Department of Information Technology
UNSIET, VBS Purvanchal University, Jaunpur

Computer Organization and Architecture KCS302

B.Tech. CSE & IT 3rd Semester

Unit –I

Introduction of Computer Organization

Ashok Kumar Yadav

Assistant Professor

Department of Information Technology

Contents

Unit -I

- 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

Course Objective

- To study the basic organization and architecture of digital computers (CPU, memory, I/O, software).
- Discussions will include digital logic and microprogramming. Such knowledge leads to better understanding and utilization of digital computers, and can be used in the design and application of computer systems or as foundation for more advanced computer-related studies.

Course Outcome

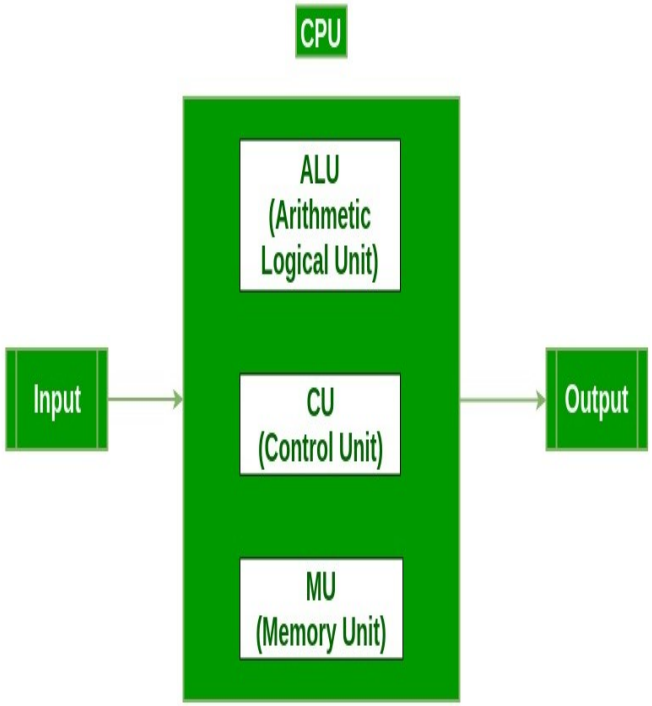
- Study of the basic structure and operation of a digital computer system.

Prerequisite and Recap

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

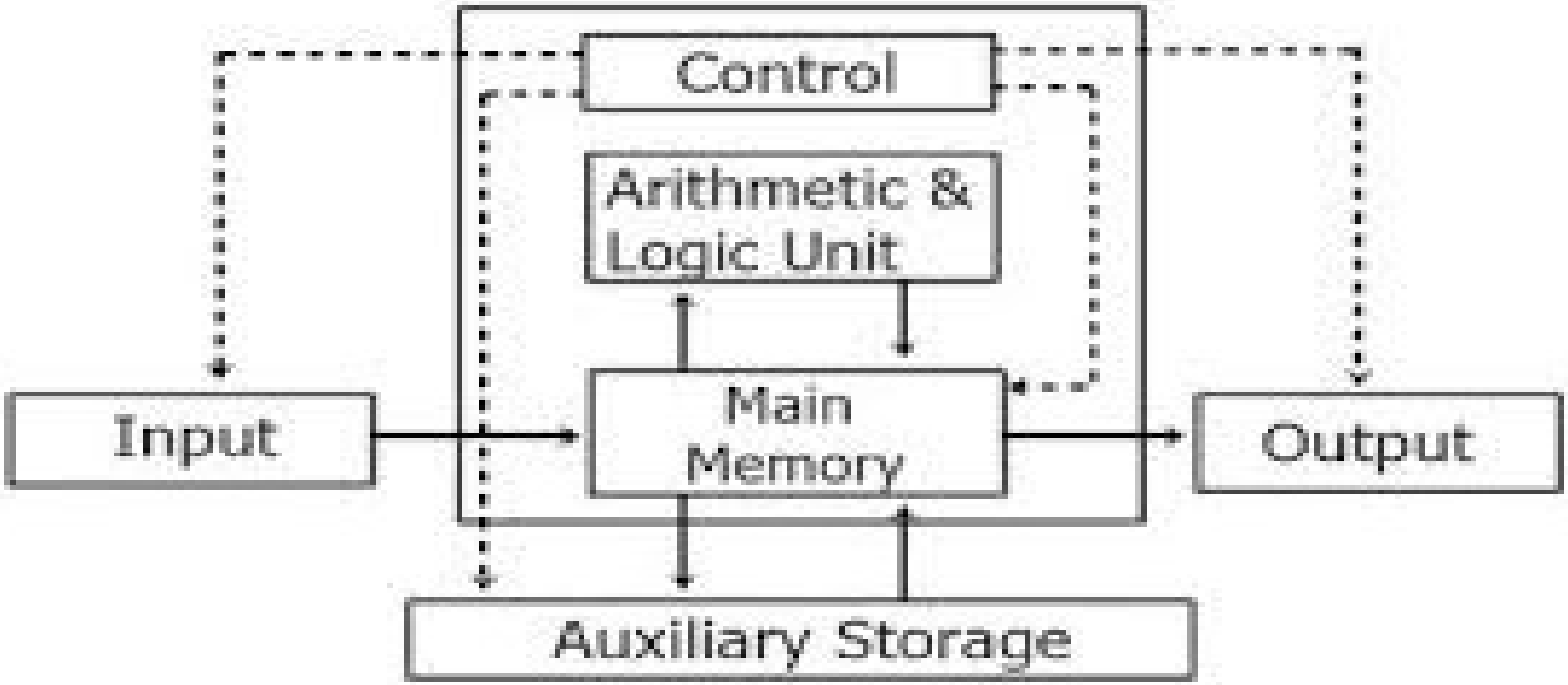
Functional Units of Computer System

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



Block Diagram

Interconnection of Functional unit

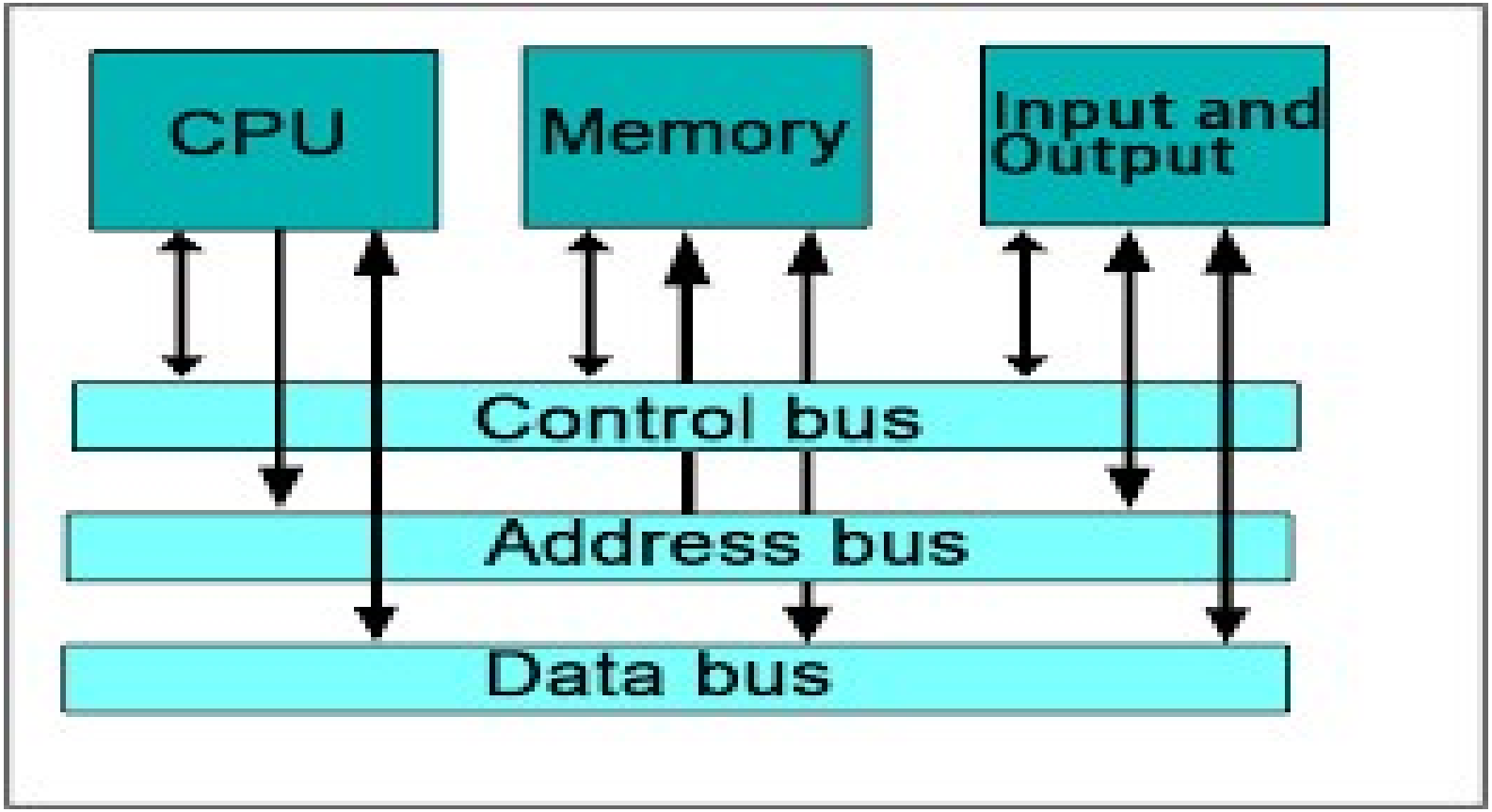


Block Diagram of Computer

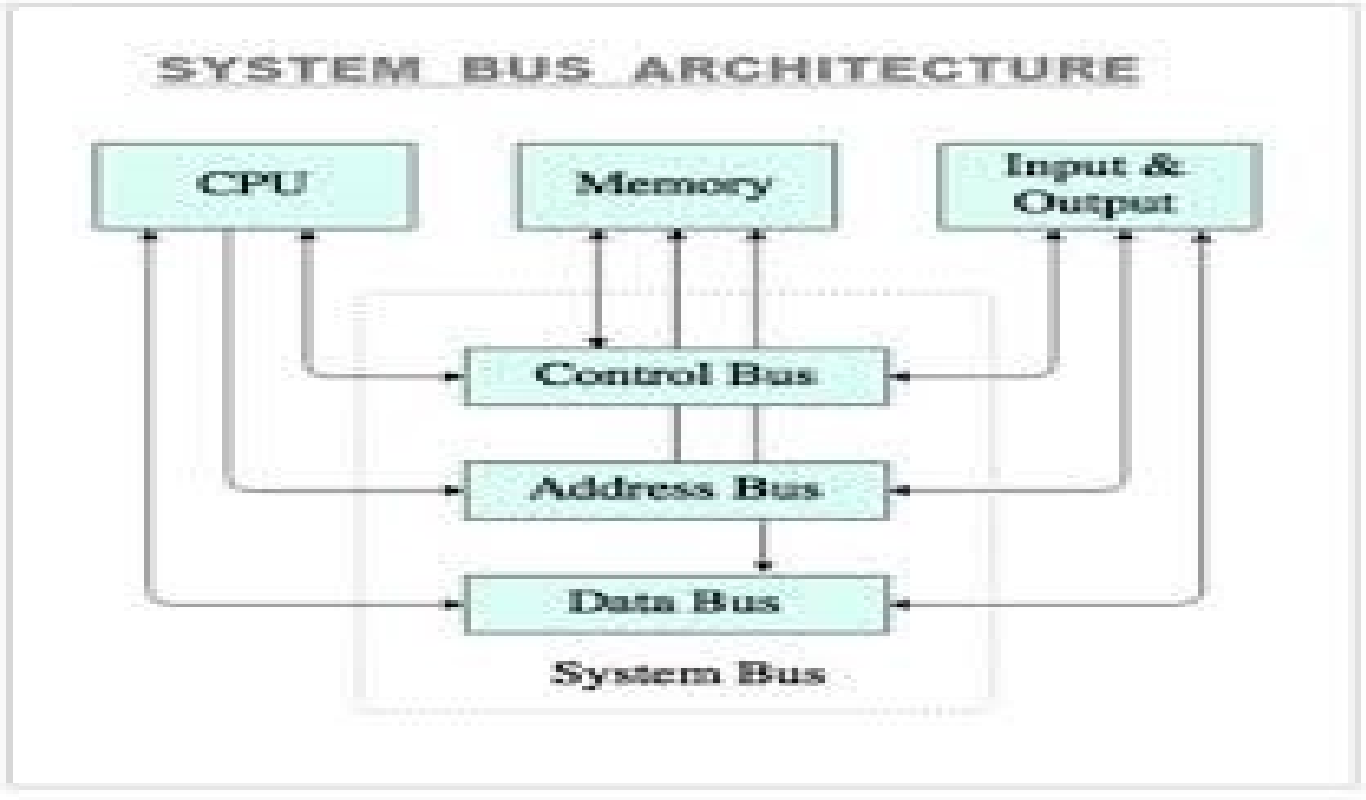
BUS and Types of BUS

- 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

Bus Architecture



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 the another bus requesting processor unit.

Bus master : The controller that has access to a bus at an instance.

Bus Arbiter decides who would become current bus master.

There are two approaches to bus arbitration:

- **Centralized bus arbitration**
- **Distributed bus arbitration**

Bus Arbitration

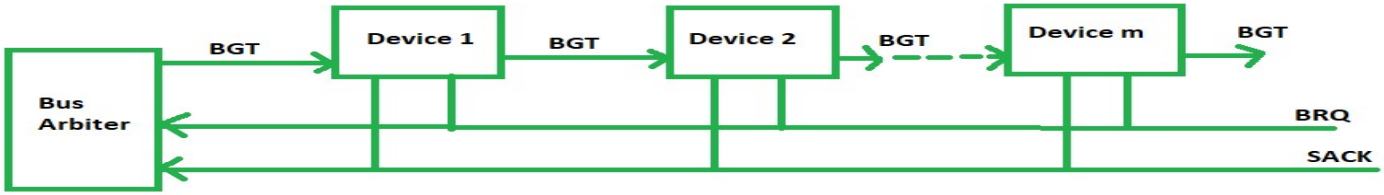
Methods of BUS Arbitration

There are three bus arbitration methods:

1. **Daisy Chaining method**
2. **Polling or Rotating Priority method**
3. **Fixed priority or Independent Request method**

Bus Arbitration

Daisy Chaining Method



Daisy chained bus arbitration

Advantages –

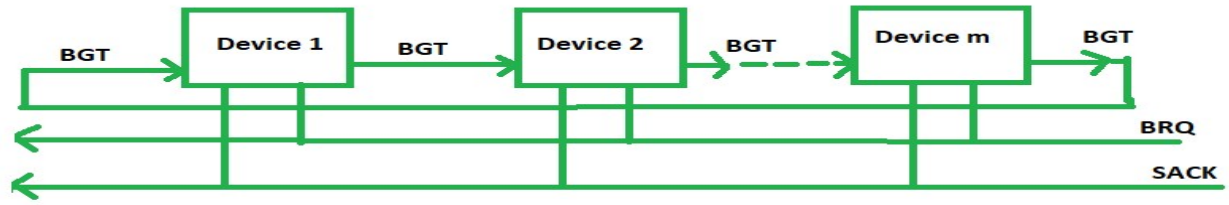
- Simplicity and Scalability.
- The user can add more devices anywhere along the chain.

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.

Bus Arbitration

Polling or Rotating Priority Method



Rotating priority bus arbitration

Advantages –

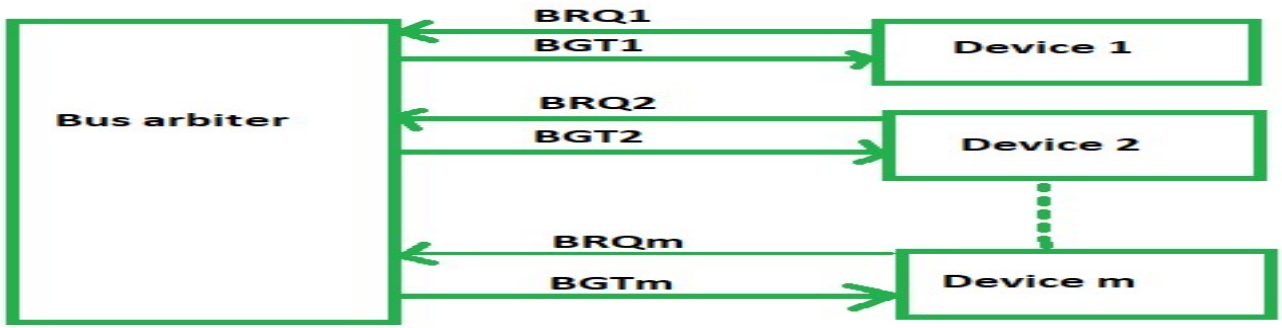
- This method does not favor any particular device and processor.
- The method is also quite simple.
- If one device fails then entire system will not stop working.

Disadvantages –

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

Bus Arbitration

Fixed Priority or 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.

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 those are used for various purpose.

Register (MDR), **Index register and 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

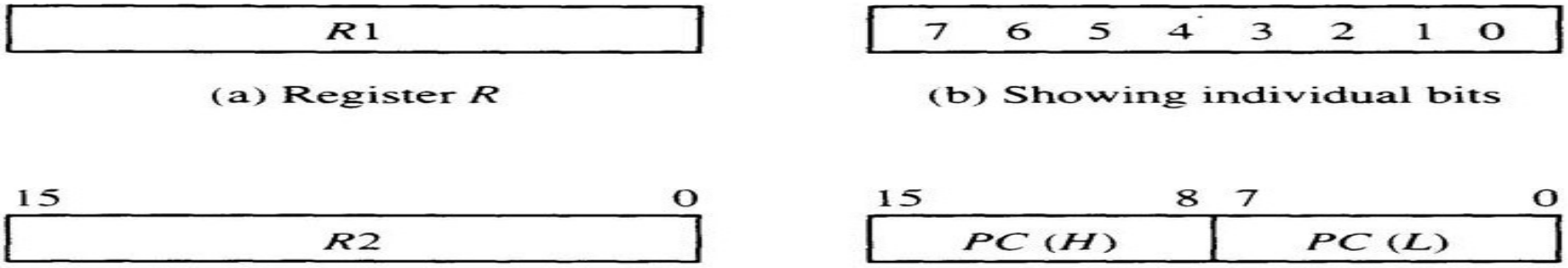
Register, Bus and Memory Transfer

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.



Register, Bus and Memory Transfer

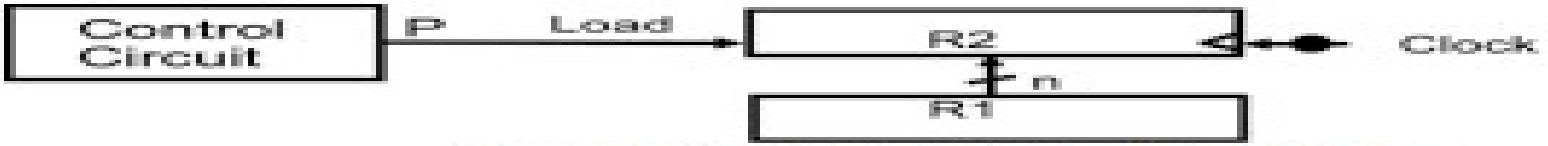
Register Transfer with block diagram

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

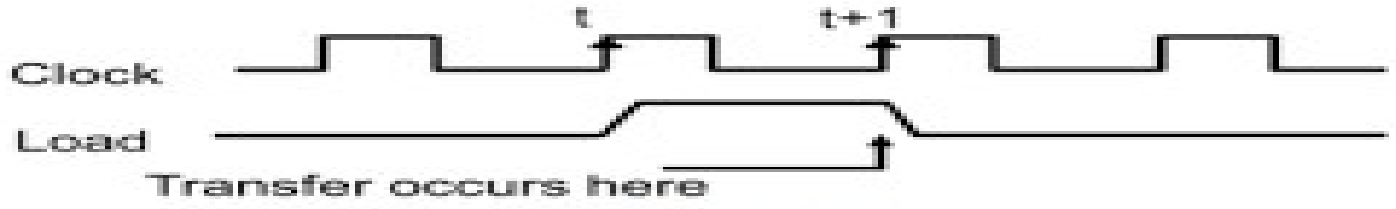
or, *If* (P = 1) *then* (R2 ⟵ R1)

P: R2 ⟵ R1,

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



Transfer from R1 to R2 when P = 1



Timing diagram

Register, Bus and Memory Transfer

Basic Symbols for Register Transfers

Symbol	Description	Examples
Letters (and numerals)	Denotes a register	*MAR*, *R2*
Parentheses ( )	Denotes a part of a register	*R2*(0–7), *R2*(*L*)
Arrow ←	Denotes transfer of information	*R2* ← *R1*
Comma ,	Separates two microoperations	*R2* ← *R1*, *R1* ← *R2*

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.

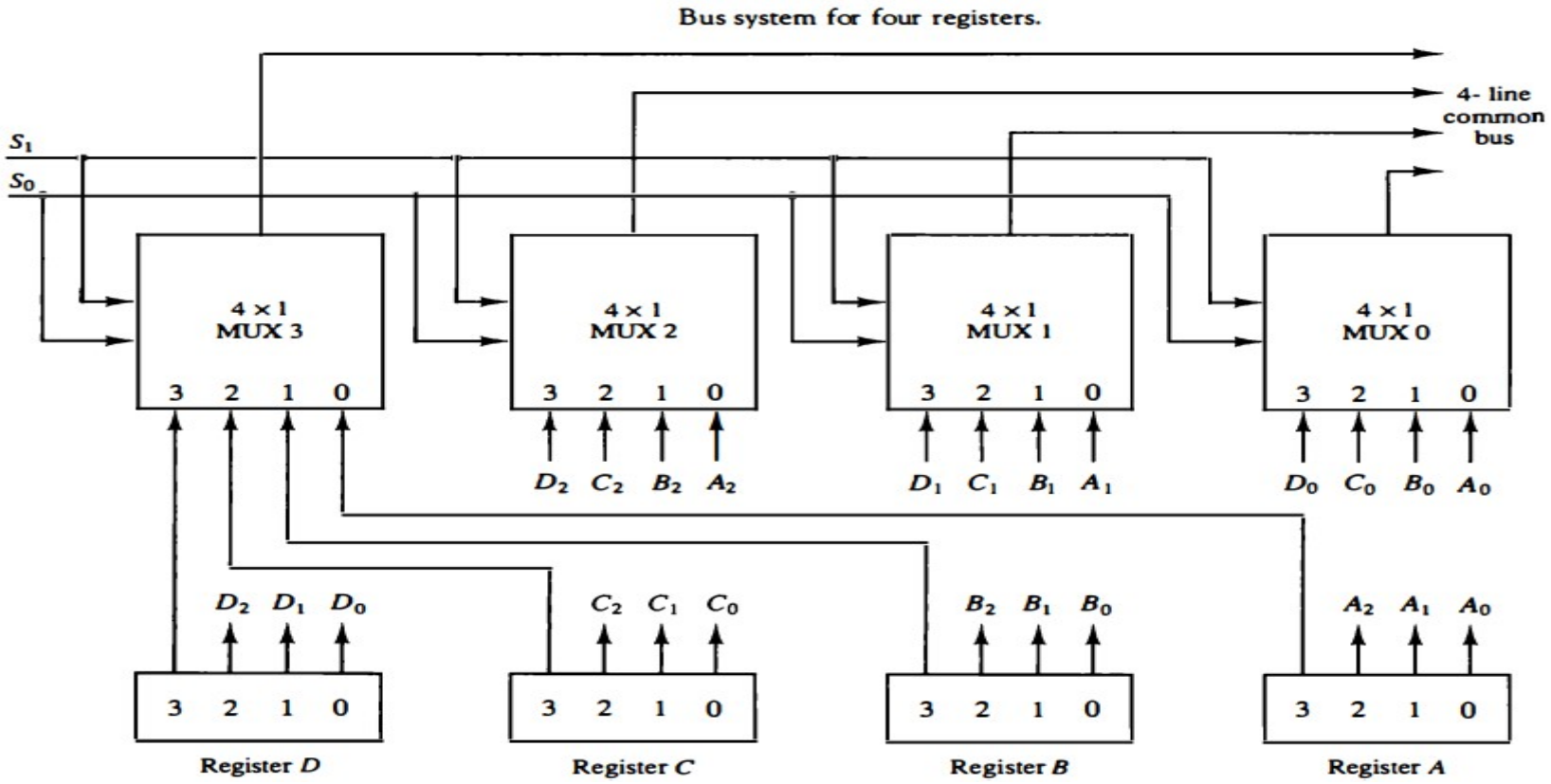
Register, Bus and Memory Transfer

- This requires *n* multiplexers – one for each bit
- The size of each multiplexer must be *k* x 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

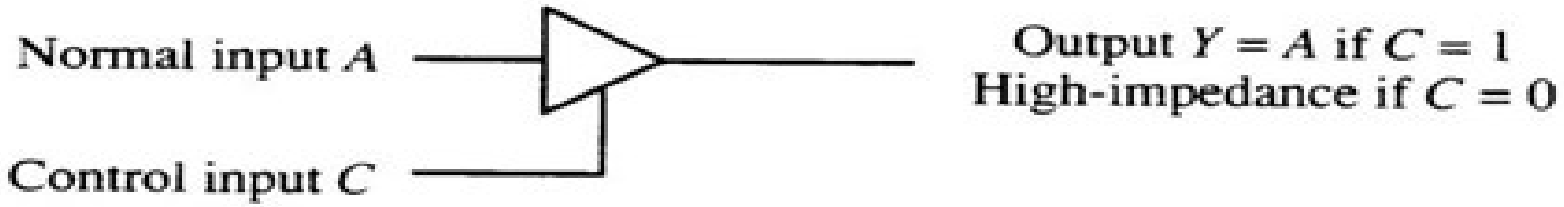


Register, Bus and Memory Transfer

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.



Register, Bus and Memory Transfer

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.

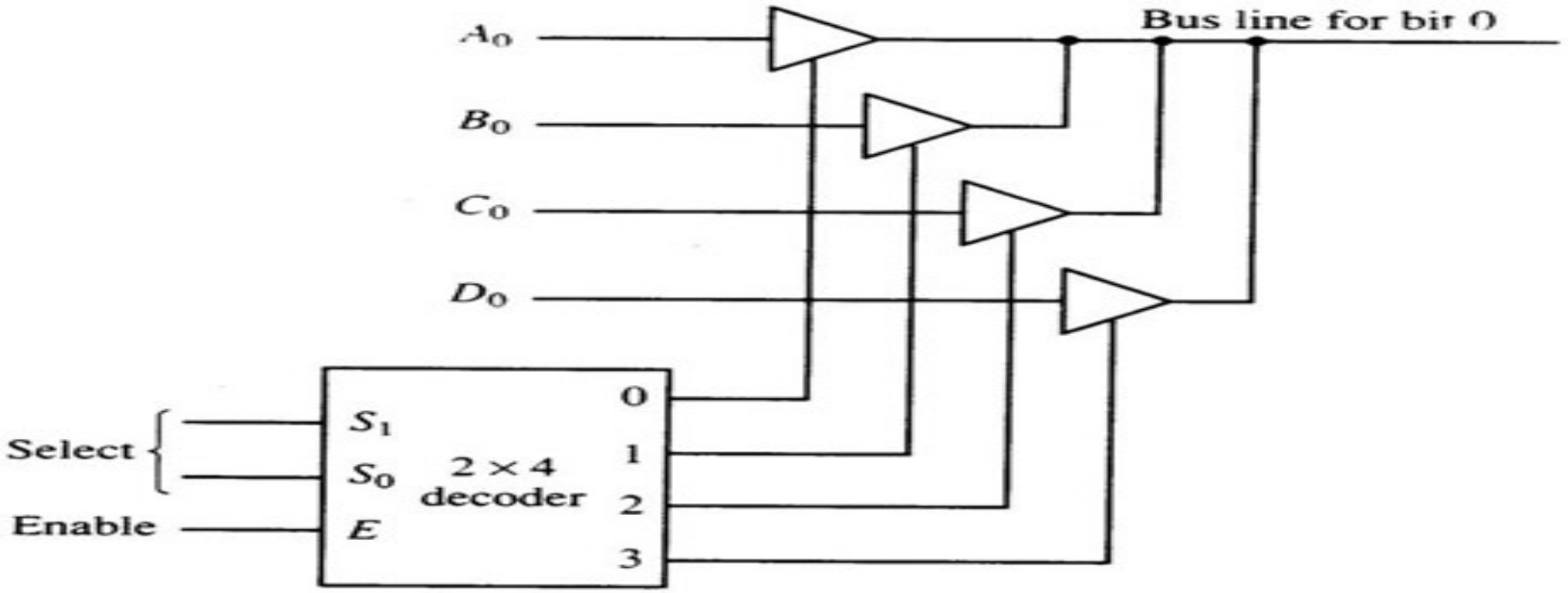
Register, Bus and Memory Transfer

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
- 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 ¬ M[AR]
- The write operation can be stated as: Write: M[AR] ¬ R1

Register, Bus and Memory Transfer

Bus Transfer using Three state buffer



Bus line with three state-buffers.

Register, Bus and Memory Transfer

Bus Transfer using Three state buffer

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 ← M [AR]

Register, Bus and Memory Transfer

Bus Transfer using Three state buffer

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

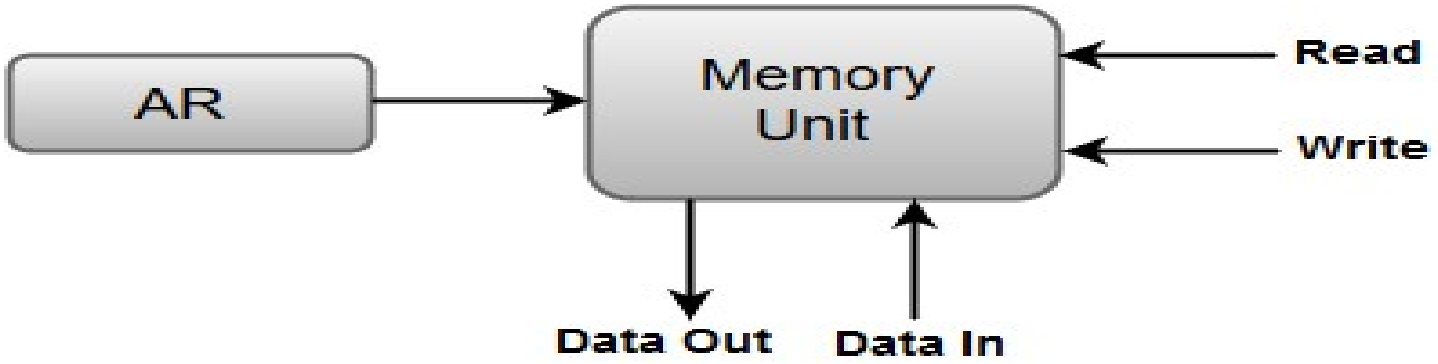
And the corresponding write operation can be stated as:

- 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] ← R1

Register, Bus and Memory Transfer

Memory Transfer Block diagram



Above Diagram showing connections to memory unit.

Write: M[AR] ← DR

Read: DR ← M[AR]

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 <--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 R 1 <---R2 + R3 .

ADD R1, R2, would denote the operation R 1 <---R1 + R2. Only register addresses for R 1 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

POP

General Register 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:

MUL R1, R2, R3

- This is an instruction of an arithmetic multiplication written in assembly language. It uses three address fields R1, R2 and R3.
- The meaning of this instruction is:

R1 <-- R2 * R3

General Register Organization

- 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 <-- 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.

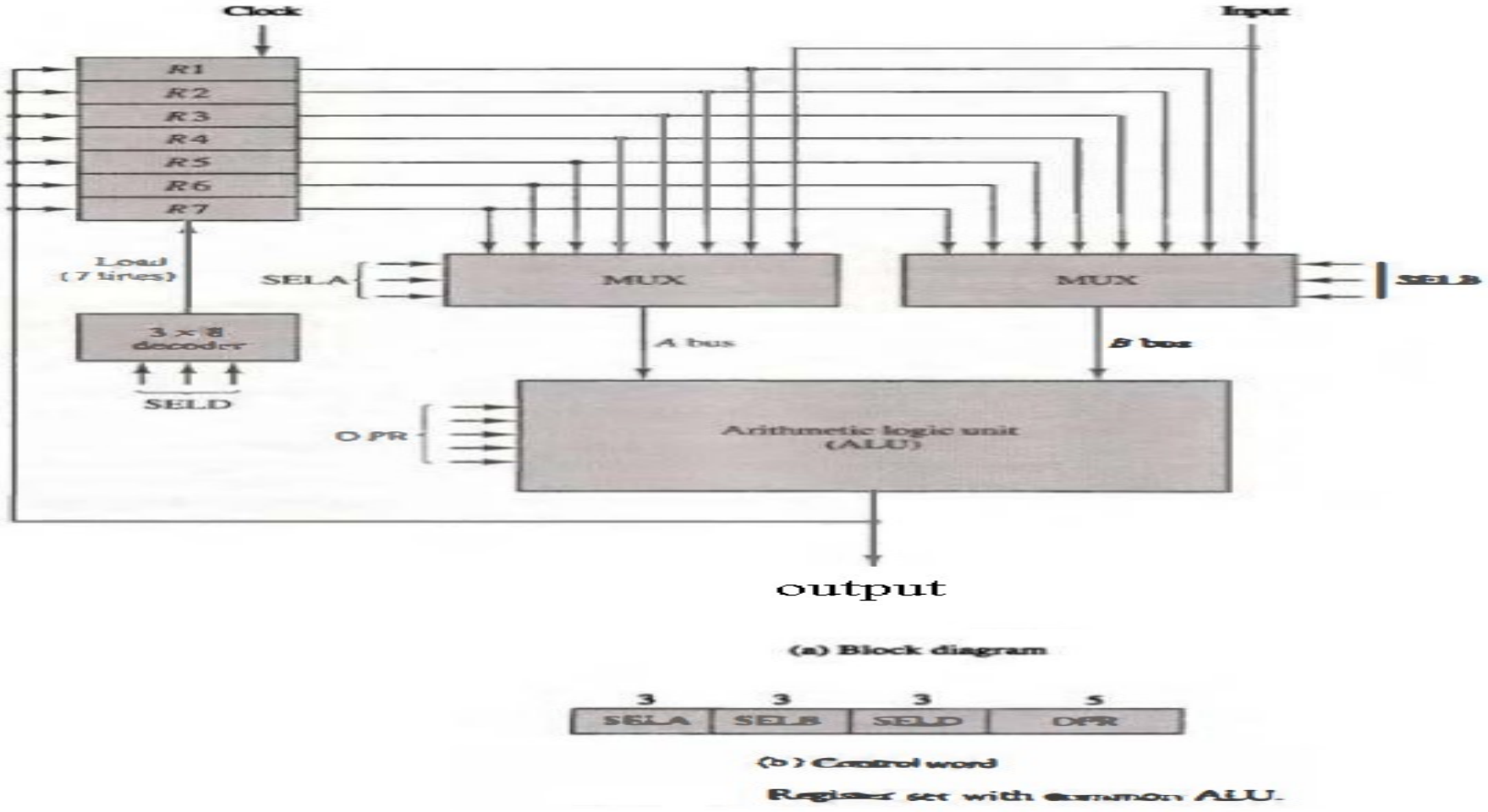
•Less memory space is used to store the program since the instructions are written in compact way.

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



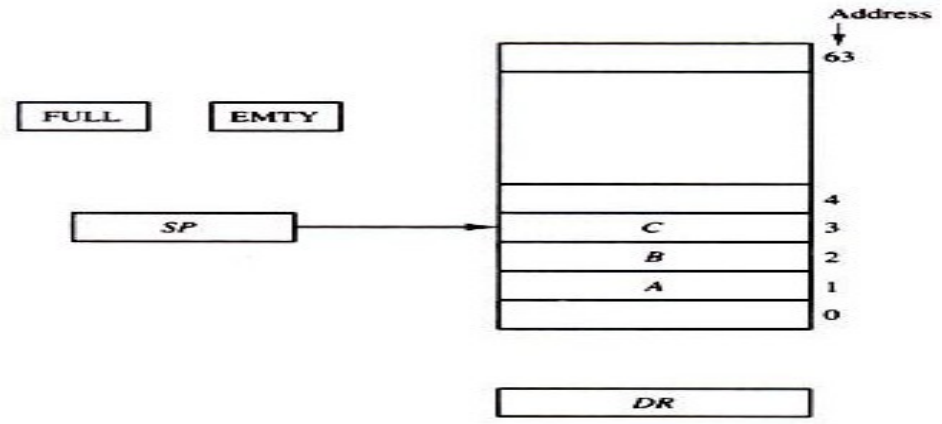
General Register Organization

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 <--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 o f R 3 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 .

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:



Stack Organization

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 ← SP + 1 Increment stack pointer

M [SP] ← DR WRITE ITEM ON TOP OF THE STACK

IF (SP = 0) then (FULL ← 1) Check is stack is full

EMTY ← 0 Mark the stack not empty

Stack Organization

POP:

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

DR ← M [SP] Read item on top of the stack

SP ← SP - 1 Decrement stack pointer IF (SP = 0) then (EMTY ← 1)

Check if stack is

empty FULL ← 0 Mark the stack not full

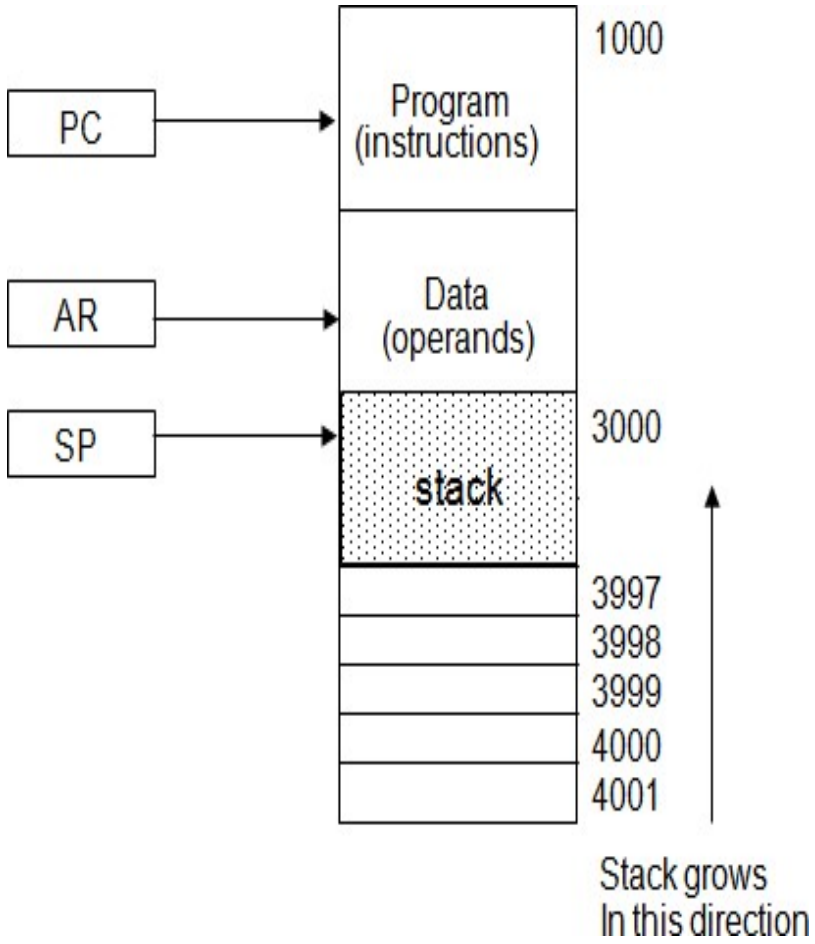
The top item is read from the stack into DR. The stack pointer is then decremented. If its value reaches zero, the stack is empty, so EMTY is set to 1.

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.



Stack Organization

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.

Stack Organization

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:

$$DR \leftarrow M[SP]$$

$$SP \leftarrow SP + 1$$

Stack Organization

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•B + C•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•B + C•D is written in reverse Polish notation as AB•CD•+

References

[1] Mano," Computer System Architecture", PHI
[2] P Pal chaudhry, ' Computer Organization & Design', PHI
[3] Patterson, Computer Organisation and Design, Elsevier Pub. 2009
[4] William Stalling, " Computer Organization", PHI