



Chapter 3: Computer Organization Fundamentals

Prof. Ben Lee
Oregon State University
School of Electrical Engineering and Computer Science

1

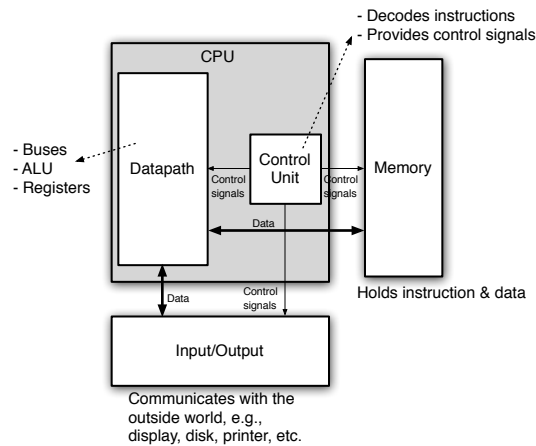


Chapter Goals

- Understand the organization of a computer system and its components.
- Understand how assembly instructions are executed on the processor.

2

Computer Organization



Ch. 3: Computer Organization Fundamentals

3

3

Memory

- Random Access Memory
- Holds instructions (program) and data
 - Unified
 - Separate instruction and data memory
- Organized into consecutive addressable memory words.
- 1 memory word
 - Memory data size
 - Size of the information accessed by the CPU (CPU register size)
 - Manufacturer's definition

Ch. 3: Computer Organization Fundamentals

4

4

Registers

Some important registers:

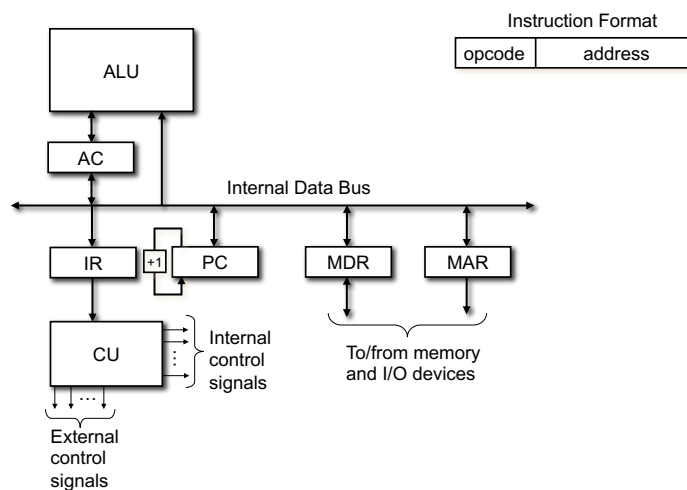
- **PC (Program Counter)** – holds the address of the next inst. to be fetched from memory
- **MAR (Memory Address Register)** – holds the address of the next instruction or data to be fetched from memory.
- **MDR (Memory Data Register)** – hold the information (word) to be sent to/from memory.
- **AC (accumulator)** – a special register which holds the data to be manipulated by the ALU.
- **IR (Instruction Register)** – holds the instruction to be decoded by the Control Unit (CU).

Ch. 3: Computer Organization Fundamentals

5

5

A Pseudo-CPU

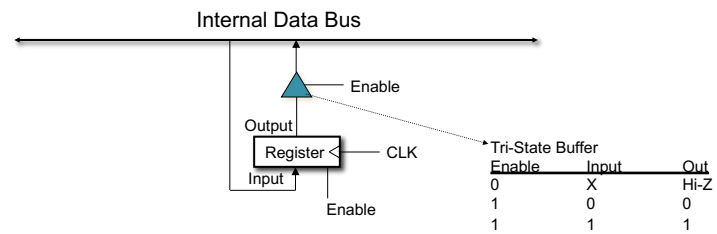


Ch. 3: Computer Organization Fundamentals

6

6

Bus-Register Connections

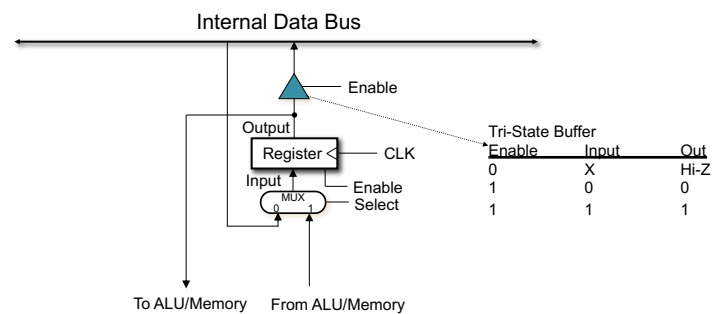


Ch. 3: Computer Organization Fundamentals

7

7

Bus-Register Connections



Ch. 3: Computer Organization Fundamentals

8

8

Fetch and Execute Cycle

- A series of steps (i.e., micro-operations) a computer takes to fetch *and* execute one instruction.
 - Each micro-operation requires a clock cycle.
- **Fetch and execute cycle \Rightarrow Instruction Cycle.**
- Number of micro-operations required to fetch an instruction is usually the same.
- Number of micro-operations required to execute each instruction differs depending on
 - Complexity of the instruction
 - e.g., Multiply takes longer than Add
 - Available hardware
 - e.g., Multiplier vs. no multiplier hardware

Ch. 3: Computer Organization Fundamentals

9

9

Fetch Cycle

- Need to describe what has to happen in each cycle.
- Will use **register transfer operations** to describe the movement of data.

Fetch Cycle

Cycle 1: $MAR \leftarrow PC$

Cycle 2: $MDR \leftarrow M[MAR]$; Read the content of memory
; location pointed to by MAR

Cycle 3: $IR \leftarrow MDR(opcode), MAR \leftarrow MDR(address)$

Cycle 4: $PC \leftarrow PC + I$

Go to beginning of Execute cycle

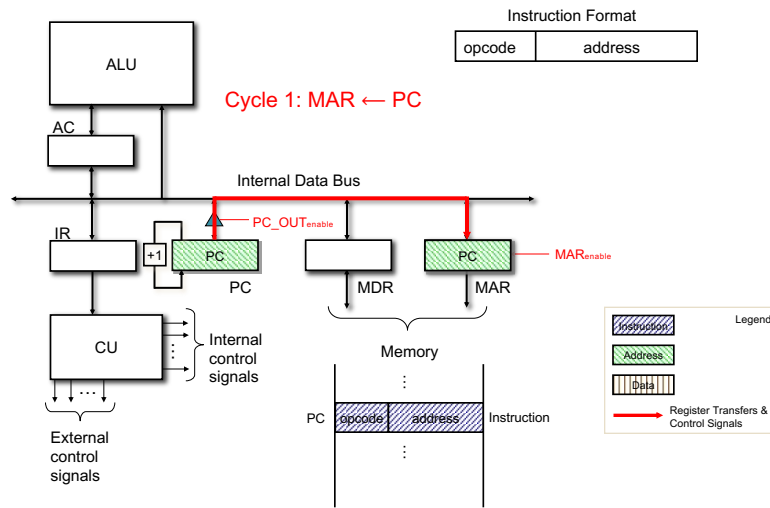
Note: Cycles 2 and 4 can be performed at the same time.

Ch. 3: Computer Organization Fundamentals

10

10

Fetch Cycle (Step 1)

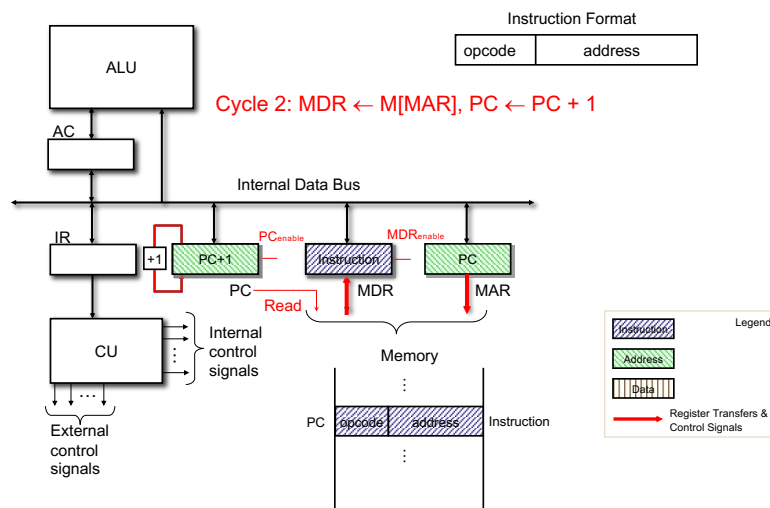


Ch. 3: Computer Organization Fundamentals

11

11

Fetch Cycle (Step 2)

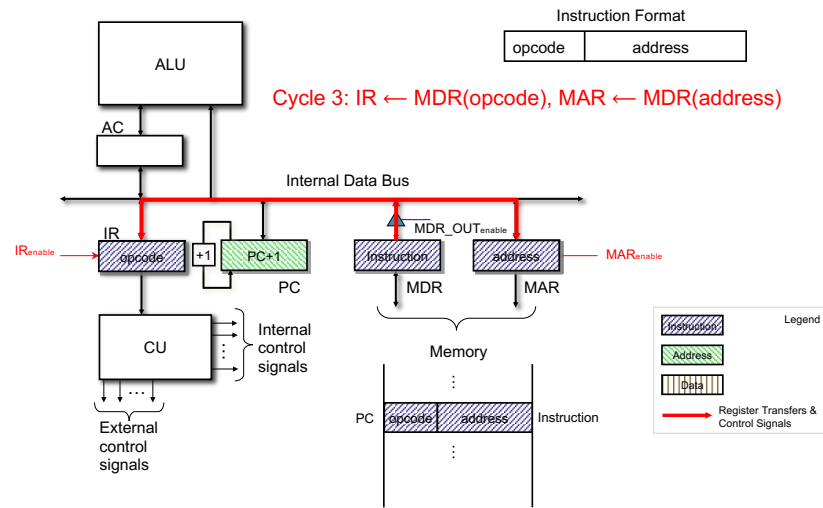


Ch. 3: Computer Organization Fundamentals

12

12

Fetch Cycle (Step 3)



Ch. 3: Computer Organization Fundamentals

13

13

Execute Cycle

- Execute cycle depends on the instruction
- Will describe execute cycle based on instruction in the pseudo-ISA:
 - Data transfer Instructions
 - LDA x (Load Accumulator) ✓
 - STA x (Store Accumulator) ✓
 - Arithmetic and Logical Instructions
 - ADD x (Add to accumulator) ✓
 - SUB x (Subtract from accumulator)
 - NAND x (Logical NAND to accumulator)
 - SHFT (Shift accumulator)
 - Control Transfer
 - J x (Jump to x) ✓
 - BNE x (Branch conditionally to x) ✓

Ch. 3: Computer Organization Fundamentals

14

14

Execute Cycle

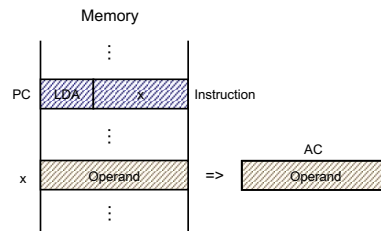
Example: LDA x (Load Accumulator)

Execute Cycle

Cycle 1: $MDR \leftarrow M[MAR]$

Cycle 2: $AC \leftarrow MDR$

Return to the beginning of the instruction cycle

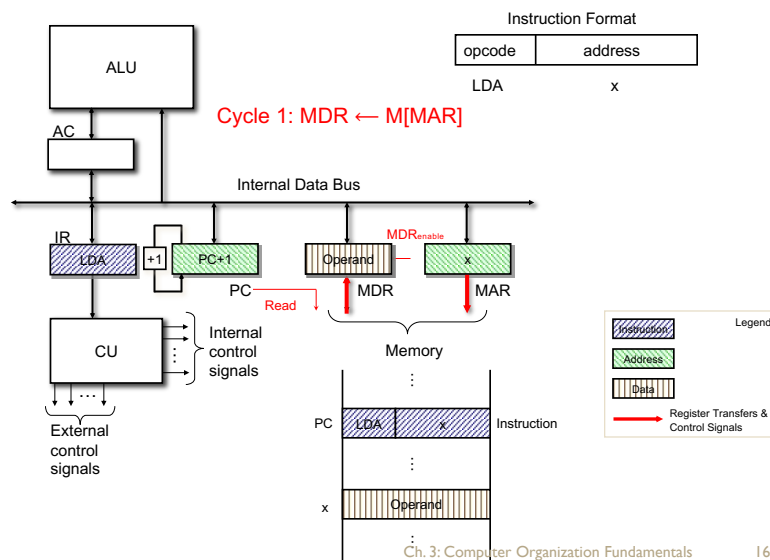


Ch. 3: Computer Organization Fundamentals

15

15

Execute Cycle (Step I)



Ch. 3: Computer Organization Fundamentals

16

16

[illegible]

17

Execute Cycle

Example: STA x (*Store Accumulator*)

Execute Cycle

Cycle 1: MDR \leftarrow AC

Cycle 2: M[MAR] \leftarrow MDR

Return to the beginning of the instruction cycle

The diagram illustrates the execution of the STA instruction. It shows a vertical stack of memory locations. The top location is labeled 'PC' and contains the instruction 'STA x'. The next location is labeled 'x' and contains the operand 'x'. To the right of the memory stack, the Accumulator (AC) is shown containing the operand 'x'. An arrow labeled '<=' points from the AC to the memory location 'x', indicating the transfer of data from the accumulator to memory.

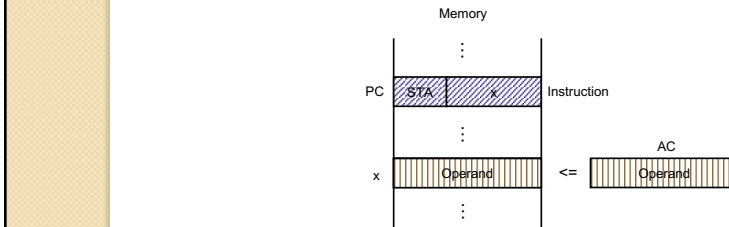
Example: STA x (Store Accumulator)

Execute Cycle

Cycle 1: $MDR \leftarrow AC$

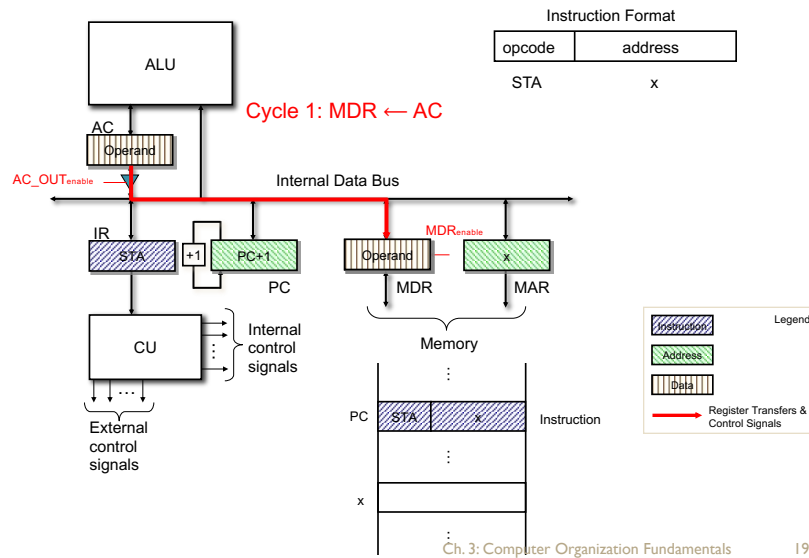
Cycle 2: $M[MAR] \leftarrow MDR$

Return to the beginning of the instruction cycle



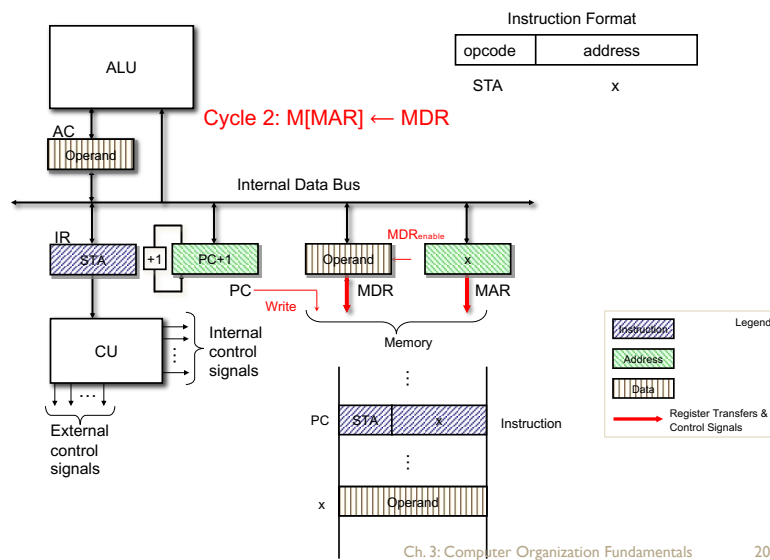
18

Execute Cycle (Step 1)



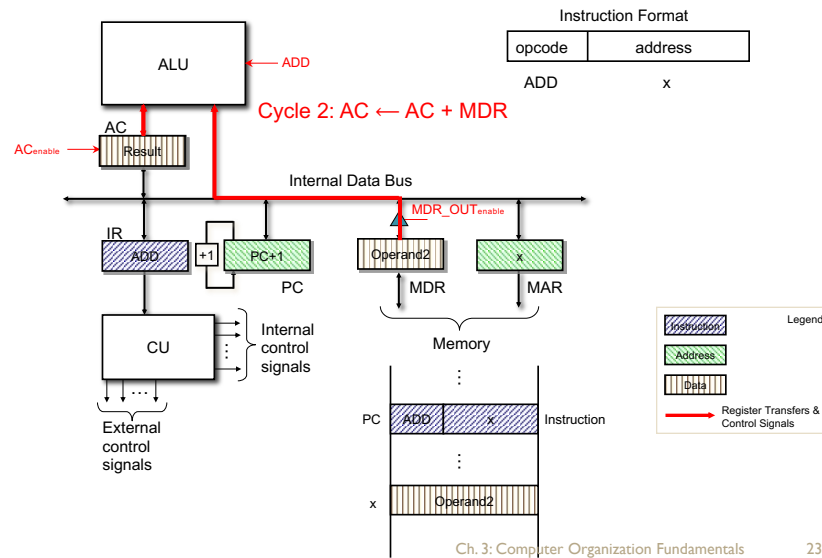
19

Execute Cycle (Step 2)



20

Execute Cycle (Step 2)



23

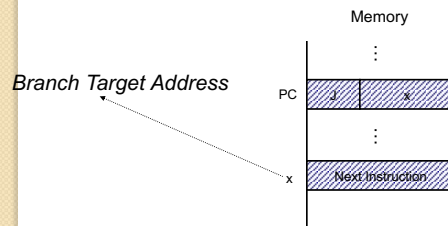
Execute Cycle

Example: J x (Jump to x)

Execute Cycle

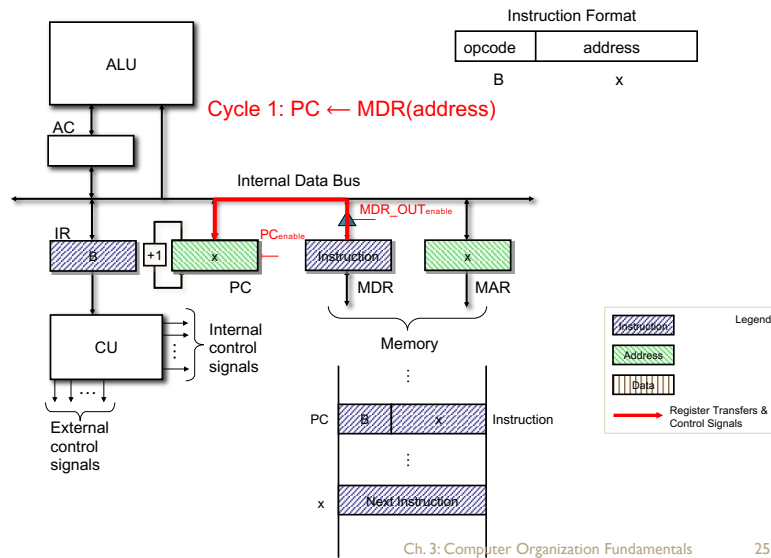
Cycle 1: $PC \leftarrow MDR(\text{address})$

Return to the beginning of the instruction cycle



24

Execute Cycle (Step I)



25

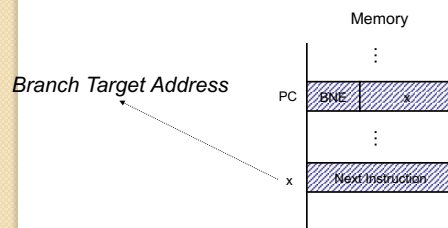
Execute Cycle

Example: BNE x (Branch Conditionally to x)

Execute Cycle

Cycle 1: If (Z != 1) then $PC \leftarrow MDR(\text{address})$

Return to the beginning of the instruction cycle

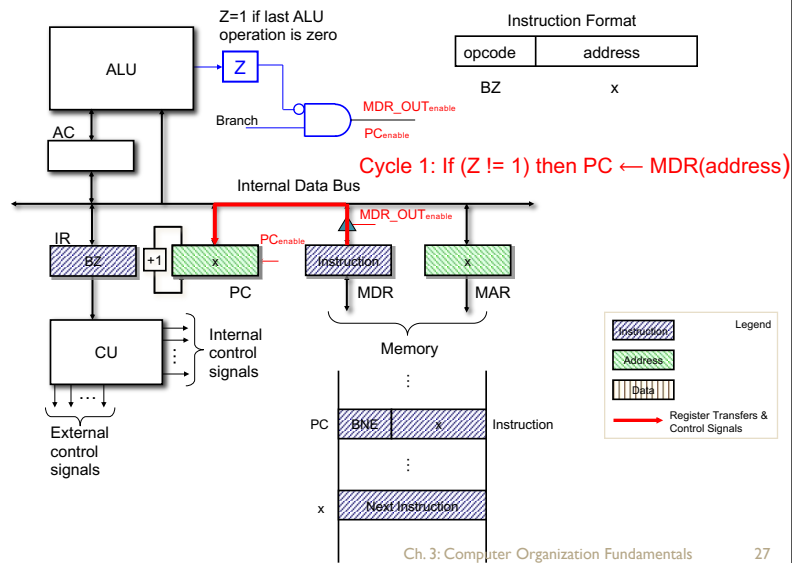


Ch. 3: Computer Organization Fundamentals

26

26

Execute Cycle (Step I)



27

One More Example...

Example: LDA (x) (Load Accumulator Indirect)

Execute Cycle

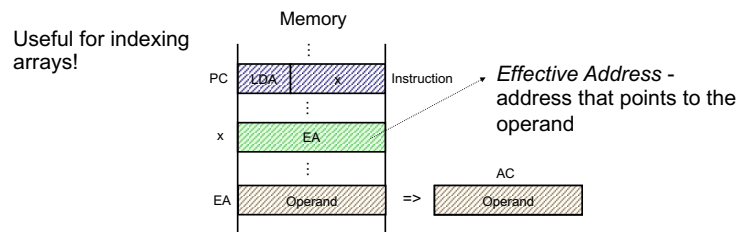
Cycle 1: MDR ← M[MAR] ; Read effective address (EA)

Cycle 2: MAR ← MDR ; Move EA to MAR

Cycle 3: MDR ← M[MAR] ; Read operand

Cycle 4: AC ← MDR ; Move operand to AC

Return to the beginning of the instruction cycle

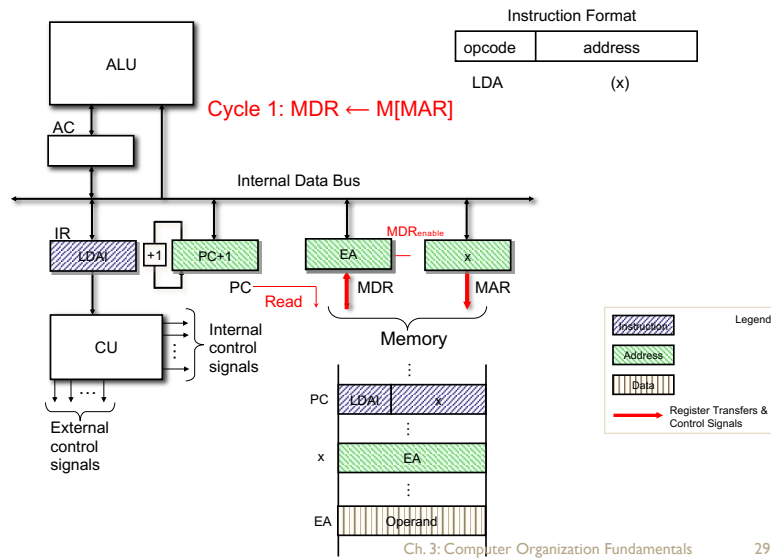


Ch. 3: Computer Organization Fundamentals

28

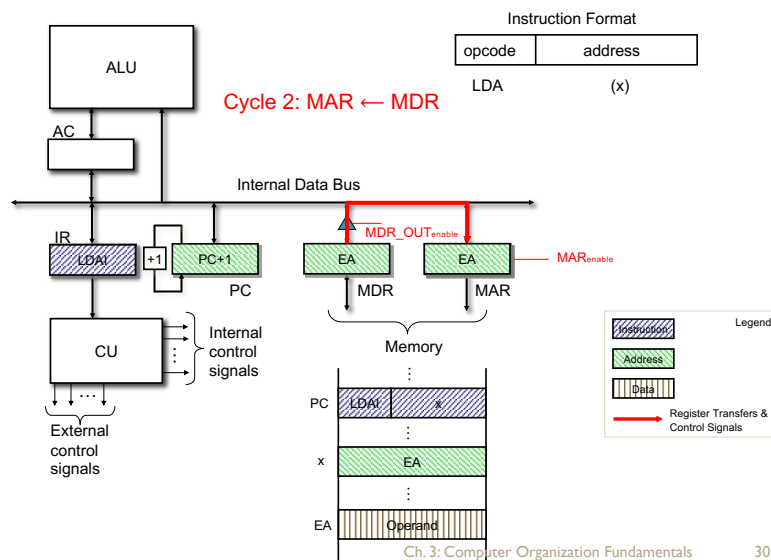
28

Execute Cycle (Step 1)



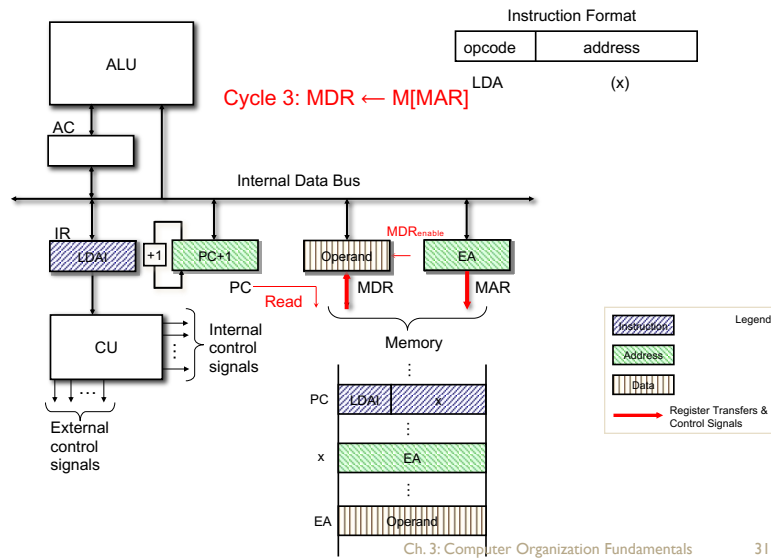
29

Execute Cycle (Step 2)



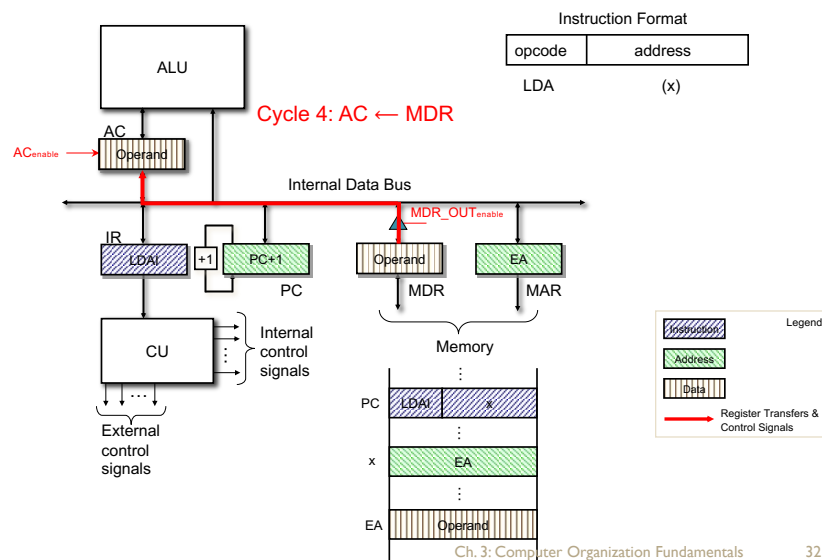
30

Execute Cycle (Step 3)



31

Execute Cycle (Step 4)



32

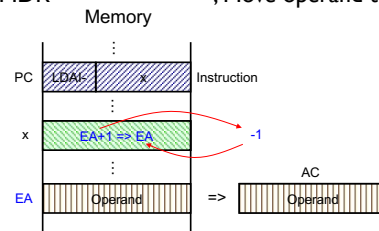
Last Example...(I promise!)

Example: LDA $-(x)$ (Load Accumulator Indirect with *Pre-decrement*)

Useful for stepping through arrays

Execute Cycle

Cycle 1: $MDR \leftarrow M[MAR]$; Read EA+I
 Cycle 2: $MDR \leftarrow MDR - 1$; Decrement EA
 Cycle 3: $M[MAR] \leftarrow MDR$; Store it back in location x (i.e., $M[x]$)
 Cycle 4: $MAR \leftarrow MDR$; Move EA to MAR
 Cycle 5: $MDR \leftarrow M[MAR]$; Read operand
 Cycle 6: $AC \leftarrow MDR$; Move operand to AC

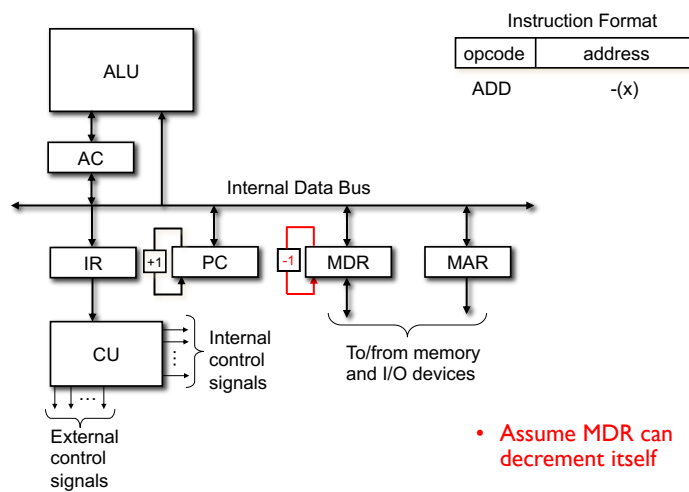


Ch. 3: Computer Organization Fundamentals

33

33

Easiest Way

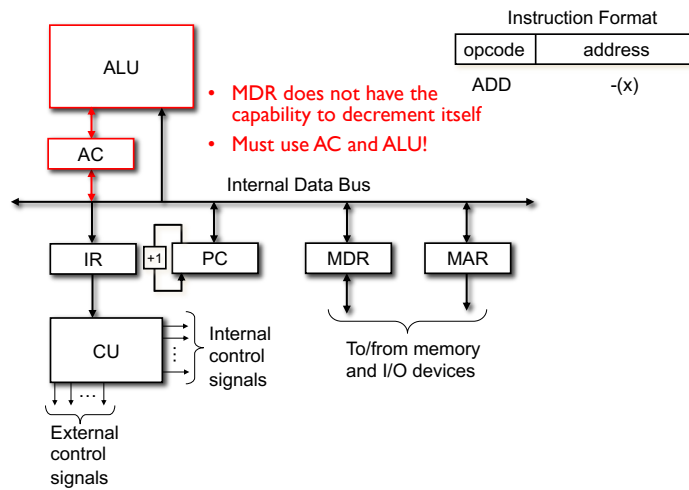


Ch. 3: Computer Organization Fundamentals

34

34

Hard Way (1)



Ch. 3: Computer Organization Fundamentals

35

35

Hard Way (2)

Execute Cycle

Cycle 1: $MDR \leftarrow M[EA+I]$; Read EA+I

Cycle 2: $AC \leftarrow MDR$; Move EA+I to AC

Cycle 3: $AC \leftarrow AC - I$; Decrement EA+I

Cycle 4: $MDR \leftarrow AC$; Move EA to MDR

Cycle 5: $M[MAR] \leftarrow MDR$; Store it back in location x (i.e., $M[x]$)

Cycle 6: $MAR \leftarrow MDR$; Move EA MAR

Cycle 7: $MDR \leftarrow M[MAR]$; Read operand

Cycle 8: $AC \leftarrow MDR$; Move operand to AC

Note: Cycles 5 and 6 can be performed at the same time.

Ch. 3: Computer Organization Fundamentals

36

36

Pseudo-CPU with TEMP Register

- There may be times when the content of AC has to be preserved.
- TEMP increases the flexibility for implementing more complicated instructions, e.g., STA $-(x)$, LDA $(x)+$, STA $(x)+$.

The diagram illustrates the internal structure of a Pseudo-CPU. At the top is the ALU (Arithmetic Logic Unit), which is connected to the AC (Accumulator) and the Internal Data Bus. The AC is also connected to the Internal Data Bus. Below the bus are the IR (Instruction Register), PC (Program Counter), MDR (Memory Data Register), and MAR (Memory Address Register). The PC has a '+1' incrementer. The CU (Control Unit) is connected to the IR, PC, and the Internal Data Bus. The CU also receives external control signals. The MDR and MAR are connected to the Internal Data Bus and are labeled 'To/from memory and I/O devices'. A TEMP register is shown above the Internal Data Bus, connected to it.

Ch. 3: Computer Organization Fundamentals 37

- 37



37

What We Will See Later...

- AVR Microcontroller
- Used in low-end embedded systems

The diagram illustrates the internal architecture of an AVR microcontroller, showing the flow of data and control signals between various components. It is divided into two main sections: Fetch and Execute.

Fetch Stage:

- Program Memory:** Provides instructions to the PC (Program Counter).
- PC (Program Counter):** Holds the current instruction address. It is updated from PC+1 or k16.
- PMAR (Program Memory Address Register):** Holds the address of the instruction being fetched.
- PC+1:** Logic that increments the PC by 1 or by a constant k.
- Multiplexers:** Select between different sources for the PC and PMAR.

Execute Stage:

- MDR (Memory Data Register):** Holds data from memory.
- IR (Instruction Register):** Holds the instruction being executed.
- DIMAR (Data Memory Address Register):** Holds the address of data in memory.
- RAR (Register Address Register):** Holds the address of the register being accessed.
- NPC (Next Program Counter):** Holds the address of the next instruction.
- SP (Stack Pointer):** Holds the address of the top of the stack.
- Register File:** Holds data in registers. It is accessed via the RAR.
- ALU (Arithmetic Logic Unit):** Performs arithmetic and logic operations on data from the Register File or Memory.
- Data Memory:** Provides data to the ALU and the Address Adder.
- Address Adder:** Calculates the address of the next instruction based on the PC and the instruction.
- Multiplexers:** Select between different sources for the ALU and the Address Adder.

The diagram shows the flow of data and control signals between these components, illustrating the internal architecture of the AVR microcontroller.

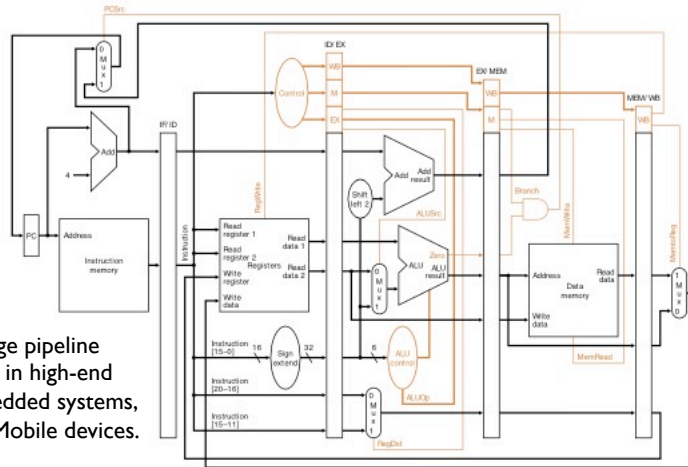
- 38



38

What You Will See in ECE 472...

- 5-stage pipeline
- Used in high-end embedded systems, e.g., Mobile devices.



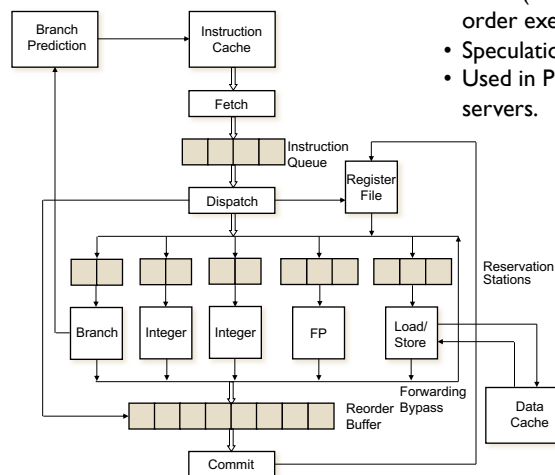
Ch. 3: Computer Organization Fundamentals

39

39

What You Will See in ECE 570...

- SuperScalar
- OoO (out-of-order execution)
- Speculation
- Used in PCs and servers.



Ch. 3: Computer Organization Fundamentals

40

40

Questions?

