

Noida Institute of Engineering and Technology, Greater Noida

Introduction of Computer Organization

Unit: 1

Computer Organization &
Architecture

B Tech 3rd Sem



Pradeep kumar
CSE
Department



Unit 1

Introduction:

- **Functional units of digital system and their interconnections**
- **Buses, bus architecture**
- **Types of buses and bus arbitration**
- **Register, bus and memory transfer.**
- **Processor organization**
- **General registers organization**
- **Stack organization**
- **Addressing modes**

Course Objective

- To study the basic organization and architecture of digital computers (CPU, memory, I/O, software).

Course Outcome

- Study of the basic structure and operation of a digital computer system.

CO-PO Mapping

COMPUTER ORGANIZATION AND ARCHITECTURE (KCS-302)

CO.K	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
KCS-302.1	3	2	1	1	1	1	1	-	1	1	1	2

CO- PSO Mapping

COMPUTER ORGANIZATION AND ARCHITECTURE (KCS-302)				
CO.K	PSO1	PSO2	PSO3	PSO4
KCS-302.1	3	3	3	2

Prerequisite and Recap

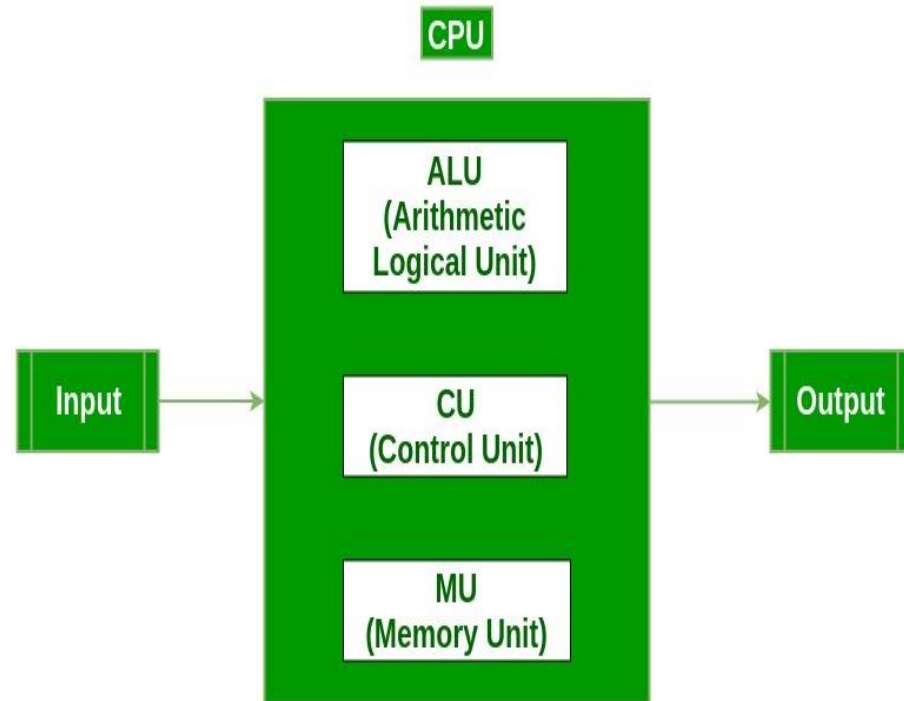
- **Basic knowledge of Digital Logic Design**
- **Basic Idea of various gates and circuit**

Computer Organization and Architecture

- **Computer Organization** is concerned with the way hardware components operate and the way they are connected together to form the **computer** system.
- **Computer Architecture** is concerned with the structure and behavior of a **computer** as seen by the user. It includes the information, formats, the instruction set, and techniques for addressing memory. Two basic types of **Computer Architectures** are
 - 1.**von Neuman architecture**:-Desktop personal computer
 - 2.**Harvard architecture**-Microcontroller based (Single chip microcomputer) computer systems and DSP based computer systems

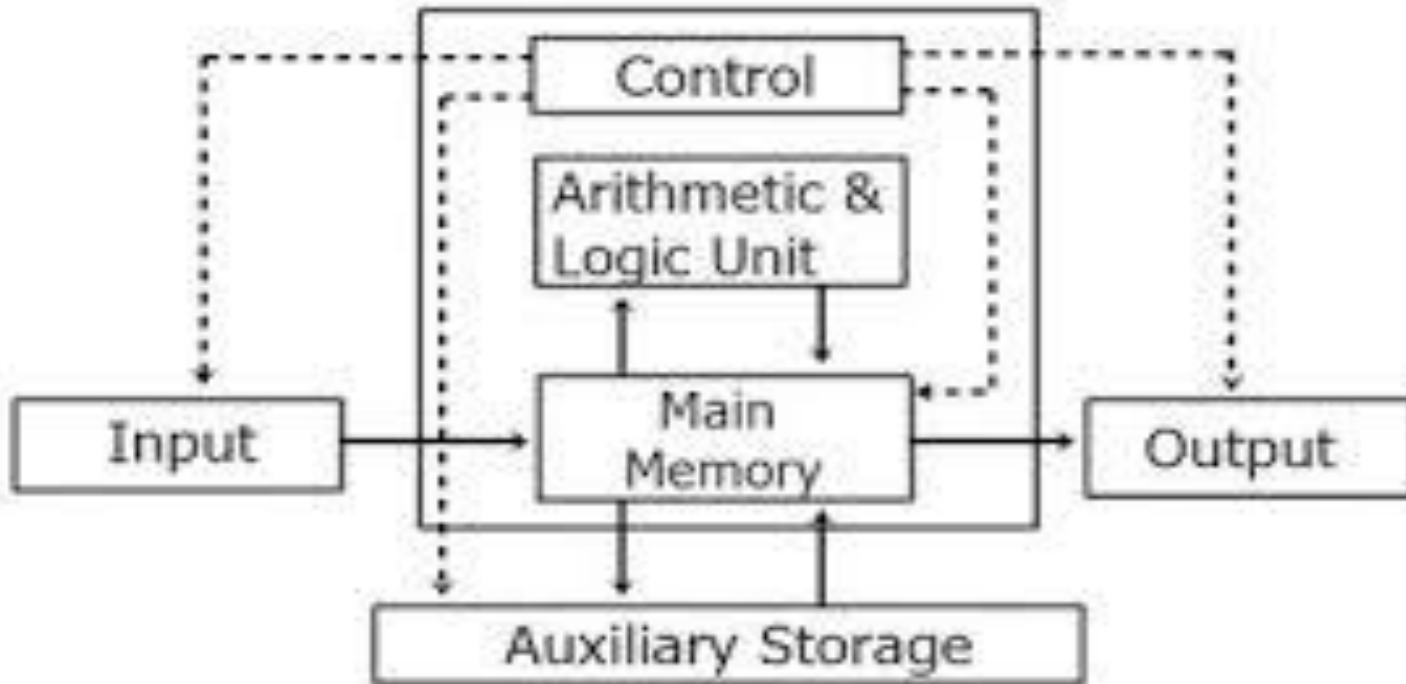
Functional Units of Computer System

- Input Ex. Keyboard, Mouse,...
- ALU
- Control Unit
- Memory Unit
- Output Ex. Printer, Speaker



Block Diagram

Interconnection of Functional unit



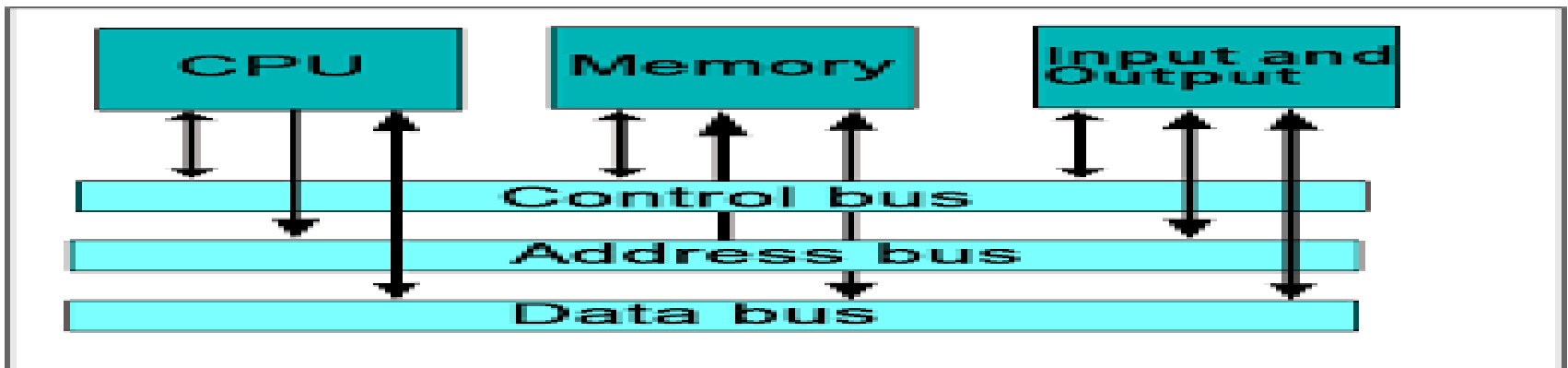
Block Diagram of Computer

- BUS : A **bus** is a **common pathway** through which information flows from one computer component to another.
- Types of Computer BUS:
 - ❖ **Data bus**
 - ❖ **Address Bus**
 - ❖ **Control Bus**
- ❖ The **data bus** is a bidirectional pathway that carries the actual data (information) to and from the main memory.
- ❖ The **control bus** carries the control and timing signals needed to coordinate the activities of the entire computer.
- ❖ The **address bus**, which is a unidirectional pathway that allows information to travel in only one direction

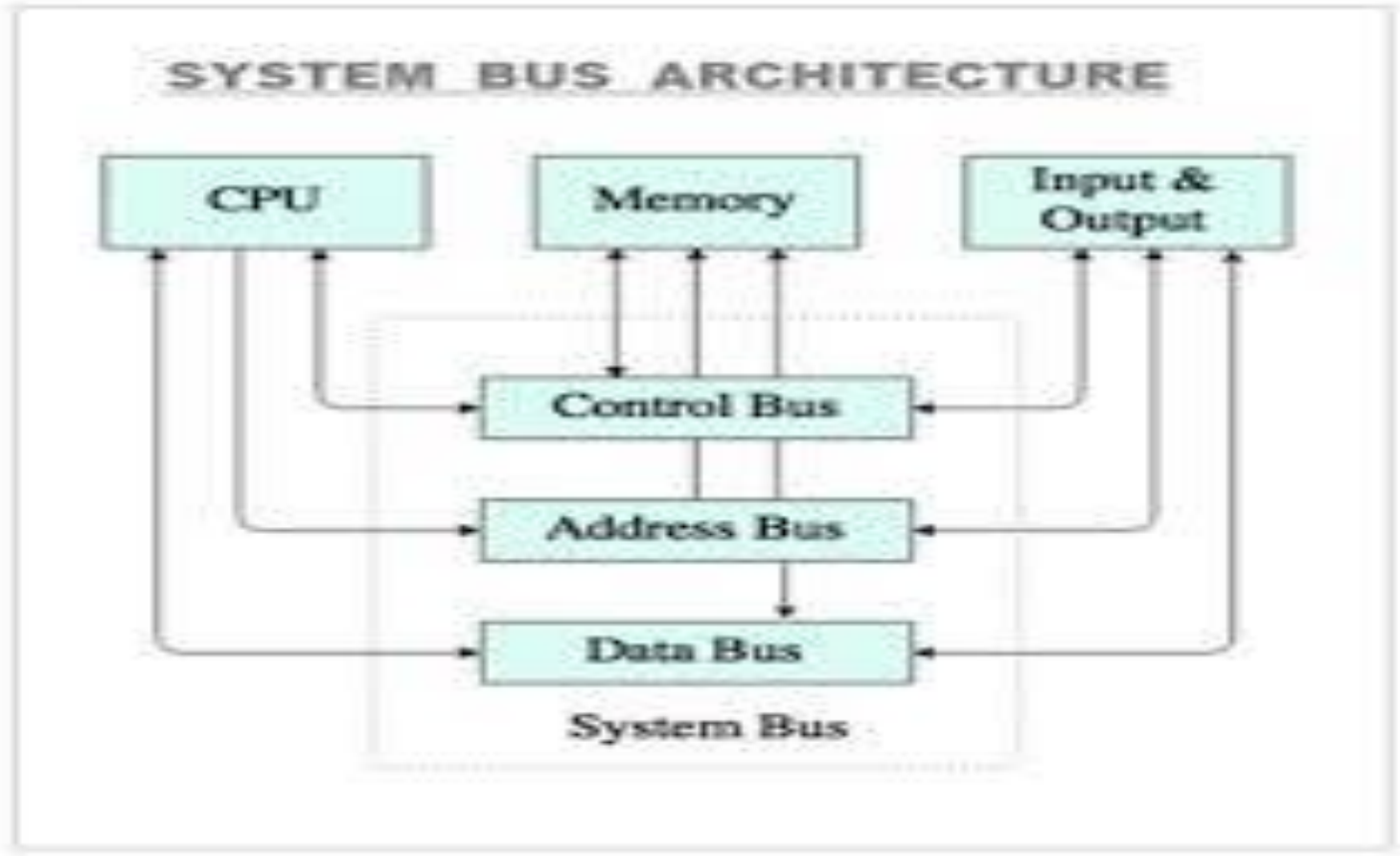
Types of Buses

Summary

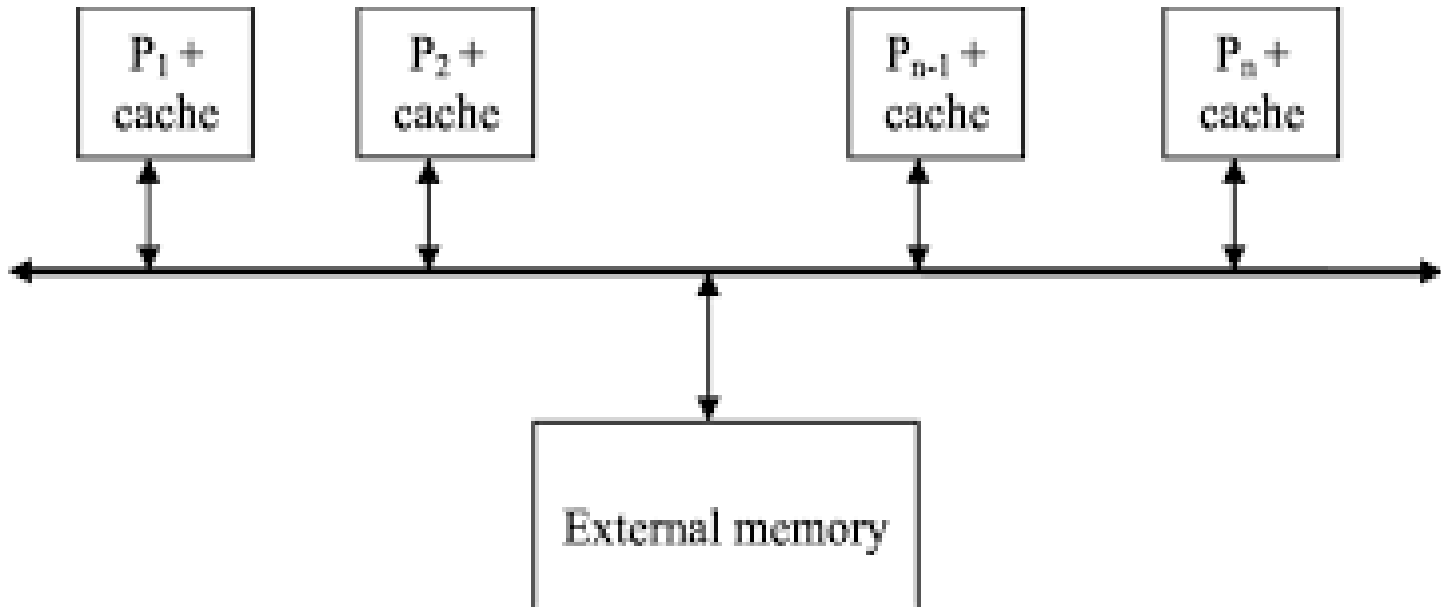
Bus Type	Description
Address bus	A unidirectional pathway – information can only flow one way
Data bus	A bi-directional pathway – information can flow in two directions
Control bus	Carries the control and timing signals needed to coordinate the activities of the entire computer



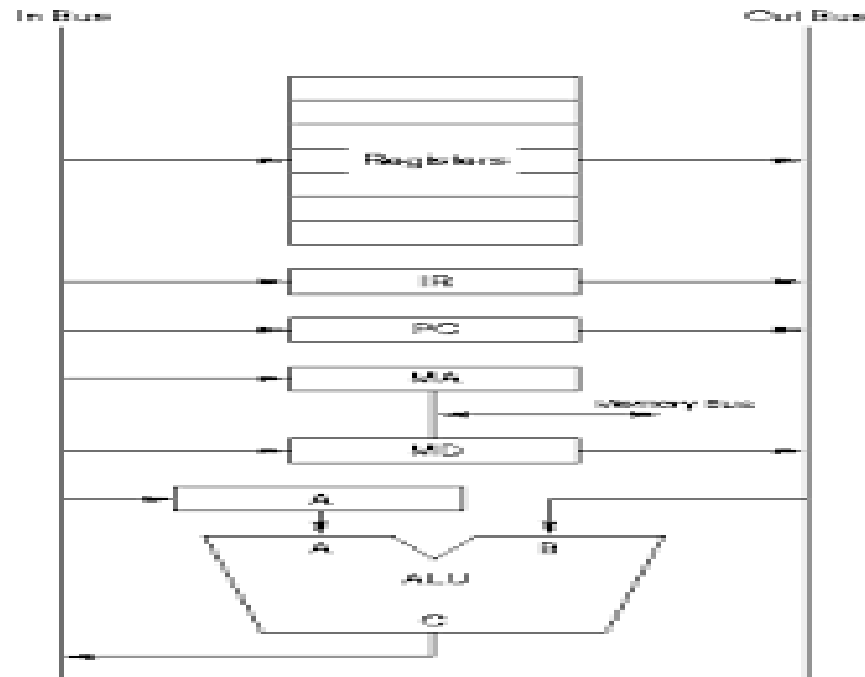
System Bus Architecture



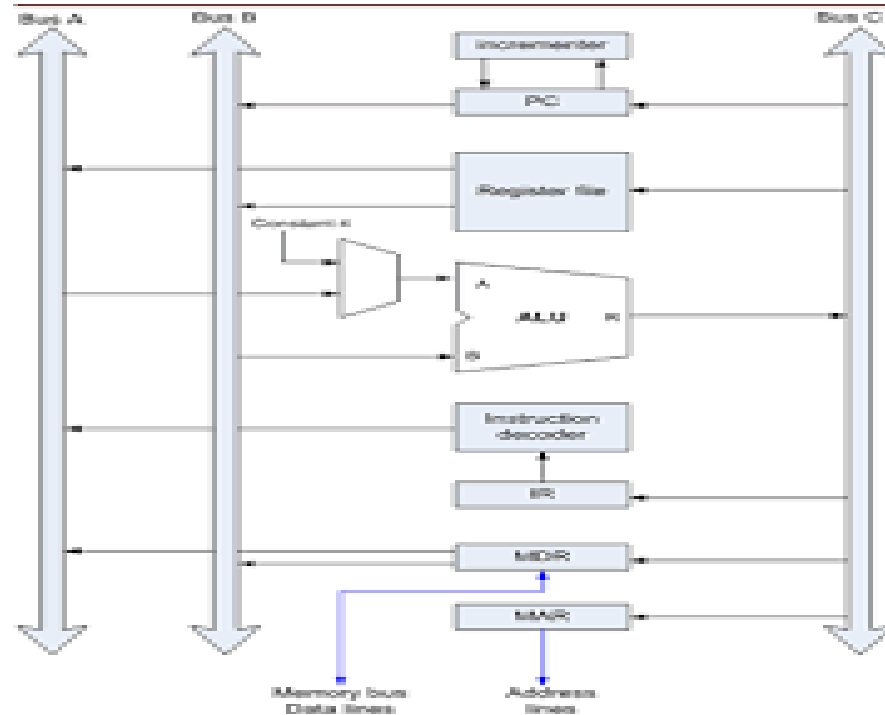
Single Bus shared memory Architecture



Two Bus Architecture



Multiple Bus Architecture



Bus Arbitration

- **Bus Arbitration** refers to the process by which the current bus master accesses and then leaves the control of the bus and passes it to another bus requesting processor unit.

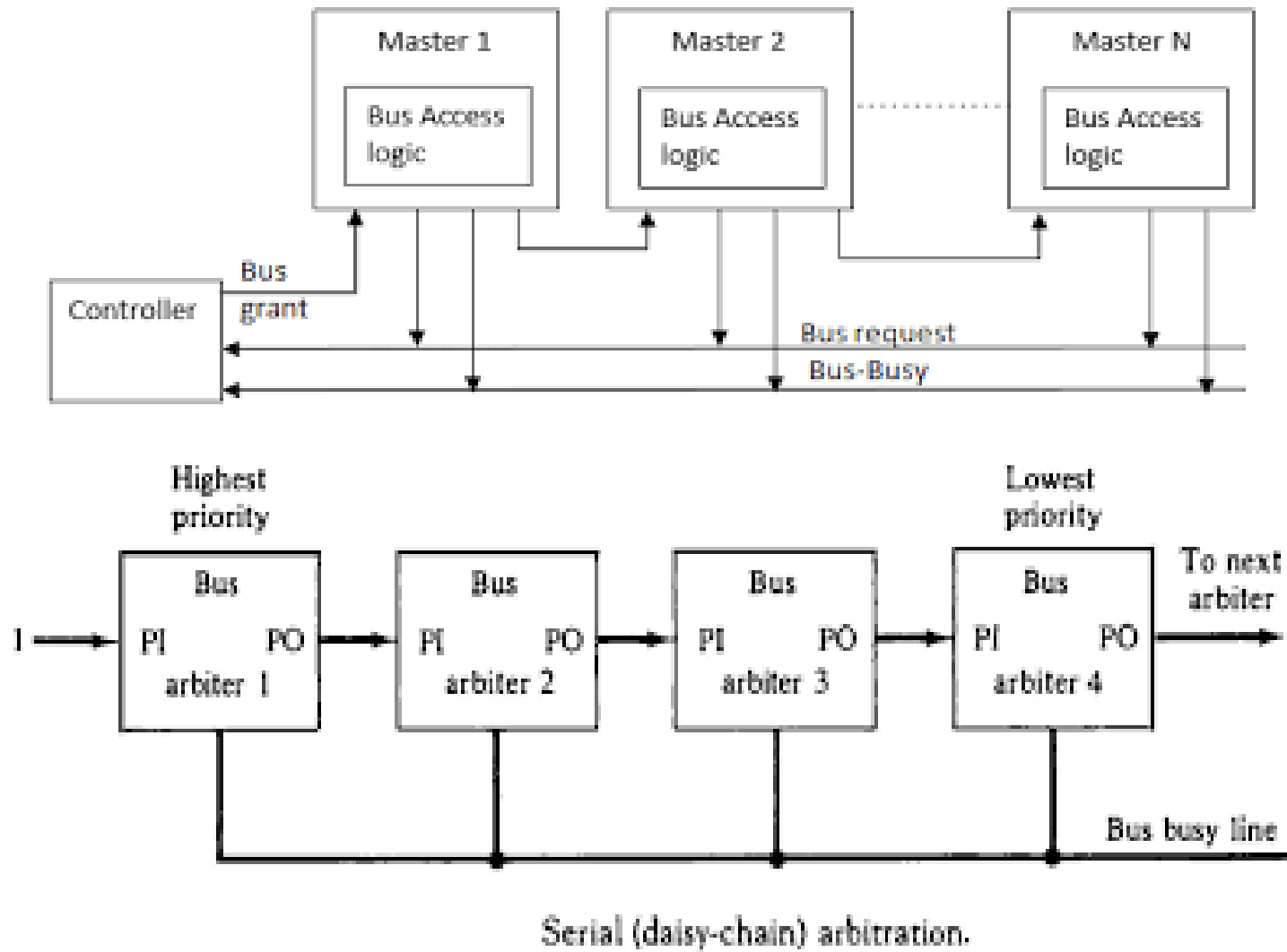
Bus master :The controller that has access to a bus at an instance . Eg DMA controller , I/O processor or processor

Bus Arbiter decides who would become current bus master. Bus arbiter can be processor or Bus controller

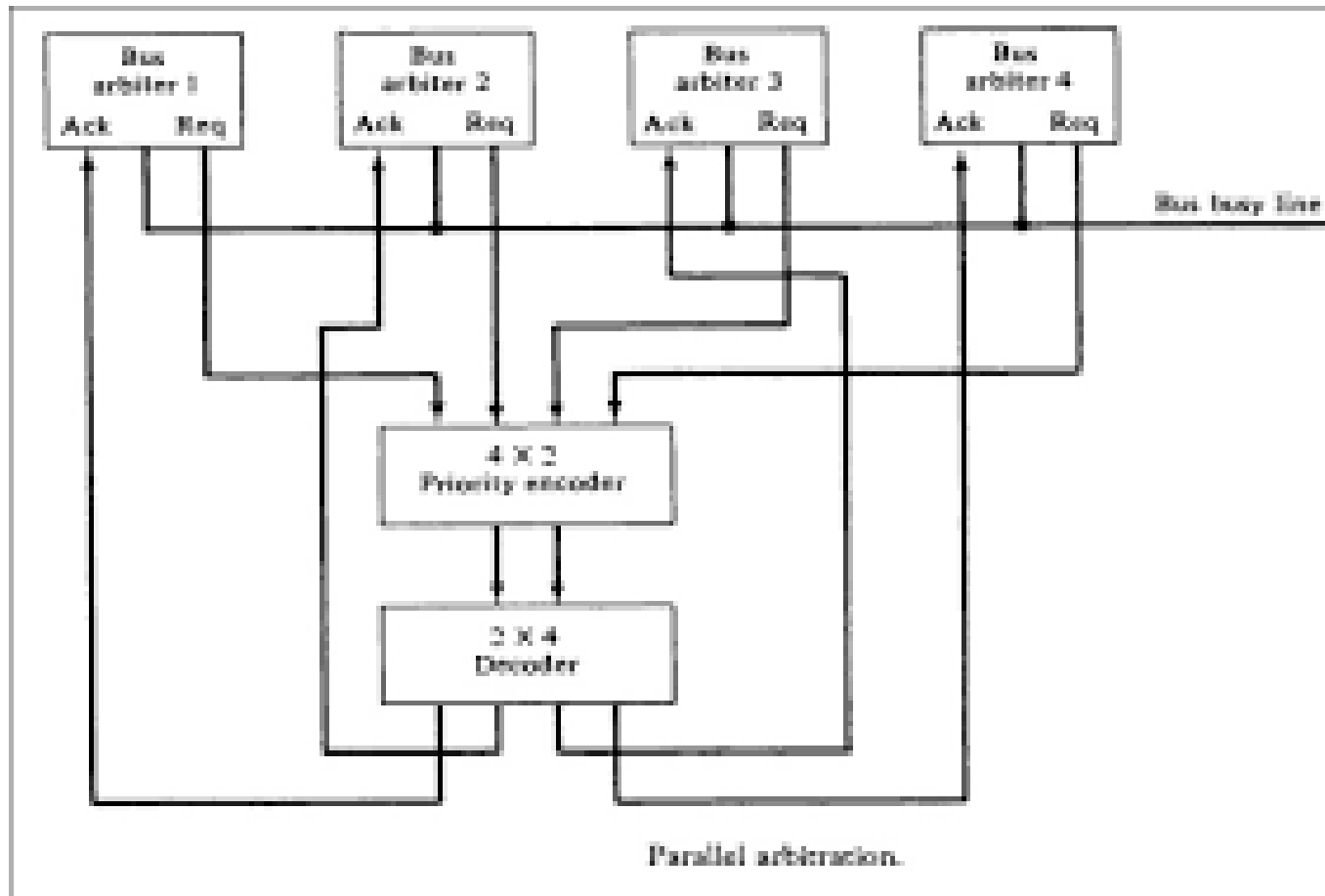
There are two types of bus arbitration priority algorithm:

- **Static priority arbitration algorithm**
 - a) Serial Arbitration
 - b) Parallel Arbitration
- **Dynamic priority arbitration algorithm**
 - a)Time Slice b) LRU c) FIFO d) Polling e)Rotating Daisy Chain

Serial (Daisy Chaining) method

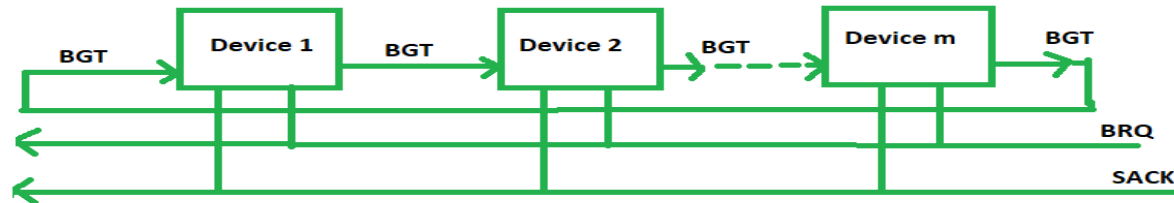


Parallel arbitration logic



Rotating Priority Method

Rotating Priority Method



Rotating priority bus arbitration

Advantages –

- This method does not favor any particular device and processor.
- The method is also quite simple.
- If one device fails then entire system will not stop working.

Disadvantages –

- Adding bus masters is difficult as increases the number of address lines of the circuit.

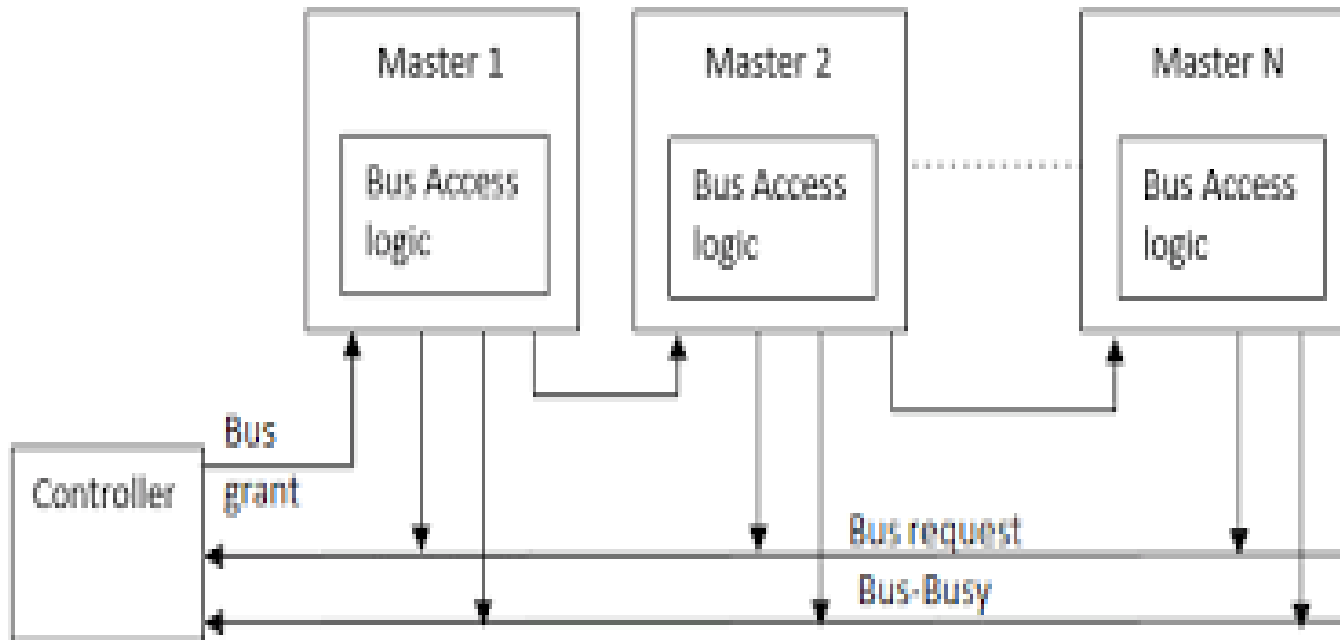
Schemes of BUS Arbitration:

a)Centralized b)Distributed

There are three bus arbitration schemes of centralized bus arbitration approach :

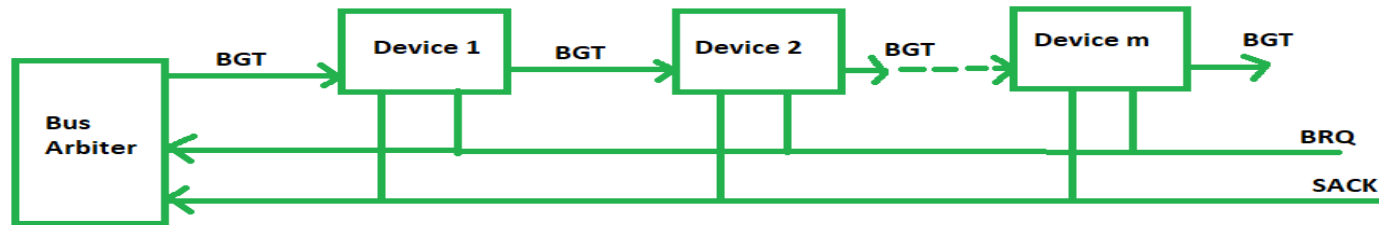
- 1. Serial (Daisy Chaining) method**
- 2. Polling method**
- 3. Independent Request method**

Serial (Daisy Chaining) method



Serial (Daisy Chaining) method

Daisy Chaining method



Daisy chained bus arbitration

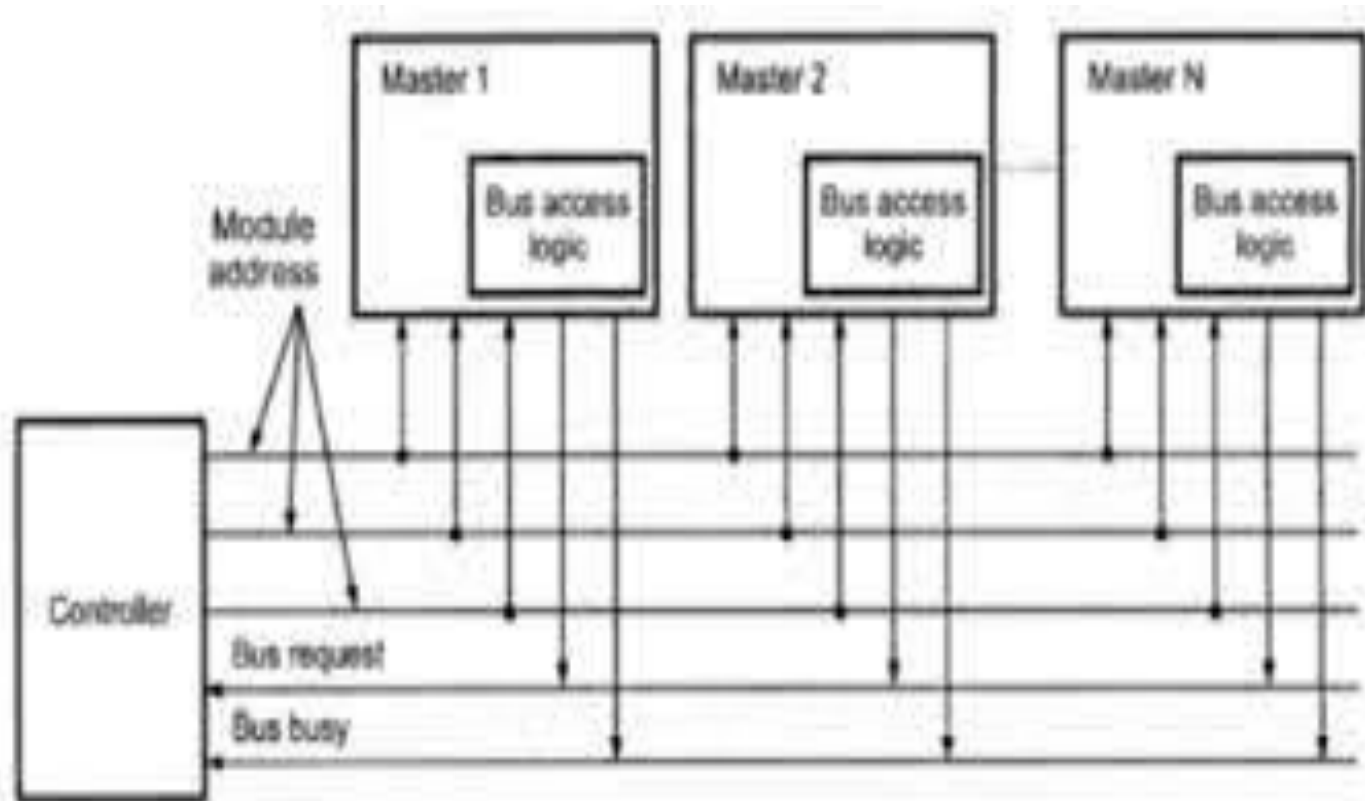
Advantages –

- Simplicity and Scalability.
- The user can add more devices anywhere along the chain

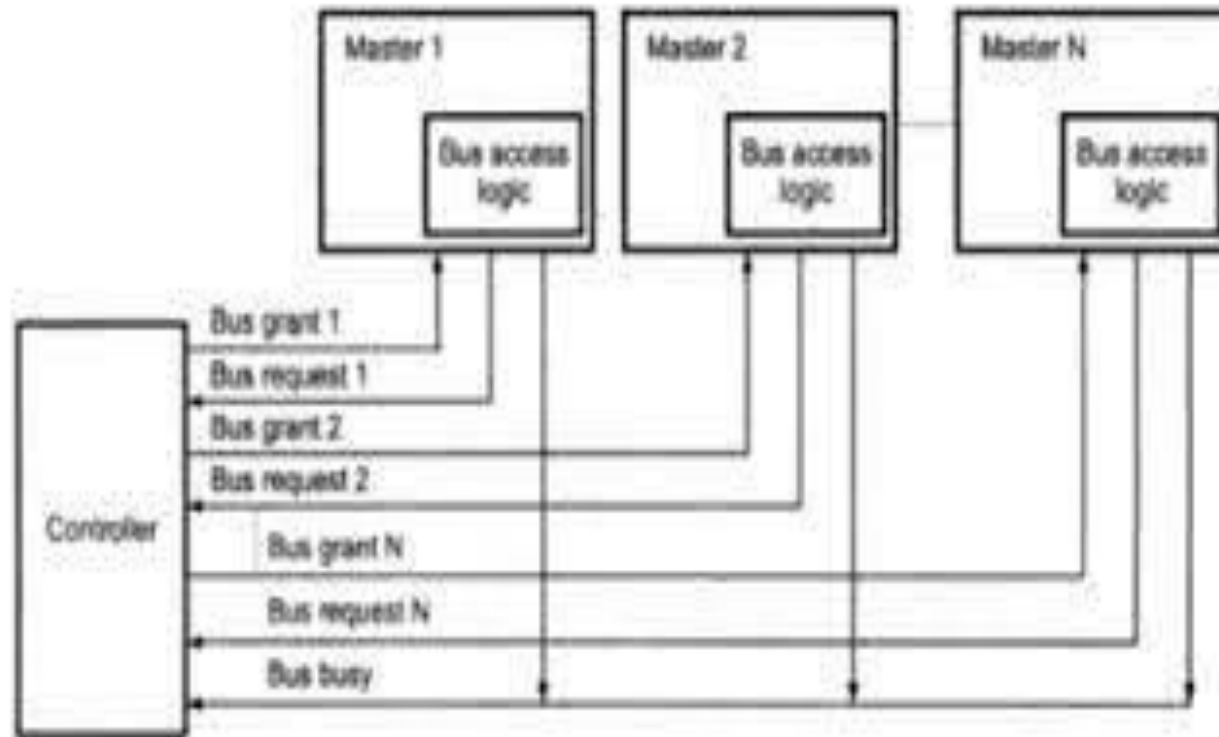
Disadvantages –

- The value of priority assigned to a device is depends on the position of master bus.
- Propagation delay is arises in this method.
- If one device fails then entire system will stop working.

polling method

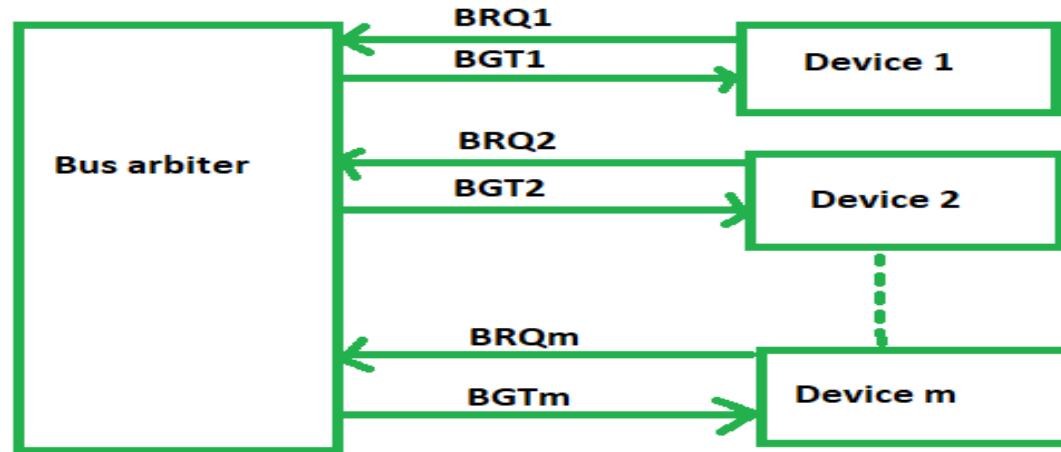


Independent Request method



Independent Request method

Independent Request method



Fixed priority bus arbitration method

Advantages –

- This method generates fast response.

Disadvantages –

- Hardware cost is high as large no. of control lines are required.

You Tube link

https://www.youtube.com/watch?v=Xi_jRzyq42g

Distributed method

- In this, all devices participate in the selection of the next bus master. Each device on the bus is assigned a 4bit identification number. The priority of the device will be determined by the generated ID

Register Transfer Language(RTL)

Register Transfer Language:

- ❖ A digital system is an interconnection of digital hardware modules that perform a specific task.
- ❖ The modules are constructed from digital components such as registers, decoders, arithmetic elements, and control logic.
- ❖ These modules are interconnected with common data and control paths to form a digital computer.
- ❖ The operations executed on data stored in registers are called microoperations.
- ❖ A microoperation is an elementary operation performed on the information stored in one or more registers.
- ❖ For any function of the computer, a sequence of microoperations is used to describe it
- ❖ The result of the operation may be
 - ❖ replace the previous binary information of a register or
 - ❖ transferred to another register
- ❖ Examples of microoperations are shift, count, clear and load.

Register Transfer Language(RTL)

- RTL is a system for expressing in symbolic form the micro operations sequences among the registers of a digital module. The symbolic notation used to describe the micro operation transfer among registers is called RTL.
- Register transfer implies the availability of hardware logic circuits that can perform a stated micro operation and transfer the result of the operation to the same or another registers
- The word language is a procedure for writing symbols to specify a given computational process

Register Transfer Language(RTL)

Register Transfer & μ -operations

3

Register Transfer Language

REGISTER TRANSFER LANGUAGE

The internal hardware organization of a digital computer is best defined by specifying :

- The set of registers it contains and their functions
- The sequence of microoperations performed on the binary information stored
- The control that initiates the sequence of microoperations

For any function of the computer, a sequence of microoperations is used to describe it

----> *Register transfer language*

- A symbolic language
- A convenient tool for describing the internal organization of digital computers
- Can also be used to facilitate the design process of digital systems.

Computer Organization

Computer Architectures Lab

Register Transfer Language

Register Transfer Language (RTL)

HIGH LEVEL LANGUAGE
Example : C++, VB, JAVA

ASSEMBLY LANGUAGE
Example : uP and uC

OPCODE

MICROCODE

RTL

- to represent registers
- specify the operations on their contents

Microcode (Micro-operations):
Operations executed on data stored in registers, performed in one clock cycle

Register Transfer Language (RTL):
Symbolic notation used to describe micro-operations

Register, bus and memory transfer

Register

Register are used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU. There are various types of registers which are used for different purpose. Eg. MAR(Memory address register),Memory Data Register(MDR),**Index register** ,**Memory Buffer Register**.

Register Symbol	Number of bits	Register Name	Register Function
DR	16	Data register	Holds memory operands
AR	12	Address register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction register	Holds instruction code
PC	12	Program counter	Holds address of instruction
TR	16	Temporary register	Holds temporary data
INPR	8	Input register	Holds input character
OUTR	8	Output register	Holds output character

Register, bus and memory transfer

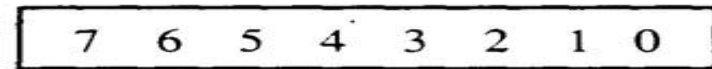
Register Transfer

- Designate computer registers by capital letters to denote its function
- The register that holds an address for the memory unit is called MAR
- The program counter register is called PC
- IR is the instruction register and R1 is a processor register

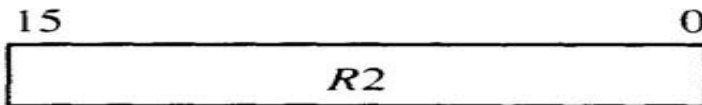
Block diagram of register.



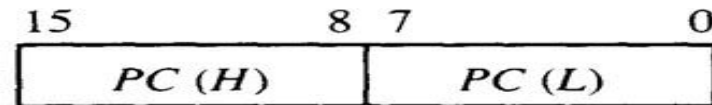
(a) Register *R*



(b) Showing individual bits



(c) Numbering of bits



(d) Divided into two parts

Register, bus and memory transfer

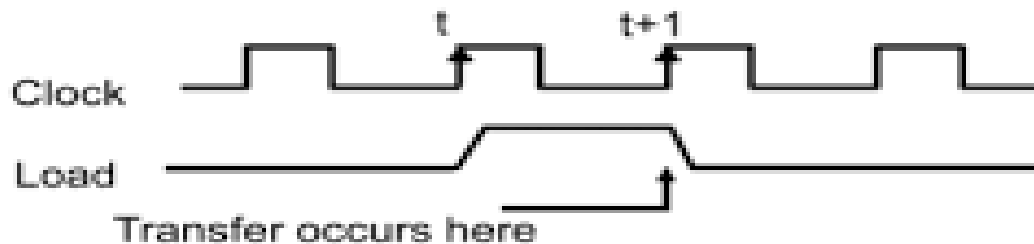
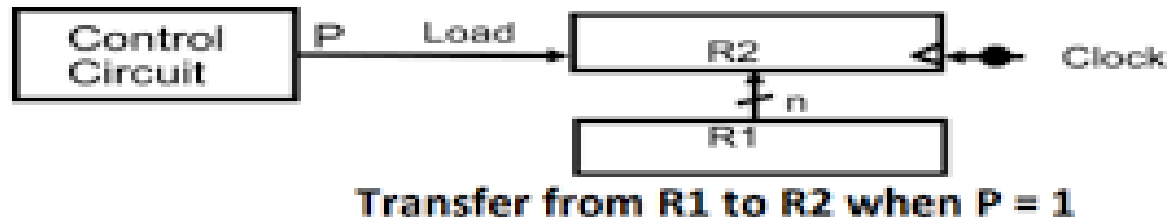
Register Transfer with block diagram

If the Register transfer is to occur only under a predetermined control condition, designate it by

or, *If* ($P = 1$) *then* $(R2 \leftarrow R1)$

P: $R2 \leftarrow R1$

where P is a control function that can be either 0 or 1



Timing diagram

Register, bus and memory transfer

Basic Symbols for Register Transfers

Symbol	Description	Examples
Letters (and numerals)	Denotes a register	<i>MAR, R2</i>
Parentheses ()	Denotes a part of a register	<i>R2(0–7), R2(L)</i>
Arrow \leftarrow	Denotes transfer of information	<i>R2 \leftarrow R1</i>
Comma ,	Separates two microoperations	<i>R2 \leftarrow R1, R1 \leftarrow R2</i>




Register, bus and memory transfer

Bus Transfer

- A bus structure consists of a set of common lines, one for each bit of a register
- Control signals determine which register is selected by the bus during each transfer
- Multiplexers can be used to construct a common bus
- Multiplexers select the source register whose binary information is then placed on the bus
- The select lines are connected to the selection inputs of the multiplexers and choose the bits of one register
- In general, a bus system will multiplex k registers of n bits each to produce an n -line common bus

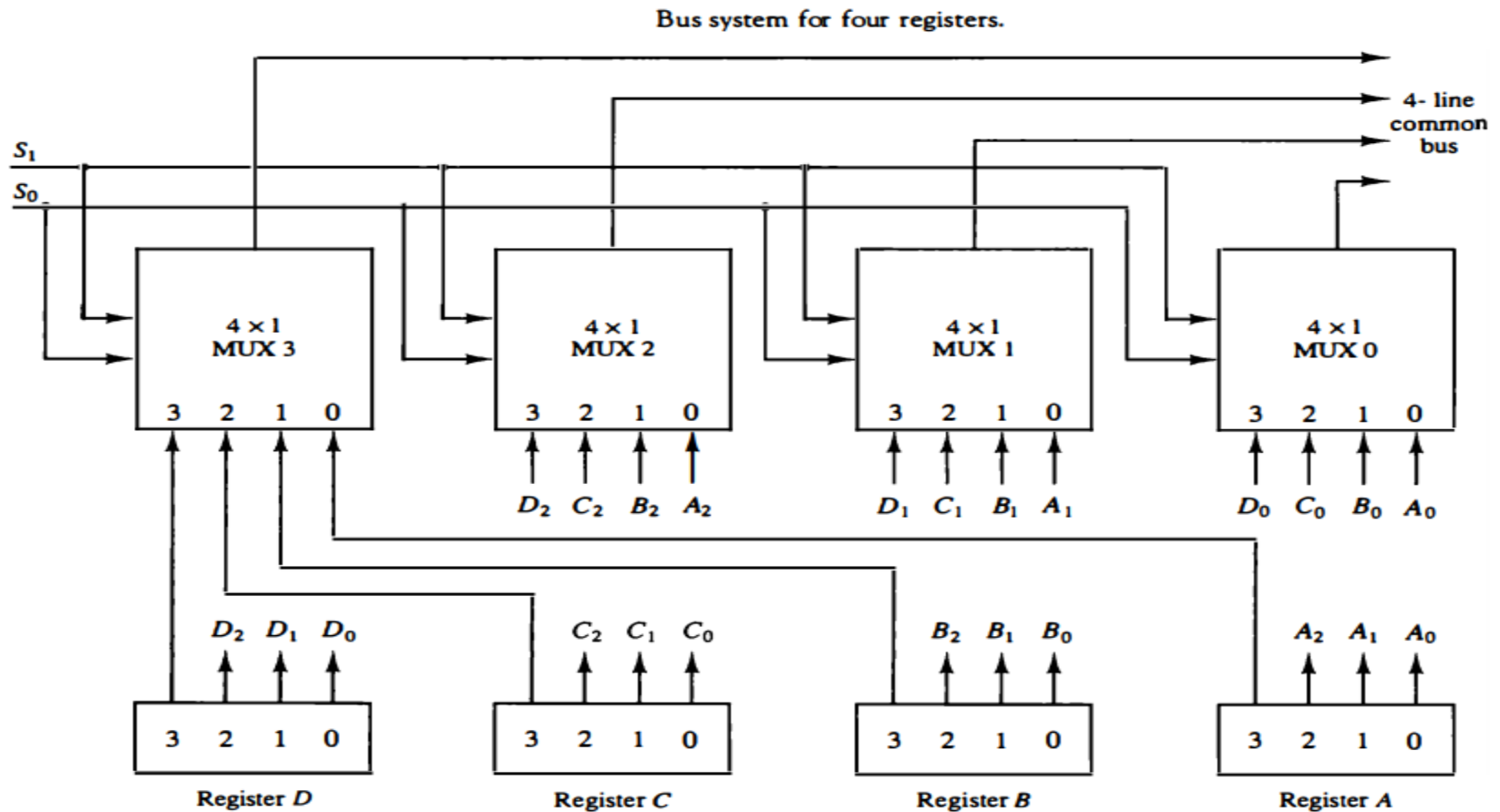
Register, bus and memory transfer

- This requires n multiplexers – one for each bit
- The size of each multiplexer must be $k \times 1$
- The number of select lines required is $\log k$
- To transfer information from the bus to a register, the bus lines are connected to the inputs of all destination registers and the corresponding load control line must be activated

BUS  C, R1  BUS, use R1  C, since the bus is implied

Register, bus and memory transfer

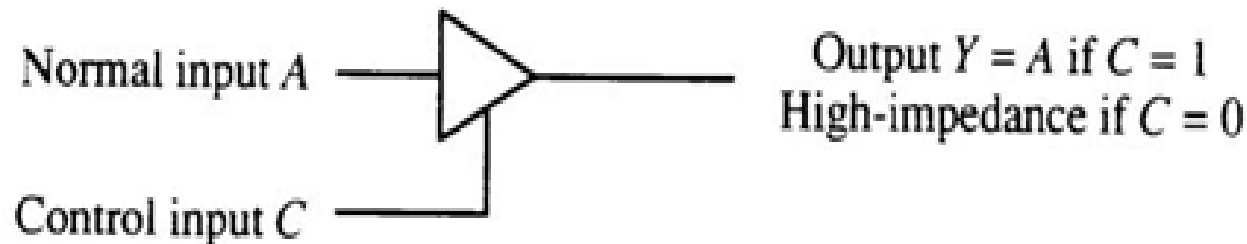
Bus Transfer



Bus Transfer using Three state buffer

- Instead of using multiplexers, *three-state gates* can be used to construct the bus system
- A three-state gate is a digital circuit that exhibits three states
- Two of the states are signals equivalent to logic 1 and 0
- The third state is a *high-impedance* state – this behaves like an open circuit, which means the output is disconnected and does not have a logic significance.

Graphic symbols for three-state buffer.



Bus Transfer using Three state buffer

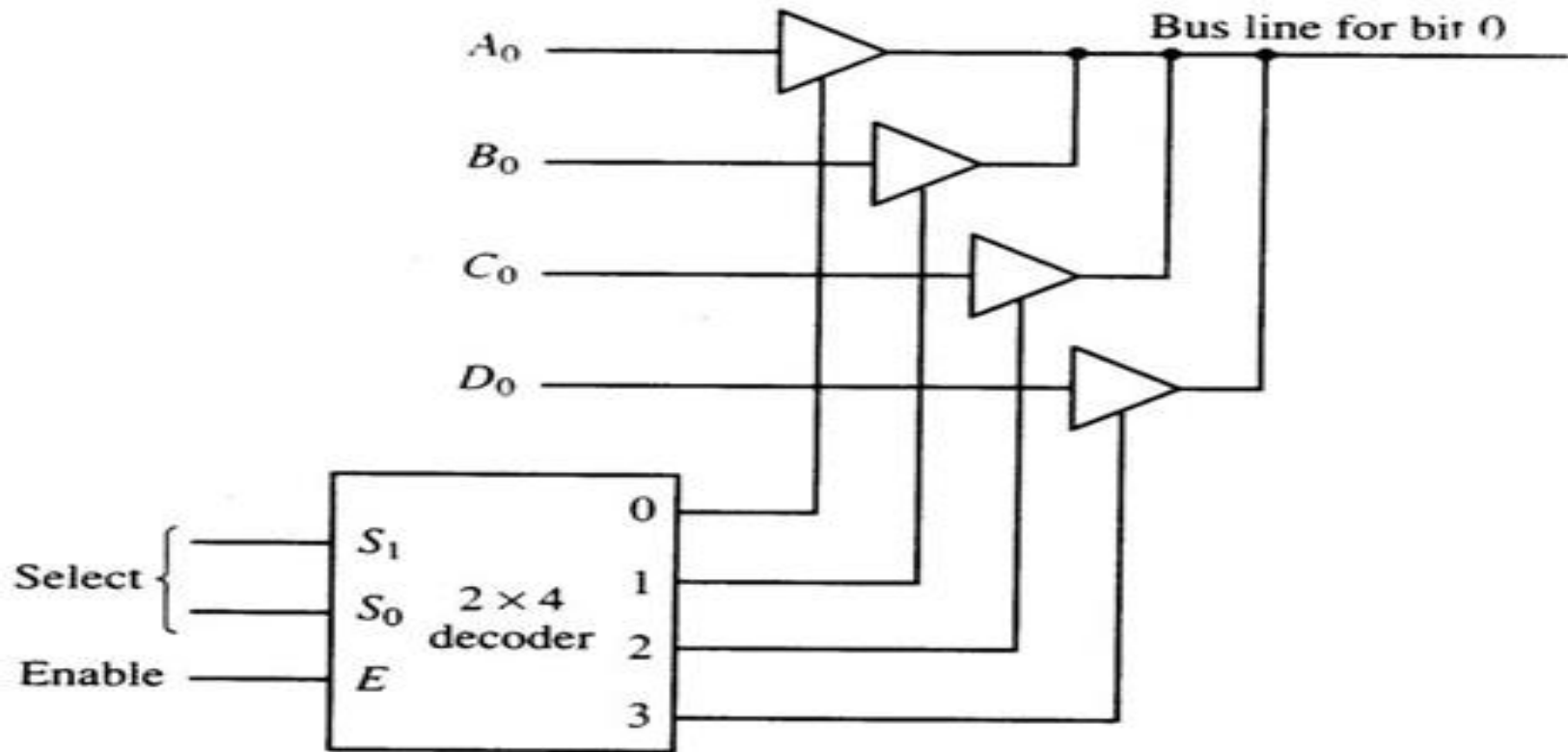
- The three-state buffer gate has a normal input and a control input which determines the output state.
- With control 1, the output equals the normal input
- With control 0, the gate goes to a high-impedance state
- This enables a large number of three-state gate outputs to be connected with wires to form a common bus line without endangering loading effects

Bus Transfer using Three state buffer

- Decoders are used to ensure that no more than one control input is active at any given time
- This circuit can replace the multiplexer in Figure 4.3
- To construct a common bus for four registers of n bits each using three-state buffers, we need n circuits with four buffers in each
- Only one decoder is necessary to select between the four registers
-
- Designate a memory word by the letter M
- It is necessary to specify the address of M when writing memory transfer operations
- Designate the address register by AR and the data register by DR
- The read operation can be stated as: Read: $DR \leftarrow M[AR]$
- The write operation can be stated as: Write: $M[AR] \leftarrow R1$

Register, bus and memory transfer

Bus Transfer using Three state buffer



Bus line with three state-buffers.

Register, bus and memory transfer

Most of the standard notations used for specifying operations on memory transfer are stated below.

- The transfer of information from a memory unit to the user end is called a **Read** operation.
- A memory word is designated by the letter **M**.
- We must specify the address of memory word while writing the memory transfer operations.
- The **address register** is designated by **AR** and the **data register** by **DR**.
- Thus, a read operation can be stated as:
1.Read: $DR \leftarrow M[AR]$

Register, bus and memory transfer

- The **Read** statement causes a transfer of information into the data register (DR) from the memory word (M) selected by the address register (AR).

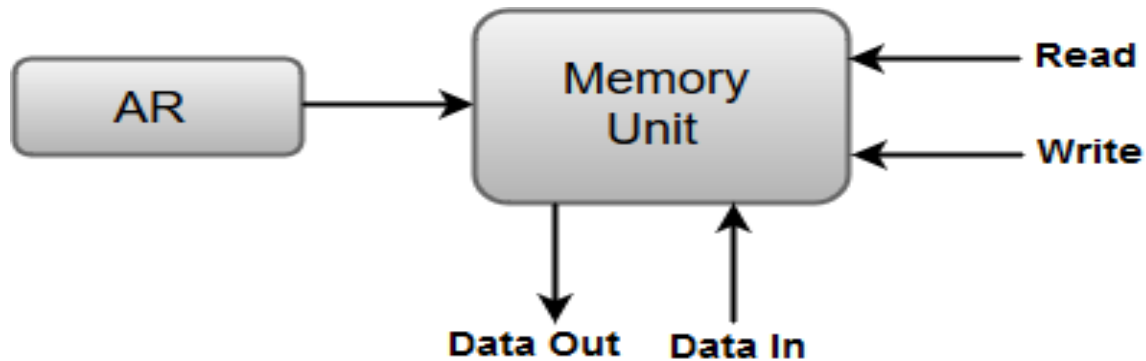
And the corresponding write operation can be stated as:

- The Write statement causes a transfer of information from register R1 into the memory word (M) selected by address register (AR).
- The transfer of new information to be stored in the memory is called a **Write** operation.

1. Write: $M[AR] \leftarrow R1$

Register, bus and memory transfer

Memory Transfer Block diagram



Above Diagram showing connections to memory unit.

Write: $M[AR] \leftarrow DR$

Read: $DR \leftarrow M[AR]$

Memory Transfer

- Designate a memory word by the letter M
- It is necessary to specify the address of M when writing memory transfer operations
- Designate the address register by AR and the data register by DR
- The read operation can be stated as: Read: $DR \leftarrow M[AR]$
- The write operation can be stated as: Write: $M[AR] \leftarrow R1$

Processor organization-General register organization

Processor organization

Most computers fall into one of three types of CPU organizations:

- Single accumulator organization.
- General register organization.
- Stack organization.

Single accumulator organization

The instruction format in this type of computer uses one address field

Example

ADD X

- where X is the address of the operand. The ADD instruction in this case results in the operation $AC \leftarrow AC + M[X]$.
- AC is the accumulator register and $M[X]$ symbolizes the memory word located at address X.

General register organization

General register organization

The instruction format in this type of computer needs three register address fields.

ADD R1, R2, R3 to denote the operation $R1 \leftarrow R2 + R3$.

ADD R1, R2, would denote the operation $R1 \leftarrow R1 + R2$. Only register addresses for R1 and R2 need be specified in this instruction.

General register-type computers employ two or three address fields in their instruction format.

Stack organization

The stack-organized CPU, Computers with stack organization would have PUSH and POP instructions which require an address field. Thus the instruction

PUSH X

PUSH Y

ADD

POP Z

General registers organization

- In this type of organization, computer uses two or three address fields in their instruction format.
- Each address field may specify a general register or a memory word. If many CPU registers are available for heavily used variables and intermediate results
- For example:
MULT R1, R2, R3
- This is an instruction of an arithmetic multiplication written in assembly language. It uses three address fields R1, R2 and R3.

General register organization

- The meaning of this instruction is:

$R1 \leftarrow R2 * R3$

- This instruction also can be written using only two address fields as:

MULT R1, R2

- In this instruction, the destination register is the same as one of the source registers. This means the operation

$R1 \leftarrow R1 * R2$

- The use of large number of registers results in short program with limited instructions.

General register organization

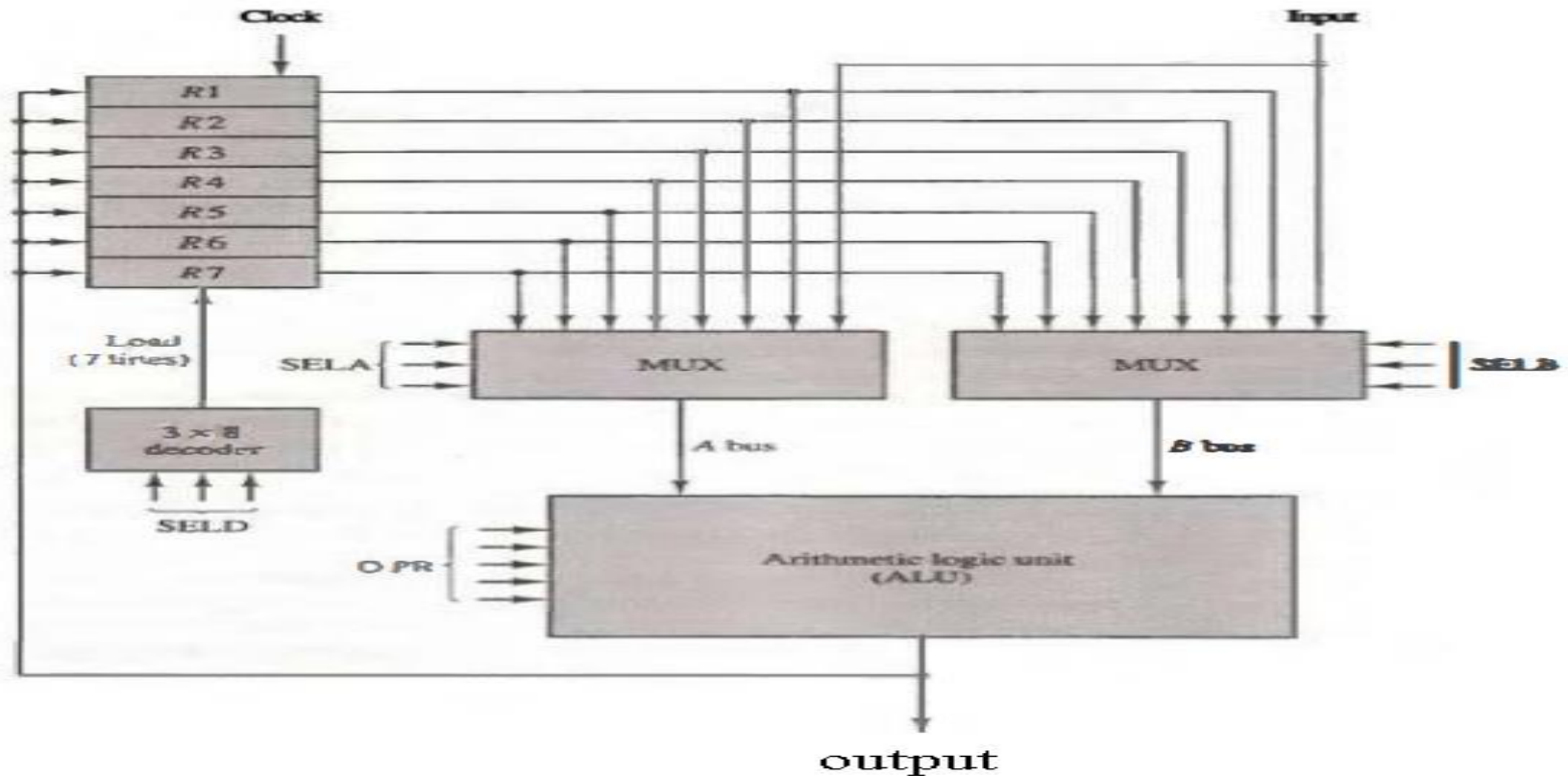
The advantages of General register based CPU organization

- Efficiency of CPU increases as there are large number of registers are used in this organization.
- Less memory space is used to store the program since the instructions are written in compact way.

The disadvantages of General register based CPU organization

- Care should be taken to avoid unnecessary usage of registers. Thus, compilers need to be more intelligent in this aspect.
- Since large number of registers are used, thus extra cost is required in this organization.

General register organization



(a) Block diagram



(b) Control word

Register set with common ALU.

General register organization

The control unit that operates the CPU bus system directs the information flow through the registers and ALU by selecting the various components in the system. For example, to perform the operation $R1 \leftarrow R2 + R3$. The control must provide binary selection variables to the following selector inputs:

1. MUX A selector (SELA): to place the content of R2 into bus A .
- 2 . MUX B selector (SELB): to place the content of R3 into bus B .
- 3 . ALU operation selector (OPR): to provide the arithmetic addition A+B.

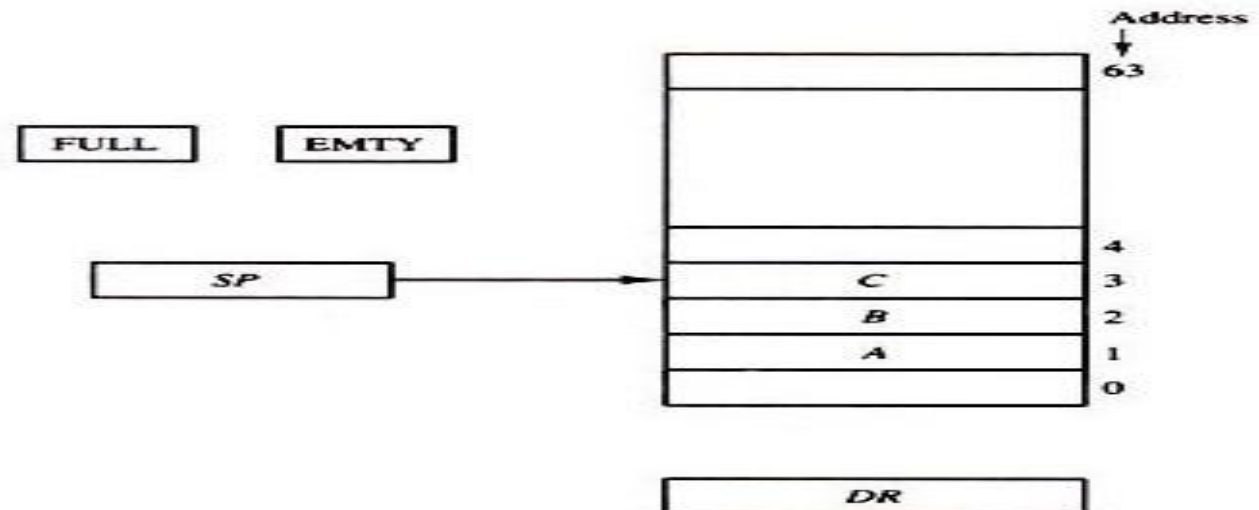
Decoder destination selector (SELD): to transfer the content of the output bus into R 1 .

Field:	SELA	SELB	SELD	OPR
Symbol:	R2	R3	R1	ADD
Control word:	010	011	001	00010

Stack Organization

- A stack is a storage device that stores information in such a manner that the item stored last is the first item retrieved.
- The stack in digital computers is essentially a memory unit with an address register that can count only. The register that holds the address for the stack is called a stack pointer (SP) because its value always points at the top item in the stack.
- The physical registers of a stack are always available for reading or writing. It is the content of the word that is inserted or deleted.

Register stack:



PUSH:

If the stack is not full ($FULL = 0$), a new item is inserted with a push operation. The push operation consists of the following sequences of micro operations:

$SP \leftarrow SP + 1$ Increment stack pointer

$M[SP] \leftarrow DR$ WRITE ITEM ON TOP OF THE STACK

IF ($SP = 0$) then ($FULL \leftarrow 1$) Check if stack is full

$EMPTY \leftarrow 0$ Mark the stack not empty

POP:

A new item is deleted from the stack if the stack is not empty (if $EMPTY = 0$). The pop operation consists of the following sequences of micro operations:

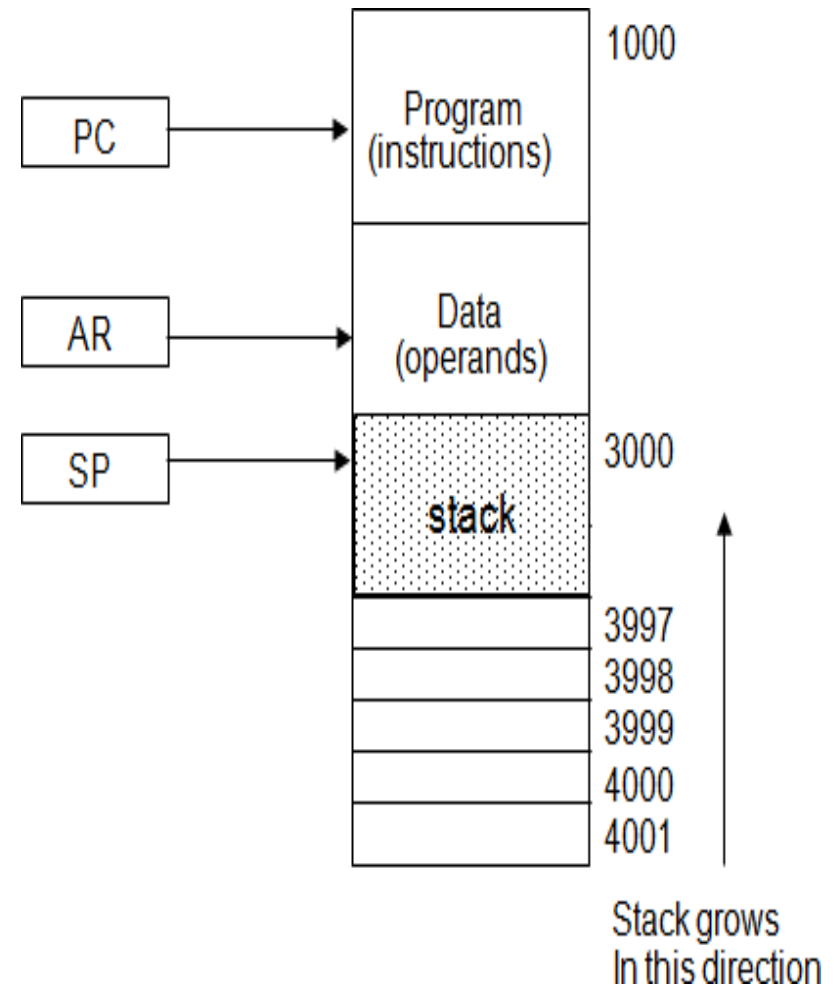
$DR \leftarrow M[SP]$	Read item on top of the stack
$SP \leftarrow SP - 1$	Decrement stack pointer
IF $(SP = 0)$ then $(EMPTY \leftarrow 1)$	Check if stack is empty
$FULL \leftarrow 0$	Mark the stack not full

The top item is read from the stack into DR. The stack pointer is then decremented. If its value reaches zero, the stack is empty, so EMPTY is set to 1.

Stack Organization

Memory Stack

- The implementation of a stack in the CPU is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer.
- The program counter PC points at the address of the next instruction in the program which is used during the fetch phase to read an instruction.



Memory Stack

- The address registers AR points at an array of data which is used during the execute phase to read an operand.
- The stack pointer SP points at the top of the stack which is used to push or pop items into or from the stack.
- The three registers are connected to a common address bus, and either one can provide an address for memory.

PUSH

A new item is inserted with the push operation as follows:

$$SP \leftarrow SP - 1$$

$$M[SP] \leftarrow DR$$

The stack pointer is decremented so that it points at the address of the next word.

A memory write operation inserts the word from DR into the top of the stack.

POP

A new item is deleted with a pop operation as follows:

$$DR \leftarrow M[SP]$$

$$SP \leftarrow SP + 1$$

Polish Notation

A stack organization is very effective for evaluating arithmetic expressions. The common arithmetic expressions are written in infix notation, with each operator written between the operands.

Consider the simple arithmetic expression

$$A \bullet B + C \bullet D$$

$A + B$ Infix notation

$+AB$ Prefix or Polish notation

$AB+$ Postfix or reverse Polish notation

The reverse Polish notation is in a form suitable for stack manipulation. The expression $A \bullet B + C \bullet D$ is written in reverse Polish notation as $AB \bullet CD \bullet +$

Addressing Modes

- The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced
- The decoding step in the instruction cycle determines the operation to be performed, the addressing mode of the instruction, and the location of the operands
- Two addressing modes require no address fields – the implied mode and immediate mode
- Types of Addressing mode**
- Implied mode:** the operands are specified implicitly in the definition of the instruction – complement accumulator or zero-address instructions
- Immediate mode:** the operand is specified in the instruction
- Register mode:** the operands are in registers
- Register indirect mode:** the instruction specifies a register that contains the address of the operand

Addressing Modes

- **Auto increment or auto decrement mode:** similar to the register indirect mode
 - **Direct address mode:** the operand is located at the specified address given
 - **Indirect address mode:** the address specifies the effective address of the operand
 - **Relative address mode:** the effective address is the summation of the address field and the content of the PC
 - **Indexed addressing mode:** the effective address is the summation of an index register and the address field
 - **Base register address mode:** the effective address is the summation of a base register and the address field
- **effective address = address part of instruction + content of CPU register**

Addressing Modes

Example

$PC = 200$
$R1 = 400$
$XR = 100$
AC

Address	Memory
200	Load to AC Mode
201	Address = 500
202	Next instruction
399	450
400	700
500	800
600	900
702	325
800	300

Figure 8-7 Numerical example for addressing modes.

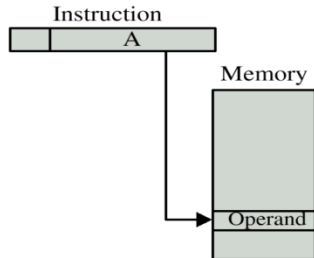
Addressing Modes

Solution

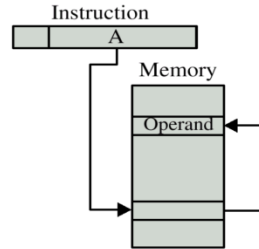
Addressing Mode	Effective Address	Content of AC
Direct address	500	800
Immediate operand	201	500
Indirect address	800	300
Relative address	702	325
Indexed address	600	900
Register		400
Register indirect	400	700
Autoincrement	400	700
Autodecrement	399	450

Addressing Modes

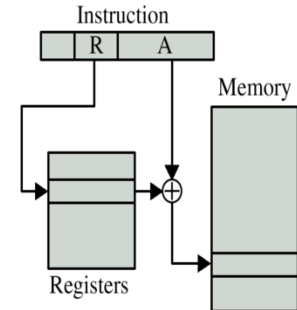
Direct Addressing



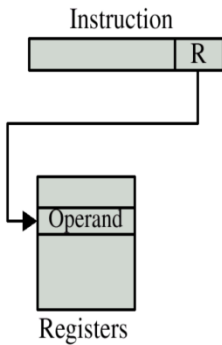
Indirect Addressing



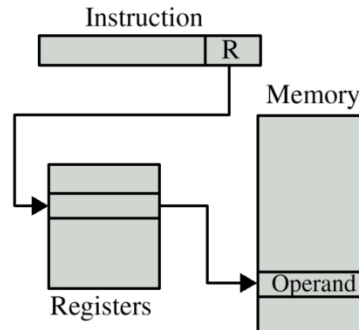
Displacement Addressing



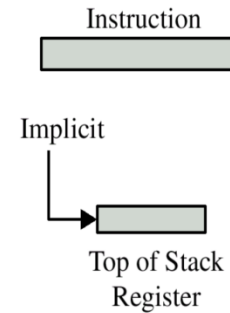
Register Addressing



Register Indirect Addressing



Stack Addressing



Addressing Modes

Summary of Addressing Modes

Mode	Algorithm	Principal Advantage	Principal Disadvantage
Immediate	Operand = A	No memory reference	Limited operand magnitude
Direct	EA = A	Simple	Limited address space
Indirect	EA = (A)	Large address space	Multiple memory references
Register	EA = R	No memory reference	Limited address space
Register indirect	EA = (R)	Large address space	Extra memory reference
Displacement	EA = A + (R)	Flexibility	Complexity
Stack	EA = top of stack	No memory reference	Limited applicability

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

Youtube/other Video Links

- https://www.youtube.com/watch?v=xBYhHC8_A6o
- https://www.youtube.com/watch?v=Xi_jRzyq42g
- https://www.youtube.com/watch?v=Xi_jRzyq42g
- <https://www.youtube.com/watch?v=KEJdEZiJxXo>
- https://www.youtube.com/watch?v=be9q_9Hcips
- https://www.youtube.com/watch?v=aDWBji_KY98
- <https://www.youtube.com/watch?v=Za7ozdjE8VI>

Daily Quiz

- Who is the father of computer ?
- Draw logic diagram of universal logic gates.
- Specify the examples of **Combinational Circuits**.
- Draw the circuit of half adder.
- Define control word .

Weekly Assignment

- 1.Explain the difference between computer architecture and computer organization.
- 2.Show the various functional units and interconnection of computers, explain with diagram.
- 3.A) Write down the various types of Registers and explain in detail.
B) Explain the PUSH and POP for type of Stack Organization
- 4.Demonstrate the Register transfer, Bus transfer and Memory transfer with suitable diagram or example.
5. Define various Addressing mode with one suitable example

- 1. A collection of lines that connects several devices is called:
a. peripheral connection wires b. bus c. Both a and b
- 2. A complete microcomputer system consist of
a. microprocessor b. memory c. peripheral equipment d. All of them
- 3. PC Program Counter is also called
a. instruction pointer b. memory pointer c. file pointer
- 4. Write types of Addressing modes-
- 5. The instructions like MOV or ADD are called as:
a. OP-Code b. Commands c. Operators d. None of the above
- 6. The instruction to be decoded is stored in:
a. PC b. MDR c. Registers d. IR
- 7. CPU does not perform the operation:
a. data transfer b. logic operation c. arithmetic operation
- Solution **1 b. 2 d. 3 a. 5 a. 6 d. 7 a**

Old Question Papers

111111 111111 1 1

■ ▼ ■ ■ ■ ■

(Following Paper ID and Roll No. to be filled in your Answer Book)

PAPER ID : 1067-NEW

Roll No.

--	--	--	--	--	--	--	--	--	--

B. Tech.

(SEM. IV) EXAMINATION, 2008-09

COMPUTER ORGANIZATION

Time : 3 Hours]

[Total Marks : 100

- Note:**
- (1) Attempt **ALL** questions.
 - (2) **All** questions carry **equal** marks.
 - (3) Be precise in your answer.
 - (4) **No Second Answer** book will be provided.

Attempt any **four** parts of the following: **5×4=20**

- (a) Represent the following conditional control statement by two register transfer statements with control function :
 if (P=1) then (R1 \leftarrow R2) else if (Q=1)
 then (R1 \leftarrow R3)
- (b) Register A holds the 8-bit binary 11011001. Determine the B operand and the logic micro-operation to be performed in order to change the value in A to :
 - (i) 01101101
 - (ii) 11111101

Old Question Papers

- (c) Design a 4-bit adder-subtractor with complete block diagram.
- (d) What do you mean by high speed adder? Discuss design of high speed adders.
- (e) Write short note on the following :
 - (i) Common Bus system
 - (ii) Bus Arbitration.
- (f) Give IEEE standard for floating point numbers.

2 Attempt any **two** parts of the following: **10×2=20**

- (a) What do you understand by hard wired control unit? Give various methods to design hardwired control unit. Describe one of the design methods for hardwired control unit with suitable diagrams.
- (b) Write short note on the following :
 - (i) Multiple-bus organization
 - (ii) Micro-programmed control unit.
- (c) (i) What do you mean by wide-branch addressing? Explain with example.
- (ii) Differentiate between a microprocessor and a microprogram? Is it possible to design a microprocessor without a microprogram? - Explain.

3 Attempt any **two** parts of the following: **10×2=20**

- (a) (i) A computer has 32-bit instructions and 12-bit addresses. If there are 250 two-address instructions, how many one-address instructions can be formulated ?

Expected Questions for University Exam

- Discuss the functional unit of computer system with their interconnection with suitable diagram.
- Define various types of Buses and structure of buses with diagrams.
- Design and analysis of various Bus Arbitration with diagram.
- Write short notes on addressing modes with suitable examples.
- Define Stack Organization with examples

In previous slides we discuss in details

- Functional units of digital system & their interconnections
- Buses, types of buses, bus architecture and bus arbitration.
- Register, bus and memory transfer.
- Processor organization, general registers organization, stack organization
- Addressing modes