Computer Organization and Architecture

Module 2

Basic Operation of Computer Systems, Instruction Set & Addressing Modes, Instruction Encoding

Prof. Indranil Sengupta

Dr. Sarani Bhattacharya

Department of Computer Science and Engineering

IIT Kharagpur

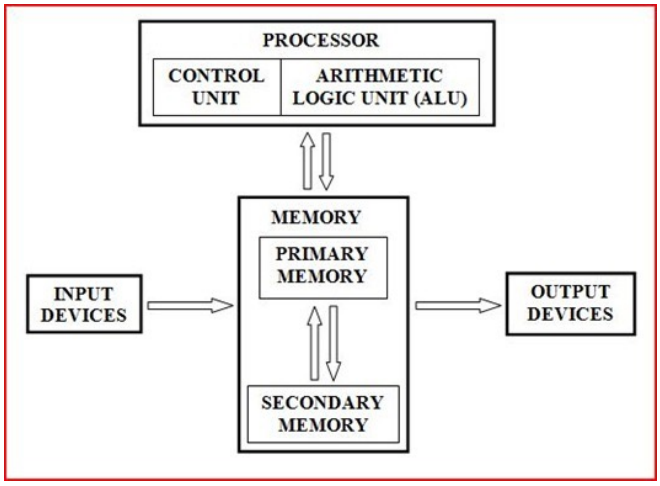
1

Basic Block Diagram of a Computer System

2

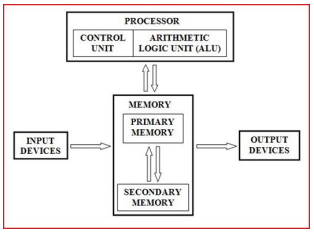
Simplified Block Diagram

- All instructions and data are stored in memory.
- An instruction and the required data are brought into the processor for execution.
- Input / Output devices interface with the outside world.
- Referred to as *von-Neumann architecture*.



3

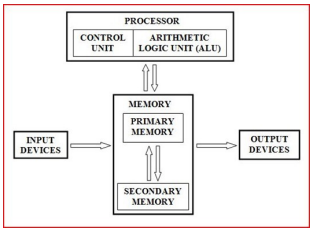
3



- **Inside the Processor**
  - Also called *Central Processing Unit* (CPU).
  - Consists of a *Control Unit* and an *Arithmetic Logic Unit* (ALU).
    - All calculations happen inside the ALU.
    - The Control Unit generates sequence of control signals to carry out all operations.
  - The processor fetches an instruction from memory for execution.
    - An instruction specifies the exact operation to be carried out.
    - It also specifies the data that are to be operated on.

4

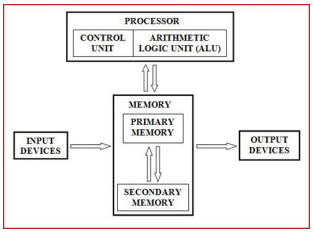
4



- **What is the role of ALU?**
  - It contains several registers, some general-purpose and some special-purpose, for temporary storage of data.
  - It contains circuitry to carry out logic operations, like AND, OR, NOT, shift, compare, etc.
  - It contains circuitry to carry out arithmetic operations like addition, subtraction, multiplication, division, etc.
  - During instruction execution, the data (operands) are brought in and stored in some registers, the desired operation carried out, and the result stored back in some register or memory.

5

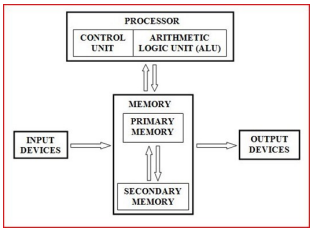
5



- **What is the role of control unit?**
  - Acts as the nerve center that senses the states of various functional units and sends control signals to control their states.
  - To carry out a specific operation (say, R1 ← R2 + R3), the control unit must generate control signals in a specific sequence.
    - Enable the outputs of registers R2 and R3.
    - Select the addition operation.
    - Store the output of the adder circuit into register R1.
  - When an instruction is fetched from memory, the operation (called *opcode*) is decoded by the control unit, and the control signals issued.

6

6



- **Inside the Memory Unit**
  - Two main types of memory subsystems:
    - *Primary or Main memory*, which stores the active instructions and data for the program being executed on the processor.
    - *Secondary memory*, which is used as a backup and stores all active and inactive programs and data, typically as files.
  - The processor only has direct access to the primary memory.
  - In reality, the memory system is implemented as a hierarchy of several levels.
    - L1 cache, L2 cache, L3 cache, primary memory, secondary memory.
    - Objective is to provide faster memory access at affordable cost.

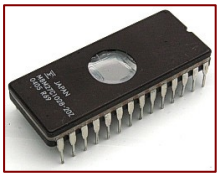
7

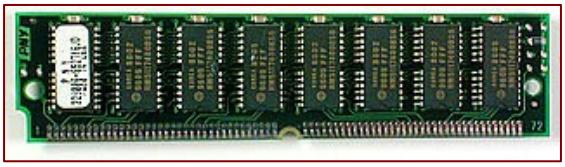
7

- Various different types of memory are possible.
  a) Random Access Memory (RAM), which is used for the cache and primary memory sub-systems. Read and Write access times are independent of the location being accessed.
  b) Read Only Memory (ROM), which is used as part of the primary memory to store some fixed data that cannot be changed.
  c) Magnetic Disk, which uses direction of magnetization of tiny magnetic particles on a metallic surface to store data. Access times vary depending on the location being accessed, and is used as secondary memory.
  d) Flash Memory or Solid-State Drives (SSD), which is replacing magnetic disks as secondary memory devices. They are faster, but smaller in size as compared to disk.

8

8





9

9

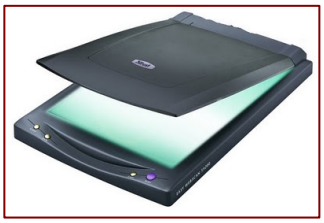
Input Unit

- Used to feed data to the computer system from the external environment.
  - Data are transferred to the processor/memory after appropriate encoding.
- Common input devices:
  - Keyboard
  - Mouse
  - Joystick
  - Camera

10

10







11

11

Output Unit

- Used to send the result of some computation to the outside world.
- Common output devices:
  - LCD/LED screen
  - Printer and Plotter
  - Speaker / Buzzer
  - Projection system

12

12






13

13

Basic Operation of a Computer

14

Introduction

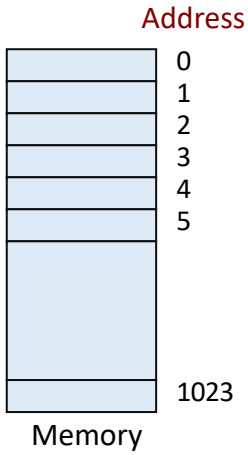
- The basic mechanism through which an instruction gets executed shall be illustrated.
- May be recalled:
  - ALU contains a set of registers, some general-purpose and some special-purpose.
  - First we briefly explain the functions of the special-purpose registers before we look into some examples.

15

15

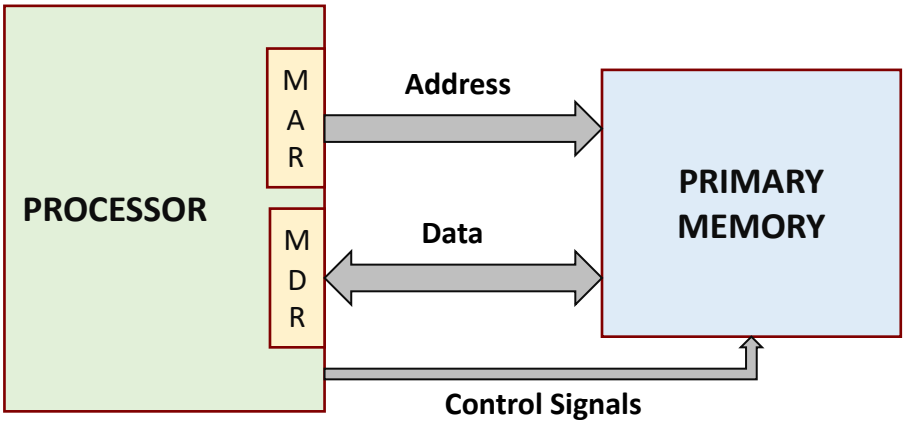
For Interfacing with the Primary Memory

- Two special-purpose registers are used:
  - **Memory Address Register (MAR)**: Holds the address of the memory location to be accessed.
  - **Memory Data Register (MDR)**: Holds the data that is being written into memory, or will receive the data being read out from memory.
- Memory considered as a linear array of storage locations (bytes or words) each with unique address.



16

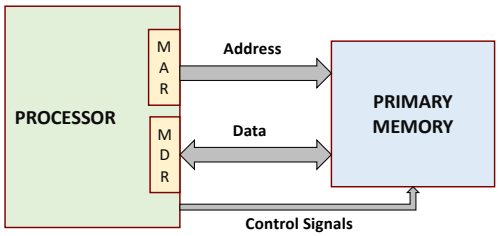
16



17

17

- To read data from memory
  - Load the memory address into MAR.
  - Issue the control signal READ.
  - The data read from the memory is stored into MDR.
- To write data into memory
  - Load the memory address into MAR.
  - Load the data to be written into MDR.
  - Issue the control signal WRITE.



18

18

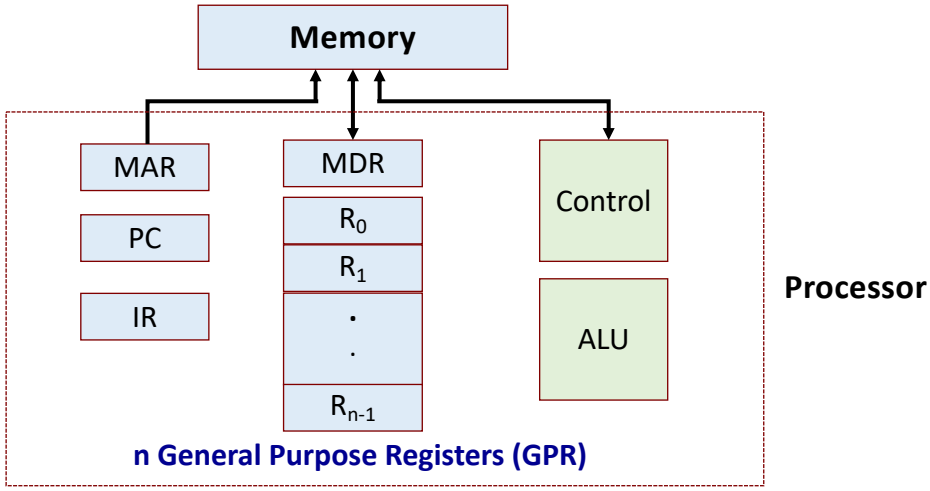
For Keeping Track of Program / Instructions

- Two special-purpose registers are used:
  - **Program Counter (PC)**: Holds the memory address of the next instruction to be executed.
    - Automatically incremented to point to the next instruction when an instruction is being executed.
  - **Instruction Register (IR)**: Temporarily holds an instruction that has been fetched from memory.
    - Need to be decoded to find out the instruction type.
    - Also contains information about the location of the data.

19

19

Overall Architecture of a Simple Processor



20

20

Example Instructions

- We shall illustrate the process of instruction execution with the help of the following two instructions:

a) ADD R1, LOCA

Add the contents of memory location LOCA (i.e. address of the memory location is LOCA) to the contents of register R1.

R1 ← R1 + Mem[LOCA]

b) ADD R1, R2

Add the contents of register R2 to the contents of register R1.

R1 ← R1 + R2

21

21

Execution of *ADD R1,LOCA*

- Assume that the instruction is stored in memory location 1000, the initial value of R1 is 50, and LOCA is 5000.
- Before the instruction is executed, PC contains 1000.
- Content of PC is transferred to MAR. MAR ← PC
- READ request is issued to memory unit.
- The instruction is fetched to MDR. MDR ← Mem[MAR]
- Content of MDR is transferred to IR. IR ← MDR
- PC is incremented to point to the next instruction. PC ← PC + 4
- The instruction is decoded by the control unit.

ADD R1	5000

22

22

- LOCA (i.e. 5000) is transferred (from IR) to MAR. MAR ← IR[Operand]
- READ request is issued to memory unit.
- The data is fetched to MDR. MDR ← Mem[MAR]
- The content of MDR is added to R1. R1 ← R1 + MDR

The steps being carried out are called micro-operations:

MAR ← PC
MDR ← Mem[MAR]
IR ← MDR
PC ← PC + 4
MAR ← IR[Operand]
MDR ← Mem[MAR]
R1 ← R1 + MDR

23

23

R1 125

Address	Content
1000	ADD R1, LOCA
1004	…

5000	75

LOCA

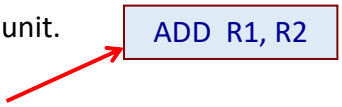
1. PC = 1000
2. MAR = 1000
3. PC = PC + 4 = 1004
4. MDR = ADD R1, LOCA
5. IR = ADD R1, LOCA
6. MAR = LOCA = 5000
7. MDR = 75
8. R1 = R1 + MDR = 50 + 75 = 125

24

24

Execution of *ADD R1,R2*

- Assume that the instruction is stored in memory location 1500, the initial value of R1 is 50, and R2 is 200.
- Before the instruction is executed, PC contains 1500.
- Content of PC is transferred to MAR. — MAR ← PC
- READ request is issued to memory unit.
- The instruction is fetched to MDR. — MDR ← Mem[MAR]
- Content of MDR is transferred to IR. — IR ← MDR
- PC is incremented to point to the next instruction. — PC ← PC + 4
- The instruction is decoded by the control unit.
- R2 is added to R1. — R1 ← R1 + R2



25

25

R1 250

R2 200

Address	Instruction
1500	ADD R1, R2
1504	…

1. PC = 1500
2. MAR = 1500
3. PC = PC + 4 = 1504
4. MDR = ADD R1, R2
5. IR = ADD R1, R2
6. R1 = R1 + R2 = 250

26

26

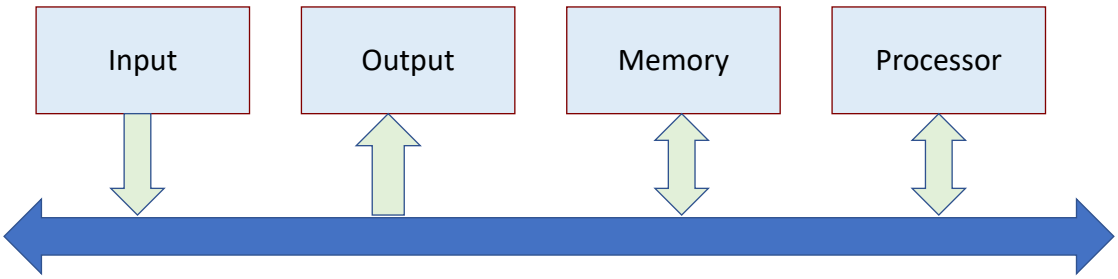
Bus Architecture

- The different functional modules must be connected in an organized manner to form an operational system.
- Bus refers to a group of lines that serves as a connecting path for several devices.
- The simplest way to connect the functional unit is to use the *single bus architecture*.
  - Only one data transfer allowed in one clock cycle.
  - For multi-bus architecture, parallelism in data transfer is allowed.

27

27

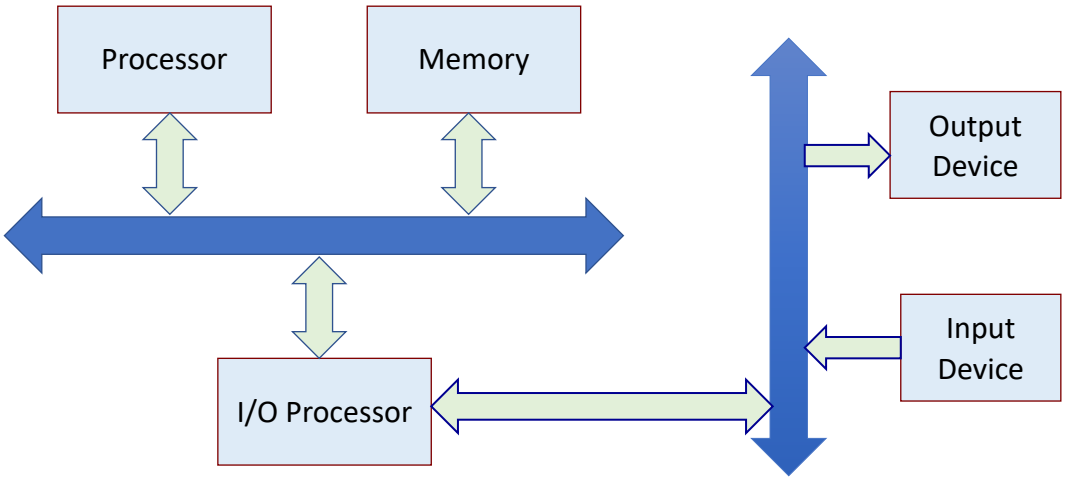
System-Level Single Bus Architecture



28

28

System-Level Two-Bus Architecture



29

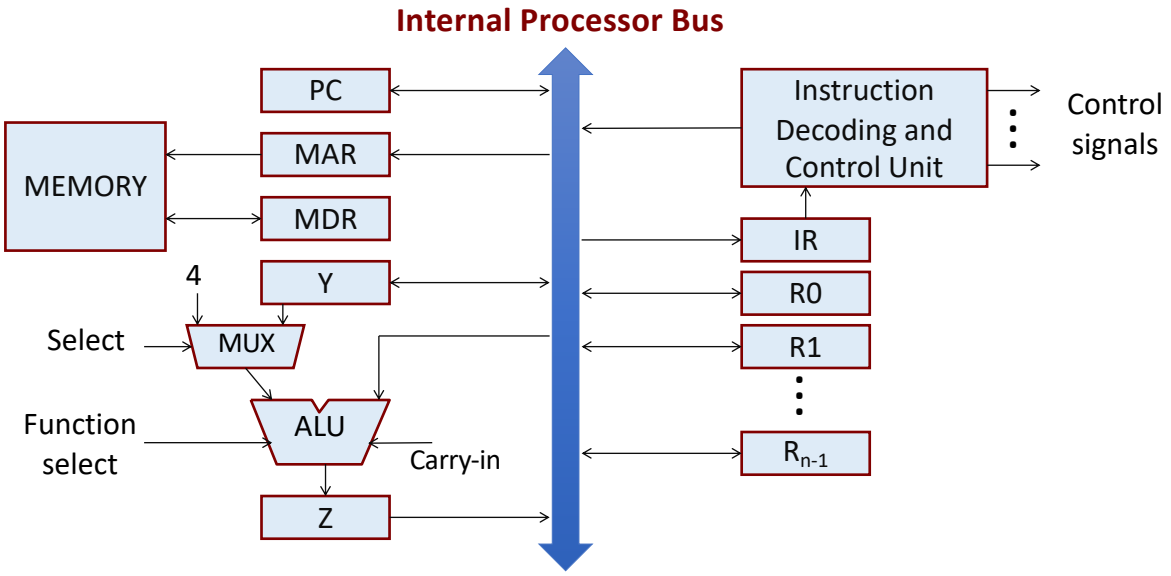
29

Single-Bus Architecture Inside the Processor

- There is a single bus inside the processor.
  - ALU and the registers are all connected via the single bus.
  - This bus is *internal* to the processor and should not be confused with the external bus that connects the processor to the memory and I/O devices.
- A typical single-bus processor architecture is shown on the next slide.
  - Two temporary registers *Y* and *Z* are also included.
  - Register *Y* temporarily holds one of the operands of the ALU.
  - Register *Z* temporarily holds the result of the ALU operation.
  - The multiplexer selects a constant operand 4 during execution of the micro-operation: $PC \leftarrow PC + 4$.

30

30



31

31

Multi-Bus Architectures

- Modern processors have multiple buses that connect the registers and other functional units.
  - Allows multiple data transfer micro-operations to be executed in the same clock cycle.
  - Results in overall faster instruction execution.
- Also advantageous to have multiple shorter buses rather than a single long bus.
  - Smaller parasitic capacitance, and hence smaller delay.

32

32