	0	1	6 ns
Load word	2	1	2	2	1	8 ns
Store word	2	1	2	2		7 ns
Branch	2	1	2			5 ns
Jump	2					2 ns

Assume that the instruction used in a program following the distribution below:

R-Type 20%
LW 10%
SW 20%
Branch 25%
Jump 25%

How long do I need to spend for each instruction?

	Average execution time per instruction (ns)	Clock rate (Hz)
Single cycle		
Multiple cycle		

Multi-cycle Processor

- The execution of each instruction is broken into a series of steps that correspond to the **functional unit operations**. Each step takes one clock cycle to complete.

- **Each functional unit can be reused per instruction**, as long as it is used on different clock cycles. This sharing can help to reduce the amount of hardware required (lower cost).
  - **One memory unit** is used for both instructions and data.
  - **One ALU** is used, instead of one ALU and two adders.
  - **Additional registers are added** after every major functional unit to hold the output of that unit until the value is used in a subsequent clock cycle.

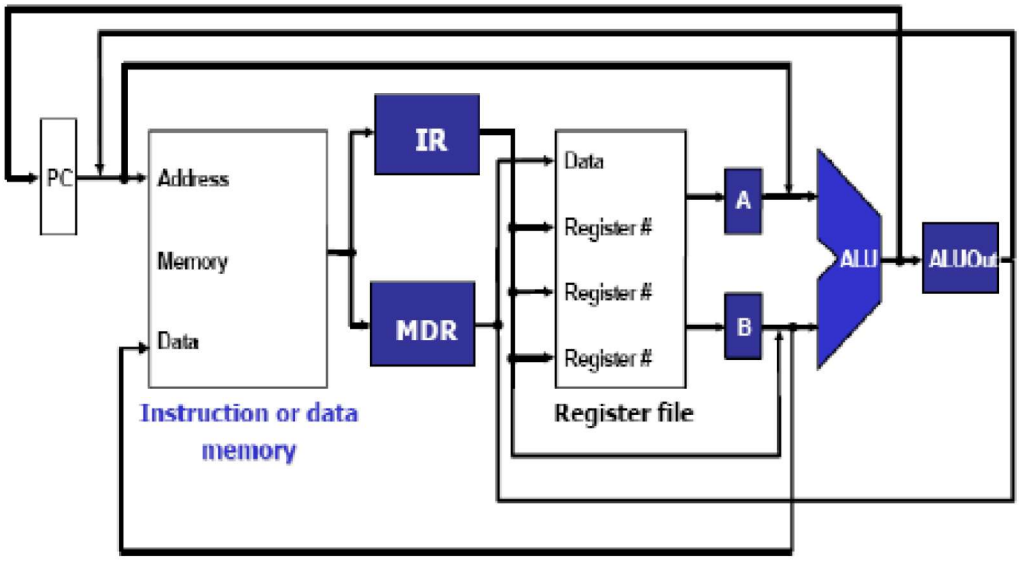
Multi-cycle Processor

- We know what steps each instruction takes
  - Datapath needs to consider all the possibilities for all instructions
  - Control unit needs to set all the control signals for different situations, and the controls will be generated by Finite State Machine (**FSM**) in each cycle (not solely determined by the machine instruction bits).

- One CPU step takes one clock cycle
  - Step 1: Instruction Fetch
  - Step 2: Instruction Decode and Register Read
  - Step 3: Execution, Memory Address Computation, or Branch Completion
  - Step 4: Memory Access or R-type instruction completion
  - Step 5: Write-back step

Multi-cycle Processor

Because of the unit sharing, 5 more registers are needed:

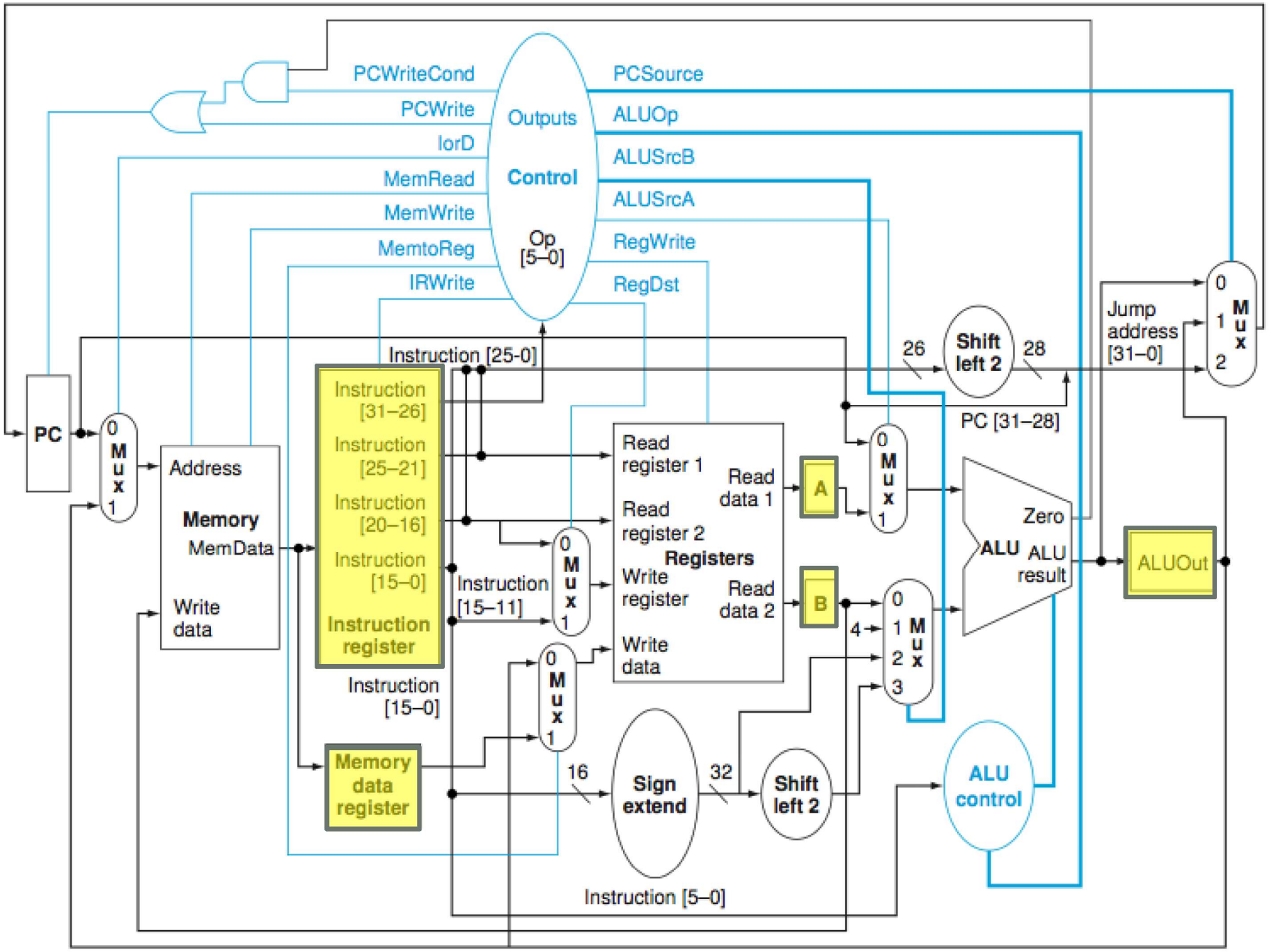


(1) IR – Instruction Register
-- to hold the instruction from memory

(2) MDR – Memory Data Register
-- to hold data from memory

(3) A,B – to hold the operand value from register file

(4) ALUOut – to hold the output of the ALU


PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Outputs
Control
Op
[5–0]
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
Instruction [25-0]
26
Shift
left 2
28
PC [31–28]
Jump
address
[31–0]
M
u
x
0
1
2
PC
Address
Memory
MemData
Write
data
Instruction
[31–26]
Instruction
[25–21]
Instruction
[20–16]
Instruction
[15–0]
Instruction
register
Instruction
[15–0]
Instruction
[15–11]
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
A
B
4
Zero
ALU
ALU
result
ALUOut
Memory
data
register
16
Sign
extend
32
Shift
left 2
ALU
control
Instruction [5–0]

Multi-cycle Processor

	Actions for instructions			
Step name	**R-type**	**Memory references**	**Branches**	**Jumps**
Instruction fetch	IR = Memory[PC] PC = PC + 4			
Instruction decode / register fetch	A = Reg[IR[25-21]] B = Reg[IR[20-16]] ALUOut = PC + (sign-extend(IR[15-0]) << 2)			
Execution, addr comp., branch/jump completion	ALUOut = A op B	ALUOut = A + sign-extend(IR[15-0])	If (A == B) PC = ALUOut	PC = PC[31-28] || (IR[25-0] << 2)
Memory access, R-type completion	Reg[IR[15-11]] = ALUOut	Load: MDR=Memory[ALUOut]		
		Store: Memory[ALUOut] = B		
Memory read completion		Load: Reg[IR[20-16]] = MDR		
Total steps	**4**	**load: 5 / store: 4**	**3**	**3**

Multi-cycle Processor

The following instruction sequence runs on the multi-cycle. Please find the data on every bus during the next 11 clock cycles.

Instruction sequence:

L1:	sw	$t1, 5($s0)	101011 10000 01001 0000 0000 0000 0101
	sub	$t0, $s1, $s2	000000 10001 10010 01000 00000 100010
	beq	$t0, $t1, L1	000100 01000 01001 1111 1111 1111 1101

Assume the registers and memory have following contents before execution:

$s0 = 100, $s1 = 5, $s2 = 3, $t1 = 2, L1 = 400, mem[105] = 6

Multi-cycle Processor

	L1: sw $t1, 5($s0)				sub $t0, $s1, $s2				beq $t0, $t1, L1		
Clock no.	1	2	3	4	5	6	7	8	9	10	11
cycle/inst	1	2	3	4	1	2	3	4	1	2	3
PC		PC= PC+4			PC	PC= PC+4			PC	PC= PC+4	PC= ALUout if (A==B)
Memory	Memory[PC]			B-> Memory[ALUout]	Memory[PC]				Memory[PC]		
Instruction Register (IR)	Instruction Fetch IR = Memory[PC]				Instruction Fetch IR = Memory[PC]				Instruction Fetch IR = Memory[PC]		
Register File (A & B)		A = $s0 B = $t1				A = $s1 B = $s2		$t0 = ALUout		A = $t0 B = $t1	
ALU output (ALUout)	Calculate PC + 4	Branch address pre-calculate	ALUout = A + 5		Calculate PC + 4	Branch address pre-calculate	ALUout = A - B		Calculate PC + 4	Branch address pre-calculate	

Refer to the Lecture Notes for Multi-Cycle Processor Tasks

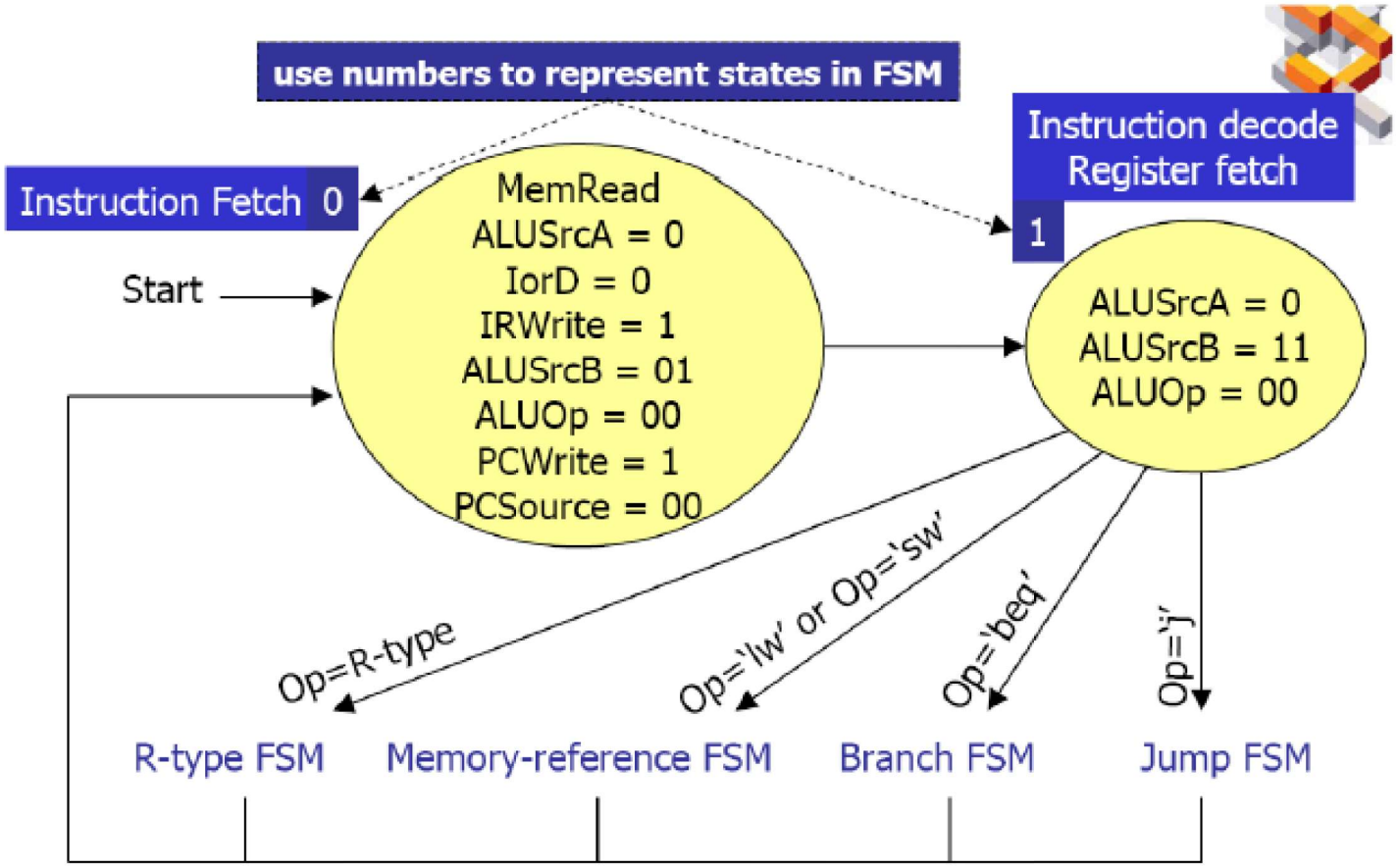
Control Signal in Multi-cycle Processor

- Unlike the single-cycle implementation which only requires a set of truth tables to specify the setting of the control signals based on the instruction class, control for the multi-cycle implementation is more complex.

- Value of control signals is dependent what instruction is being executed and which step is being performed

- **Finite State Machine** (FSM) is used to determine the control signals.

Control Signal in Multi-cycle Processor



Control Signal in Multi-cycle Processor