COMPUTER ORGANIZATION AND ARCHITECTURE (IT 2202)

Lecture 3

Prof Hafizur Rahaman

Indian Institute of Engineering Science and Technology, Shibpur
(Formerly, Bengal Engineering and Science University, Shibpur)
Howrah 711 103, West Bengal, India

Basic Operation Of A Computer

- We will discuss the basic mechanism through which an instruction gets executed.
- ALU has different kind of registers
- General purpose registers
- Special purpose registers
- Temporary Registers

We will discuss the function of these registers.

Interfacing with Primary Memory

- Instructions and data are stored in memory. From memory, the data and the instruction are brought to the processors and then instruction will be executed.
- For this purpose, we require two special purpose registers
  - Memory Address Register (MAR)
  - Memory Data Register (MDR)
- Memory address register (MAR) holds the address of a memory location to be accessed. MAR holds the address of memory location that is to be accessed. We can access a memory location for reading an instruction or we can access a memory location for reading a data and we can also access a memory location for writing back the data. MAR holds the address of the instruction that is to be read or it holds the address of the data that is to be read from the memory or the address of the memory where the data is to be written.

Interfacing with Primary Memory

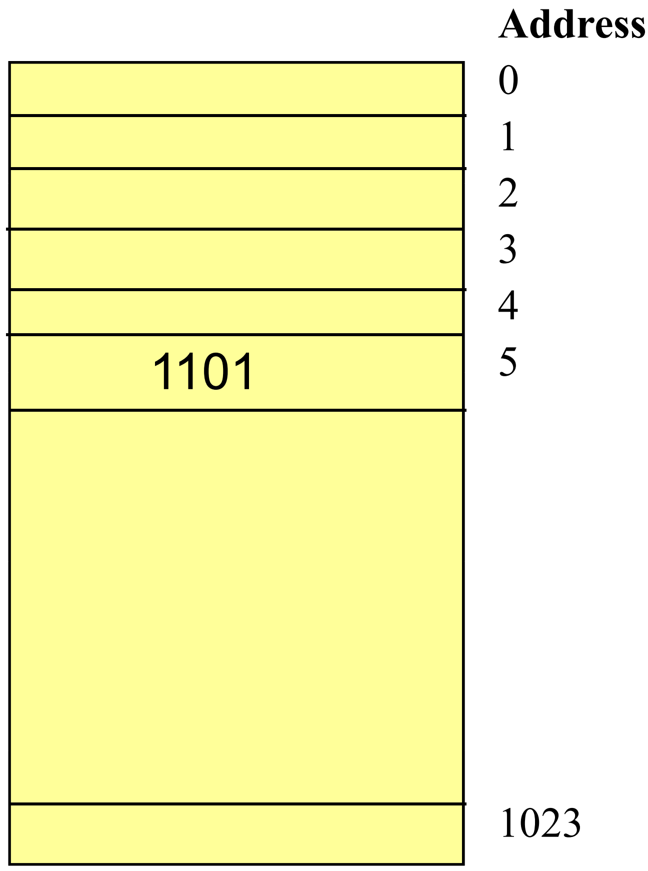
- Memory Data Register (MDR) holds the data that will be written into memory or the data that we will receive when read out from the memory location. When we write, we always write a data into the memory. But when we read, we can read an instruction or we can read a data.

- MDR will contain the instruction that is to be read from the memory or the data that is to be read from the memory or the data that is to be written into the memory. MDR contain any of these three.

- A memory is considered as a linear array of storage locations, bytes or words, each with a unique address.

Interfacing with Primary Memory

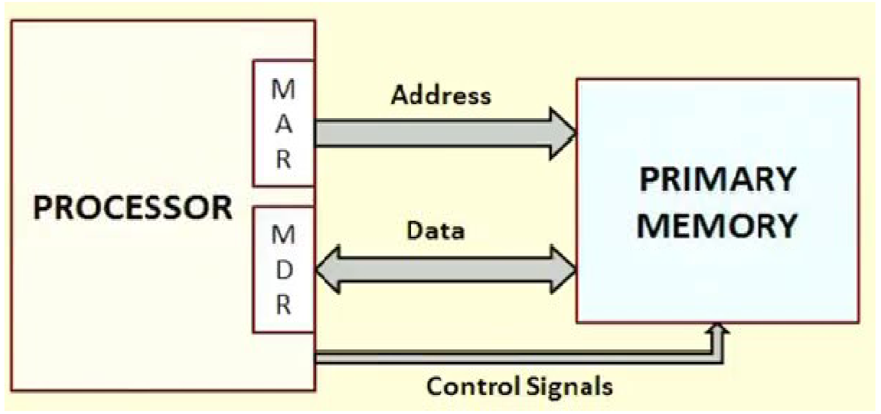
- In this Fig., address is spanning from 0 to 1023.
- Number of memory locations that we can address is starting from 0 to 1023, are 1024.
- A memory is considered as a linear array of storage locations,.
- Each location which forms a unique address stores bytes or words.
- These address of the memory location starts from 0 and is incremented one by one and it has got 1024 locations.



- Memory address register will hold one of such address that is either 0, 1, 3,4 or 5 or anything
- Memory data register will hold the content of that location.
- If it is 5, whatever content will be there in that particular location that will be present in MDR.

Interfacing with Primary Memory

- This diagram shows the connection between processor and main memory.
- MAR is connected with primary memory through address bus.
- MDR is connected with primary memory through data bus. Some control signals are also required.



- When a particular data will be read from the memory or the particular data will be written into the memory, we need to specify the location from where, we need to read a data or in where, we need to write back data. These operations will be synchronized by control signals.
- First, we provide the address in the address bus that hits to the primary memory, then we provide the control signals either Read or Write depending on that, a particular value is read from that particular address and it comes to MDR.
- If we want to write a value, we have to put that value in MDR and the address where we want to write in MAR, and then we activate the write control signal. According to read or write control, a word is read or written from or into memory.

Interfacing with Primary Memory

- To read the data from memory, we first load the memory address into MAR then we issue the control signals i.e., read then the data from memory is read and it is stored into MDR.
- To write into memory, we have to load the memory address into MAR, the data that is to be written must be loaded into MDR and then we issue write control signal.

Handling Program/Instructions

- For this purpose, we have two special purpose registers.
- One is called Program Counter (PC) that holds the memory address of the next instruction to be executed. Automatically, it is incremented to point to the next instruction when an instruction is fetched and about to be executed. So, PC holds the memory address of the next instruction to be executed. So, once we read and execute an instruction, PC must point to the next instruction that we have to execute next and so on.

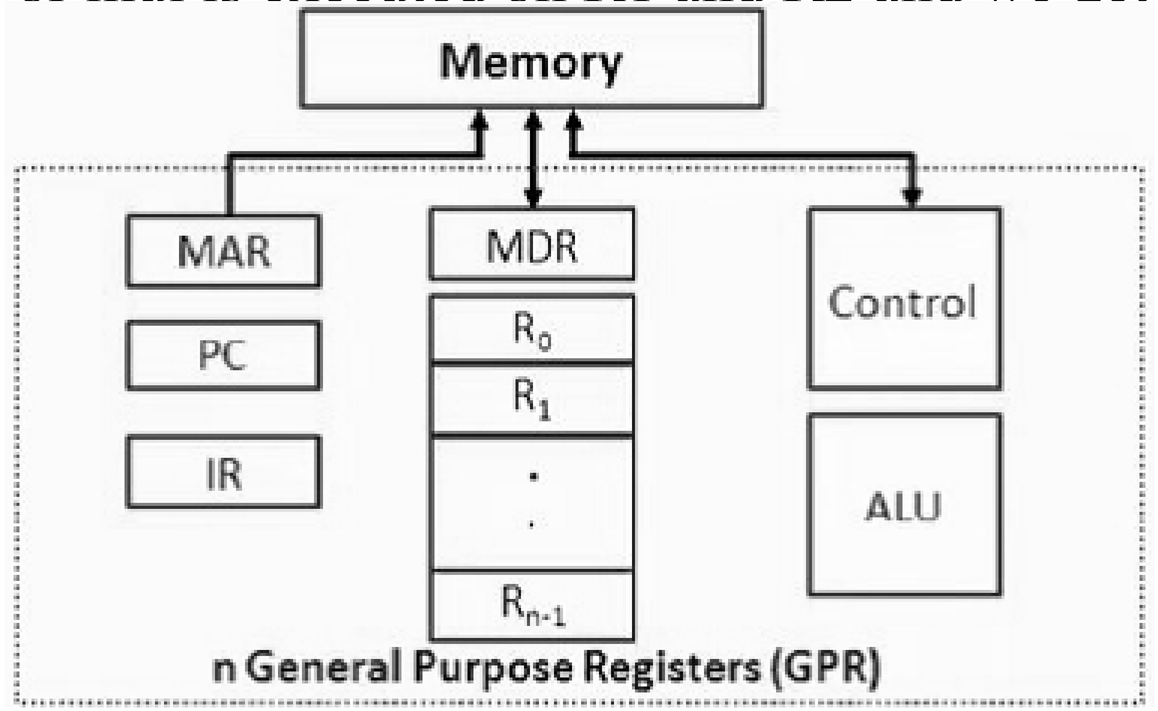
- Another register is the Instruction Register. Actually, PC will contain the location from where we have to read the instruction. Now, we have to read this instruction and store it somewhere. The register where it will get stored is known as Instruction Register. Suppose, according PC content, the location which contains ADD R1, R2 is identified, where this entire instruction ADD R1,R2 is fetched and will be stored in IR. So, IR temporarily holds an instruction that has been fetched from memory. Now, once an instruction is fetched from memory, we need to decode that instruction.

Handling Program/Instructions

- As per instruction, the computer will perform the activity.
- ADD R1,R2 instruction is saying that it adds the content of register R1 with R2, and store back in R1. Here, the instruction is decoded and then the computer understands it and then executes the instruction that adds the content of R1 and R2 and, store back the result in R1.
- The instruction register temporarily holds an instruction that has been fetched from memory and is decoded to find out the instruction type, which contains information about the location of the data.
- IR contains the entire instruction. After we get this instruction, it is decoded to know that this is add, and we also know locations of R1, R2 registers.

Architecture of Processor: Example

- The architecture of this processor shows how memory system is connected with the processor.
- The processor consists of special purpose registers: MAR, MDR, PC and IR.
- It contains some general purpose registers, control unit and ALU.
- In ADD R1,R2 instruction, R1 and R2 are registers that are present inside the processor.
- These registers (R1 and R2) are brought to ALU and then the particular instruction like add or mul is executed on R1 and R2 and we get the result.



Architecture of Processor: Example

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

- ADD R1, MEM

  Add the contents of memory location (MEM) (i.e., address of the memory location is MEM) to the contents of The processor consists of register R1.

  R1←R1+memory[MEM]

- ADD R1,R2

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

  R1 ← R1+R2)

Architecture of Processor: Example

- We will explain the execution of a instruction say, ADD R1, 5000.
- First, this instruction is stored in some memory location. We assume here that the instruction is stored in location 1000, and the initial value of R1 is 50, and MEM value is 5000. Before the instruction is executed, PC contains the value 1000. Now, the content of PC is transferred to MAR as we need to read the instruction. To read the instruction from memory, the address needs to be loaded in MAR, and we need to activate the read control.
- Hence 1000 now should be transferred to MAR, then a read request is issued to memory unit. After the reading is performed, the instruction is in MDR; and then from MDR, it should be transferred to IR Instruction Register. And while doing these steps, we have to increment the PC to point to the next instruction.
- In computer, there will be sequential execution of instruction and the instructions are stored one by one. Here, first PC was pointing to 1000, we fetch the instruction from location 1000 and then the PC points to the next location that is 1001. Next, instruction read from memory is loaded in MDR and from MDR, it is transferred to IR. In IR, the instruction is decoded and then it is executed.

Architecture of Processor: Example

- First, PC value is transferred to MAR. Read signal is activated. The content specified by the memory location (MAR) is read and it is stored in MDR. From MDR, it is transferred to IR and at the same time PC is incremented to 4.

- In byte addressable organization, every time PC will be incremented by 4. Say, initial location is 2000, and the next location will be 2004 in case of 32-bit machine. The PC is incremented by 4. If it is a 64-bit machine, the PC will get incremented by 8.

- Now, once the instruction comes to IR, it needs to be decoded and executed.

- For example, ADD R1, 5000, after decoding, it has understood that one of the operand is in memory. 5000 is transferred from IR to MAR and this operand is read from memory location (5000) pointed by MAR and other operand will come from R1. It will be executed and addition of the content of R1 and content of memory location will be performed and the result will be saved in R1.

Micro-operations

The steps being carried out are called micro-operations

MAR ← PC

MDR ← MEM [MAR]

IR ← MDR

PC ← PC+4

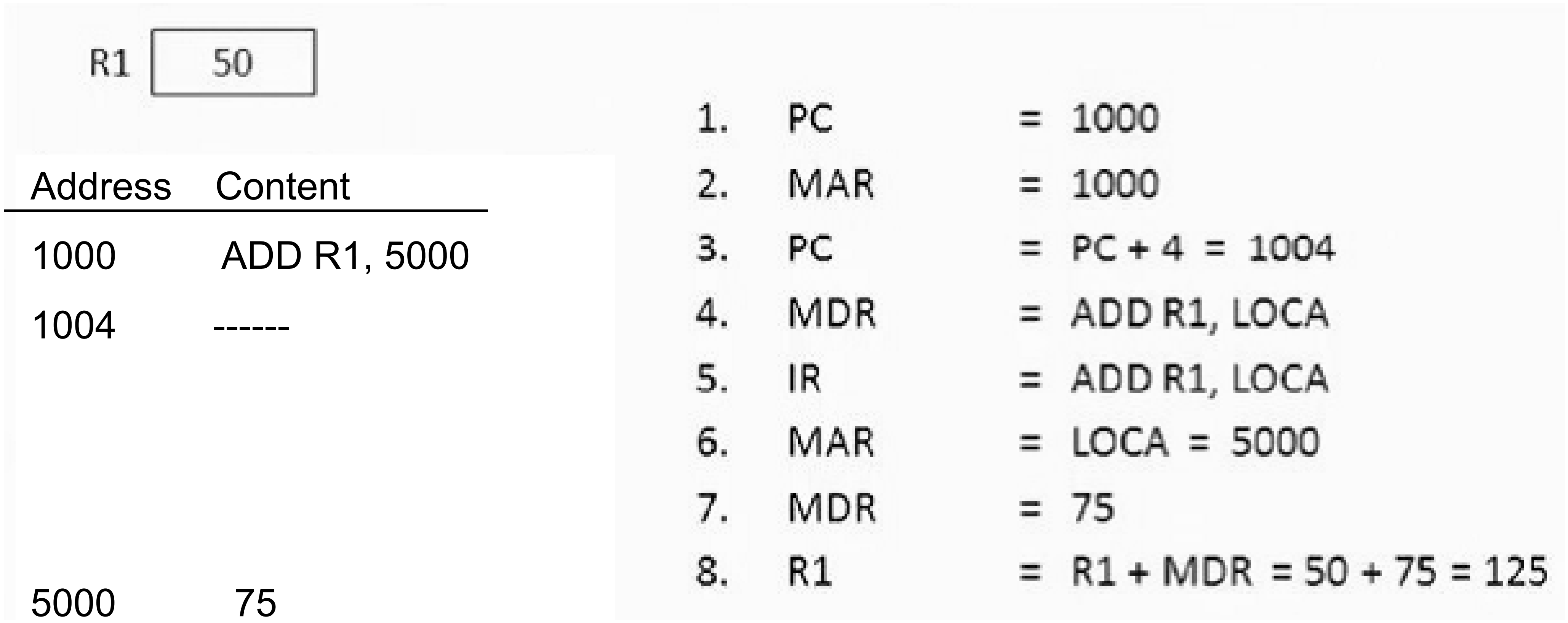
MAR ← IR [OPENAND]

MDR ← MEM[MAR]

R1 ← R1+MDR

Micro-operations: Example

Micro-operations Instruction ADD R1, 5000



R1 $\boxed{50}$

Address	Content
1000	ADD R1, 5000
1004	------
5000	75

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

Micro-operations: Example

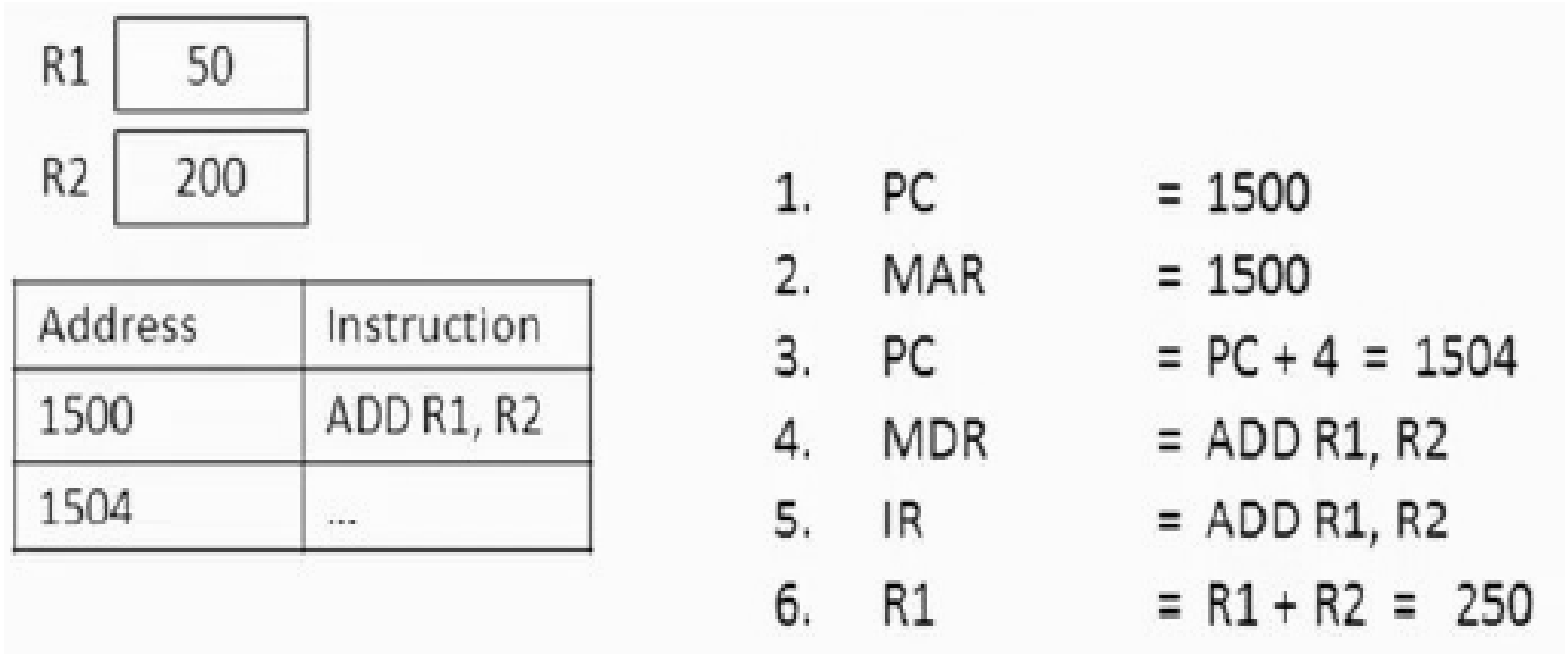
Micro-operations Instruction ADD R1, R2

- Suppose, ADD R1, R2 is saved in memory location 8000. Initial Value: R1=100 AND R2=150
- Before instruction is executed, PC=8000

• Content of PC is transferred to MAR.	**MAR ← PC**
• READ request is issued to memory.	
• 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**
• Instruction is decoded by the control unit.	
• R2 is added to r1.	**R1 ← R1+R2**

Micro-operations: Example

Micro-operations Instruction ADD R1, R2



R1	50
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

Bus Architecture

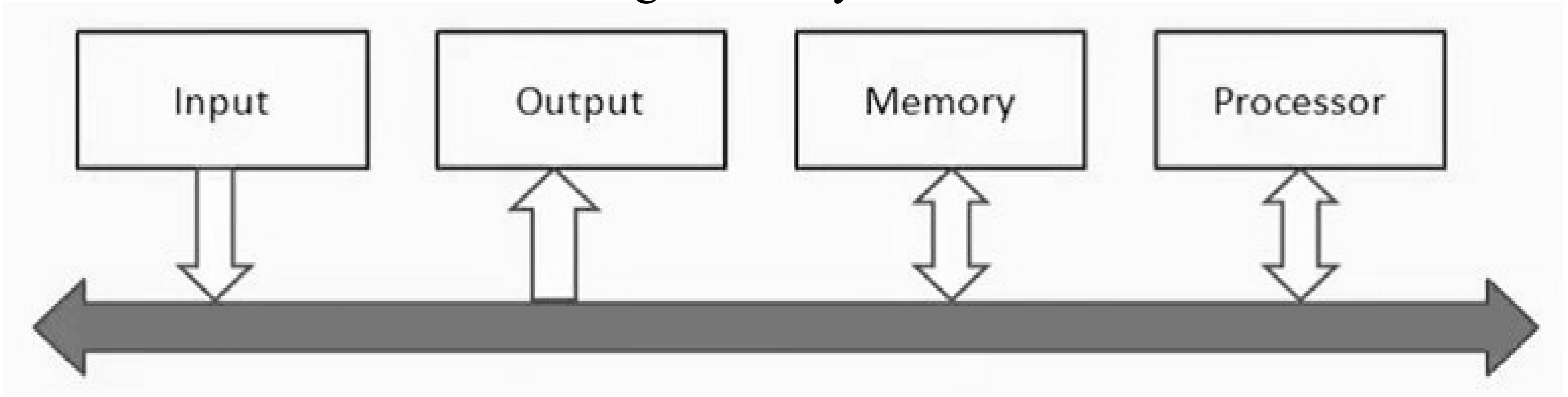
- Computer system consists of processor module, memory module, input output module.
- All these modules are communicating between each other.
- A communicating pathway is needed to communicate between each of these functional units. So, the different functional modules must be connected in an organized manner to form an operational system.
- Bus refers to a group of lines that serve as a connecting path for several devices. So, the simplest way to connect all these is through a single bus architecture, where only one data transfer will be allowed in one cycle.
- For multi bus architecture parallelism in data transfer is allowed.

System-Level Single Bus Architecture

- All the modules:-memory, processor, input and output are connected to a single bus. If the processor wants to communicate with memory, it has to use this bus and no other modules will be able to use at that moment.

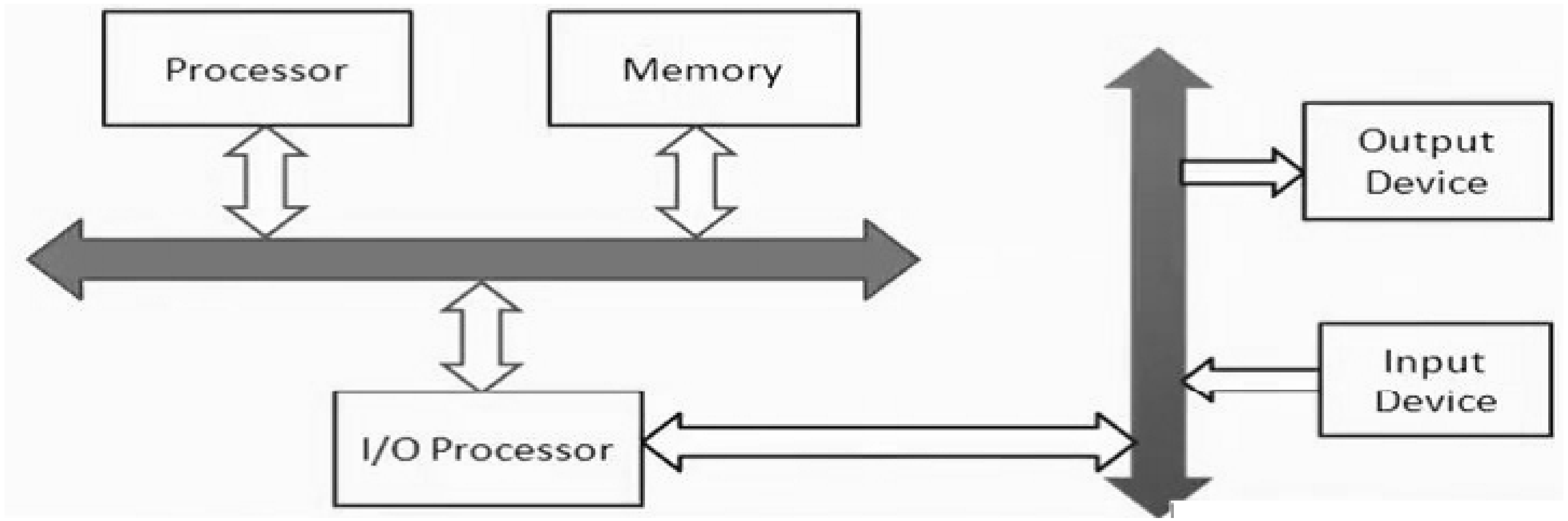
- In the same way if from memory something needs to be sent to output device, no other module can communicate.

- This is a bottleneck of a single level system bus.



System-Level Two-Bus Architecture

- System-level two-bus architecture
- There is a bus dedicated to processor memory and IO processor is also connected to it.
- For input and output, there is a separate bus and this bus will be communicating with the IO processor in turn and the IO processor will then be communicating with the processor or the memory as and when it is required. When there is more communication between processor and memory, then we must have a bus dedicated for this.

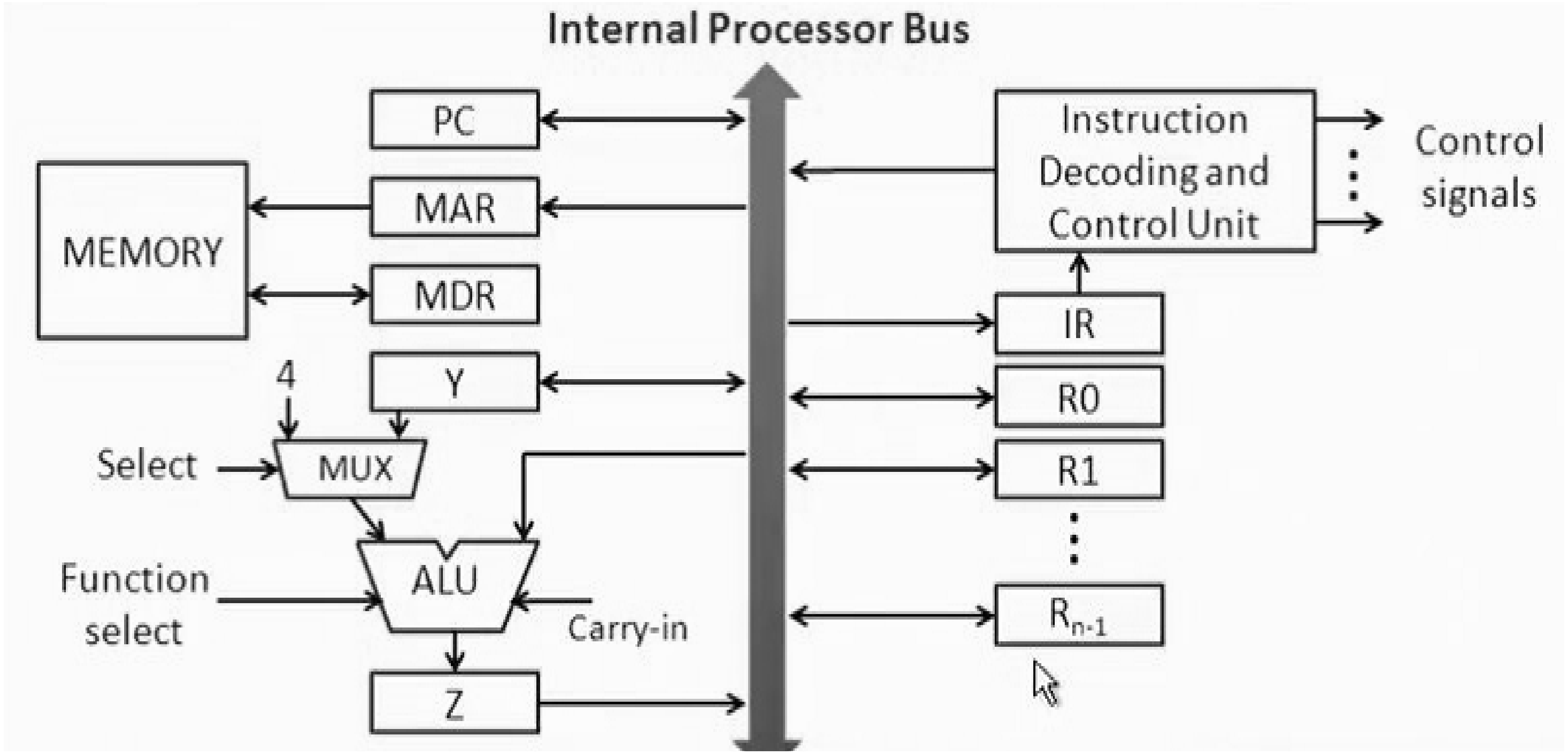


Single-Bus Architecture Inside the Processor

- 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: $PC \leftarrow PC + 4$.

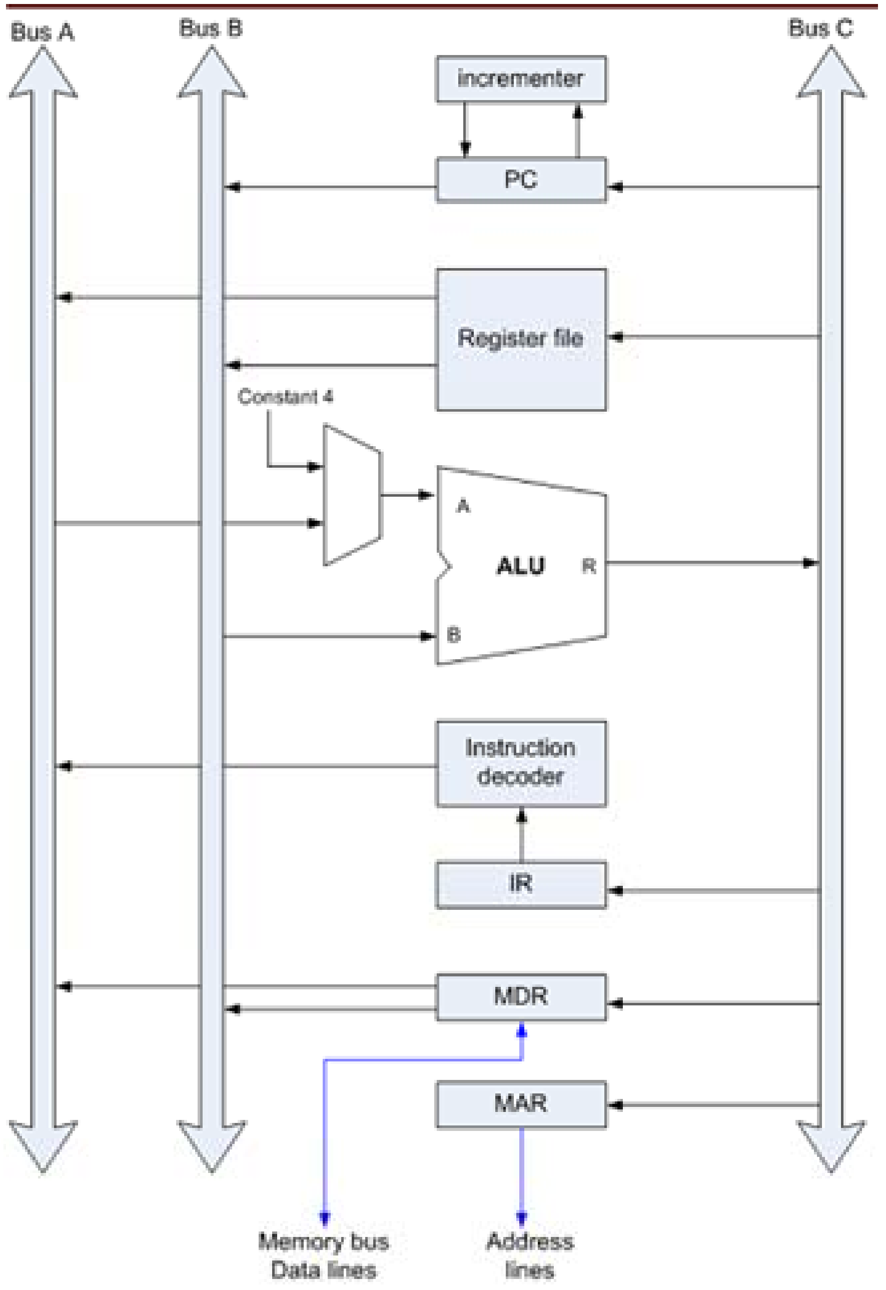
Single-Bus Architecture Inside the Processor

- This is an internal processor bus. All the elements: PC, MAR, MDR, Registers, ALU are inside the processor. Temporary registers Z, Y, general purpose registers, IR, instruction decoding and control unit are connected. Information is getting transferred within all these modules.
- A data transfer from R1 to R2 or from PC to MAR or from MDR to R1 or R2 or to ALU, will be done through this internal processes bus.
-



Multi-Bus Architecture

- Modern processors use the multi-bus architecture. Here, within the processor to communicate between various registers, we have multiple-bus.
- The advantage is that more operations can be performed and we get results much faster. So, there will be a overall improvement in instruction execution time.


Bus A
Bus B
Bus C
incrementer
PC
Register file
Constant 4
A
ALU
R
B
Instruction decoder
IR
MDR
MAR
Memory bus Data lines
Address lines