UNIT III: Register Transfer Logic

FACULTY

Dr. K. ARTHI MCA, M.Phil., Ph.D.,
Assistant Professor,
Postgraduate Department of Computer Applications,
Government Arts College (Autonomous),
Coimbatore-641018.

Register Transfer Language

- ❖ A specific notation used to specify the digital system is called register transfer language.
- ❖ For any function of the computer, the register transfer language can be used to describe the (sequence of) micro-operations
- ❖ 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

Micro-operations

❖ Simple digital systems are frequently characterized in terms of

- the registers they contain,and
- the operations that they perfonn.

❖ The operations on the data in registers are called micro- operations.

❖ The functions built into registers are examples of micro- operations

- Shift
- Load
- Clear
- Increment ... etc

Micro-operations

- Computer system micro-operations are of four types:
  - Register transfer micro-operations
  - Arithmetic micro-operations
  - Logic micro-operation s
  - Shift micro-operations

Register Transfer

- ❖ Copying the contents of one register to another is a register transfer
- ❖ A register transfer is indicated as R2RI
  - ➢ In this case the contents of register R2 are move into register RI
  - ➢ A simultaneous transfer of all bits from the source RI to the destination register R2, during one clock pulse
  - ➢ Note that this is a destructive; i.e. the contents of RI are not altered by copying (loading) them to R2

Arithmetic Micro-operations

- The basic arithmetic micro-operations are
  - Addition
  - Subtraction
  - Increment Decrement
- The additional arithmetic micro-operations are
  - Add with carry
  - Subtract with borrow
  - Transfer/Load etc. ...

Summary of Typical Arithmetic Micro-Operations

Arithmetic micro-operations

R3 <- R1+R2	Contents of R1 Plus R2 Transferred to R3
R3<- R1 –R2	Contents of R1 minus R2 transferred to R3
R2<-R2’	Complement the contents of R2
R2<-R2+1	2’s complement the contents of R2(negate)
R3<- R1+R2’+1	Subtraction
R1<- R1 + 1	Increment
R1<- R1 – 1	Decrement

Logical Micro-operations

❖ Specify binary operations on the strings of bits in registers
- Logic micro – operations are bit-wise operations, i.e., they work on individual bits of data

❖ There are in principle, 16 different logic functions that can be defined over two binary input variables

A	B	F0	F1	F2 ……	F13	F14	F15
0	0	0	0	0 …….	1	1	1
0	1	0	0	0 …….	1	1	1
1	0	0	0	1 …….	0	1	1
1	1	0	1	0 …….	1	0	1

However, most systems only implement four of these
- AND (^), OR (), XOR, Complement / NOT

The Others can be created from combination of these

Shift Micro-operations

- ❖ Shift Micro Operation are used for serial transfer of data
- ❖ They are also used in conjuction with arithmetic, logic and other data processing operation
- ❖ The contents of register can be shifted to the left or right
- ❖ At the same time that the bits are shifted, the first flip- flop receives its binary information from the serial input.

❖ Shift Left Operation :

-> Serial input transfers a bit into the rightmost Operation.

❖ Shift Right Operation :

-> Serial input transfers a bit into the leftmost Operation.

❖ Type of Shift Micro Operation:

The information transferred through the serial input determines the type of shift

❖ 3 Types of Shifts:

i) Logical Shift
ii) Circular Shift
iii) Arithmetic shift

❖ Logical Shift:

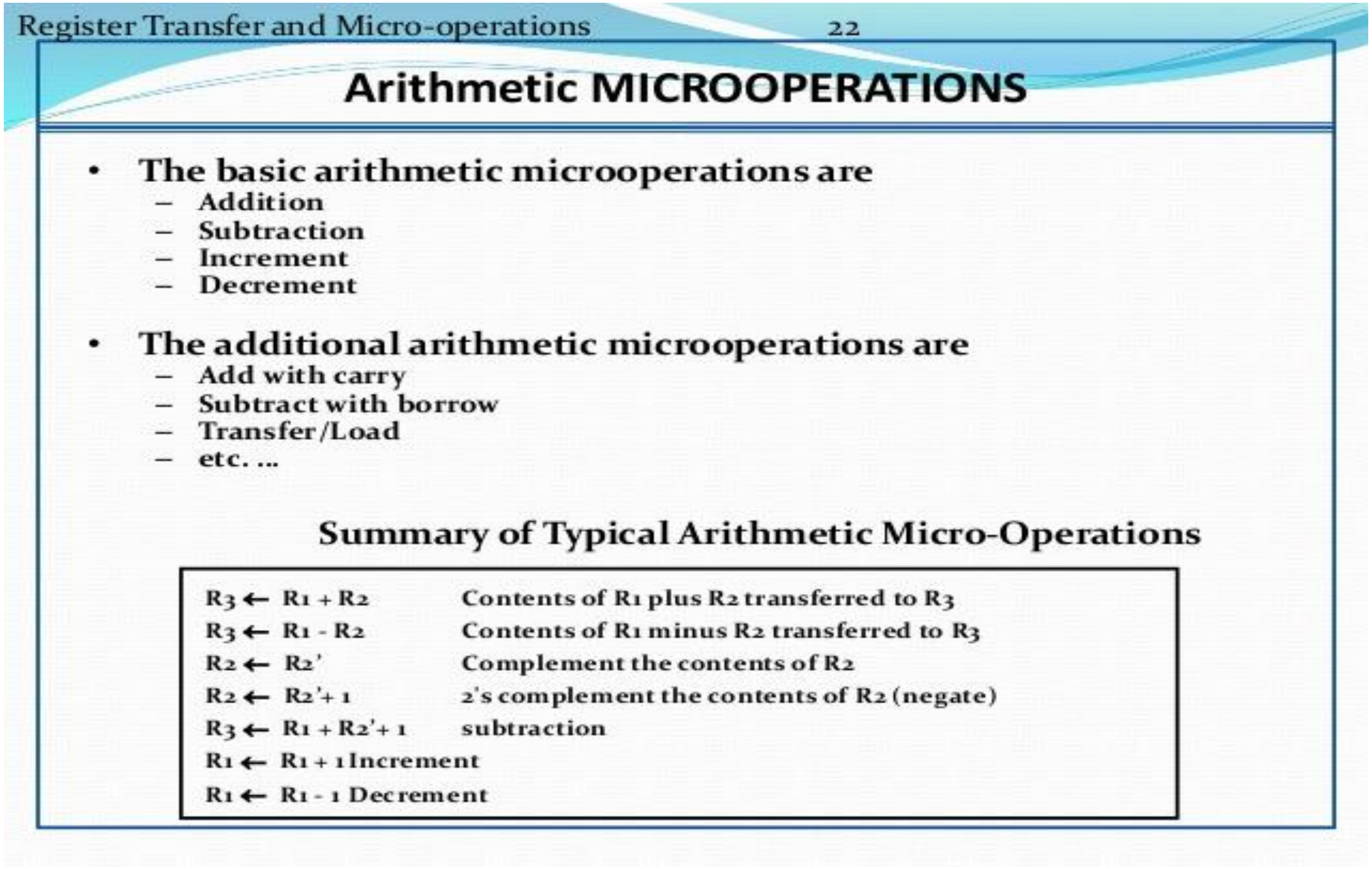
  -> Transfer 0 through the serial input

❖ Circular Shift:

  -> Circulates the bits of the register around the two ends without loss of information.

❖ Arithmetic Shift:

  -> Shifts a signed binary number to the left or right.



Arithmetic MICROOPERATIONS

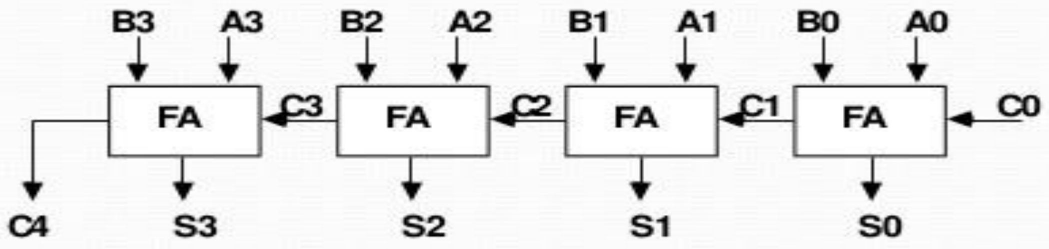
- The basic arithmetic microoperations are
  - Addition
  - Subtraction
  - Increment
  - Decrement
- The additional arithmetic microoperations are
  - Add with carry
  - Subtract with borrow
  - Transfer/Load
  - etc. …

Summary of Typical Arithmetic Micro-Operations

$R3 \leftarrow R1 + R2$	Contents of R1 plus R2 transferred to R3
$R3 \leftarrow R1 - R2$	Contents of R1 minus R2 transferred to R3
$R2 \leftarrow R2'$	Complement the contents of R2
$R2 \leftarrow R2' + 1$	2's complement the contents of R2 (negate)
$R3 \leftarrow R1 + R2' + 1$	subtraction
$R1 \leftarrow R1 + 1$ Increment	
$R1 \leftarrow R1 - 1$ Decrement	

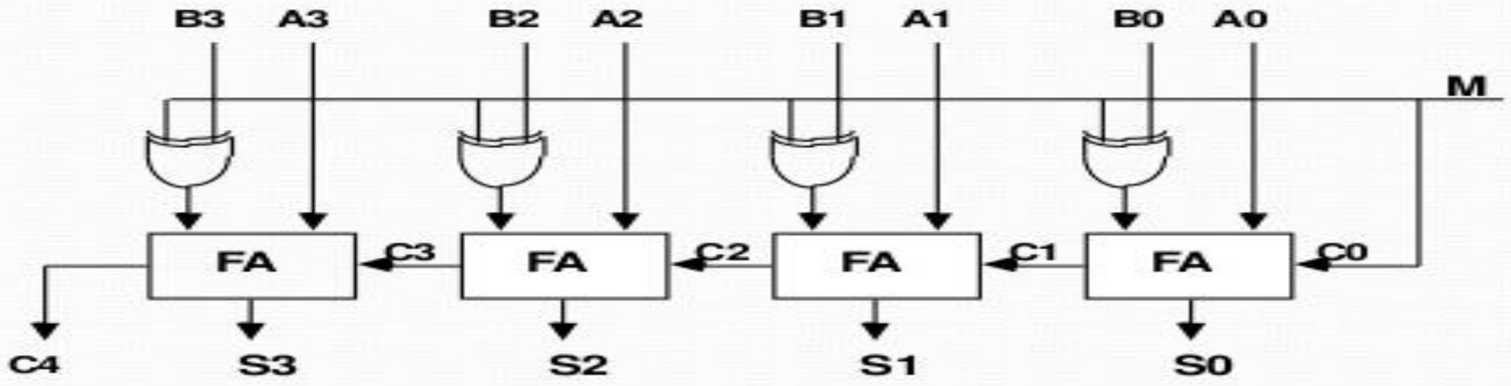
Binary Adder

Binary Adder



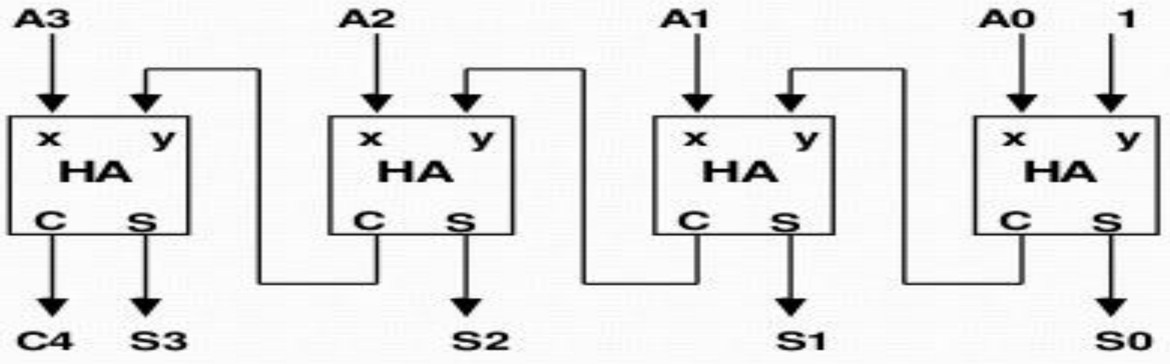
Binary Adder-Subtractor

Binary Adder-Subtractor

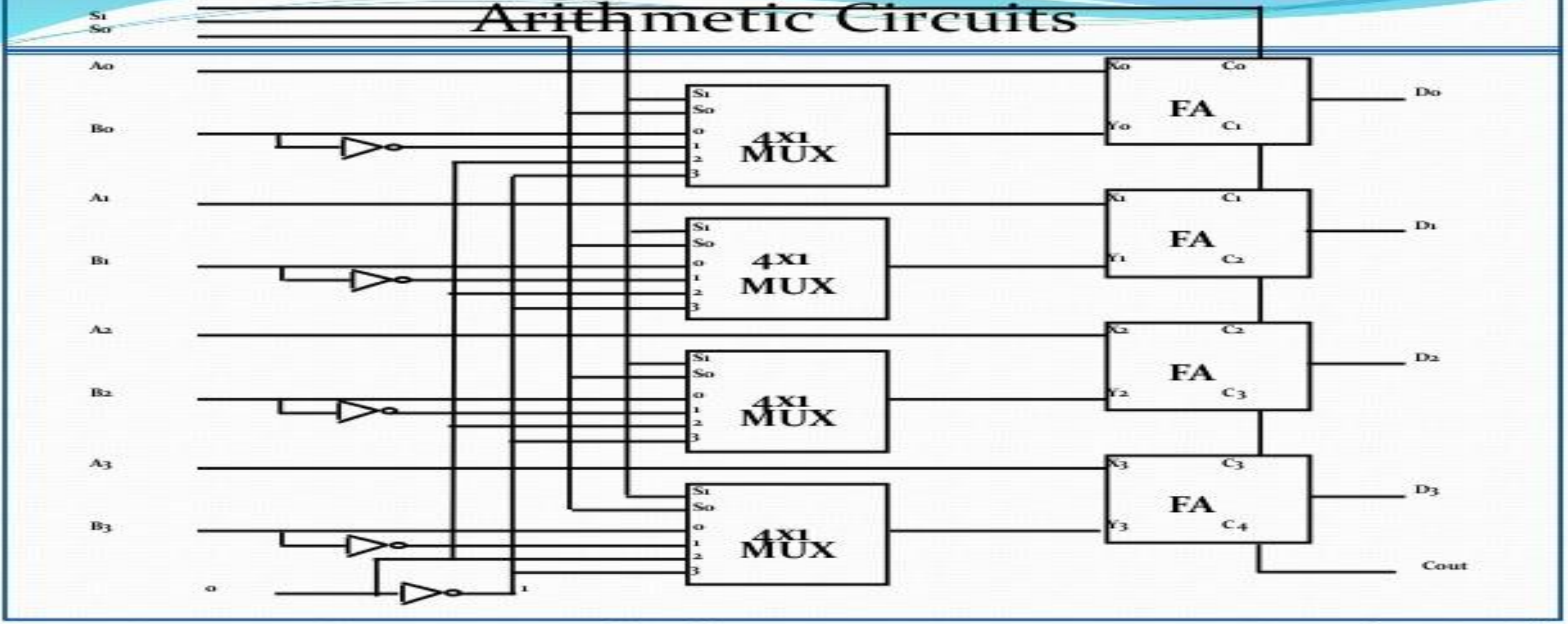


Binary Incrementer

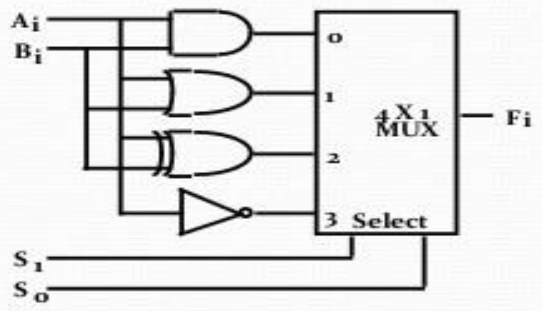
Binary Incrementer



Arithmetic Circuits



Hardware Implementation

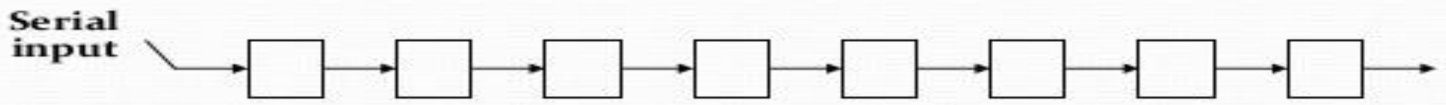


Function table

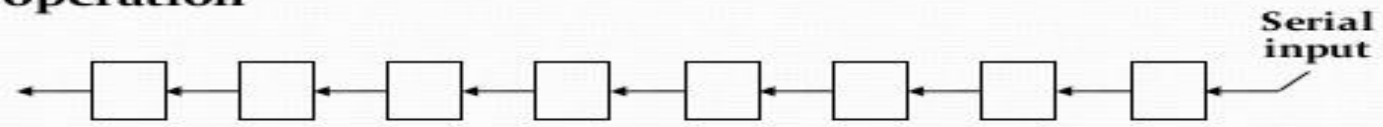
S_1 S_0	Output	μ-operation
0 0	$F = A \wedge B$	AND
0 1	$F = A \vee B$	OR
1 0	$F = A \oplus B$	XOR
1 1	F = A’	Complement

Shift Microoperations

- There are three types of shifts
  - *Logical shift*
  - *Circular shift*
  - *Arithmetic shift*
- What differentiates them is the information that goes into the serial input
  - A right shift operation

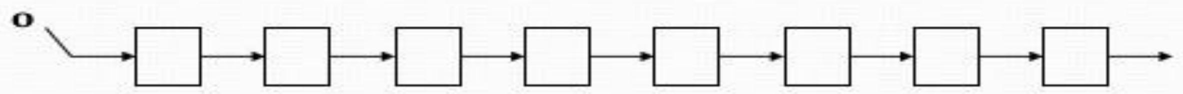


  - A left shift operation

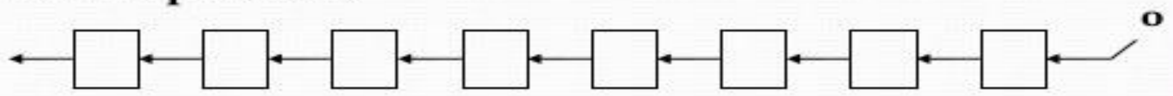


Logical Shift

- **In a logical shift the serial input to the shift is a 0.**
- **A right logical shift operation:**



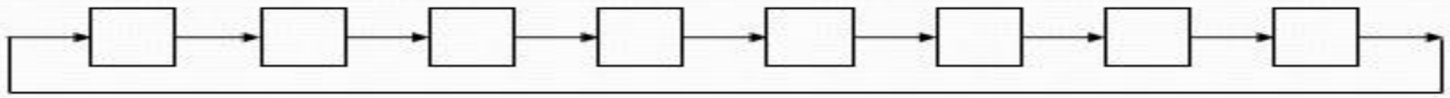
- **A left logical shift operation:**



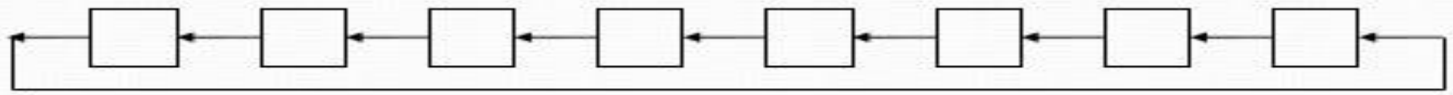
- **In a Register Transfer Language, the following notation is used**
  - *shl* for a logical shift left
  - *shr* for a logical shift right
  - Examples:
    - $R2 \leftarrow shr\ R2$
    - $R3 \leftarrow shl\ R3$

Circular Shift

- **In a circular shift the serial input is the bit that is shifted out of the other end of the register.**
- **A right circular shift operation:**



- **A left circular shift operation:**

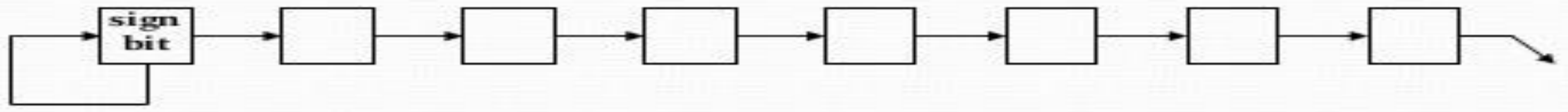


- **In a RTL, the following notation is used**
  - ***cil*** **for a circular shift left**
  - ***cir*** **for a circular shift right**
  - **Examples:**
    - **R2 ← *cir* R2**
    - **R3 ← *cil* R3**

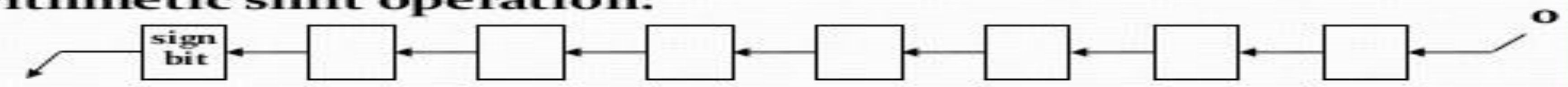
Arithmetic Shift

- **An arithmetic shift is meant for signed binary numbers (integer)**
- **An arithmetic left shift multiplies a signed number by two**
- **An arithmetic right shift divides a signed number by two**
- **The main distinction of an arithmetic shift is that it must keep the sign of the number the same as it performs the multiplication or division**

- **A right arithmetic shift operation:**

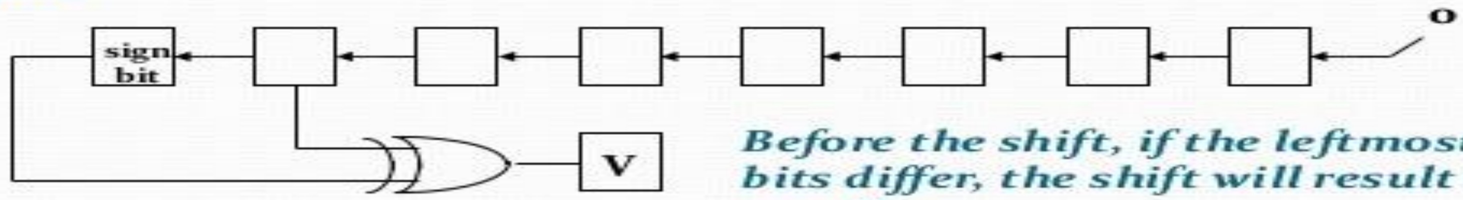


- **A left arithmetic shift operation:**



Arithmetic Shift

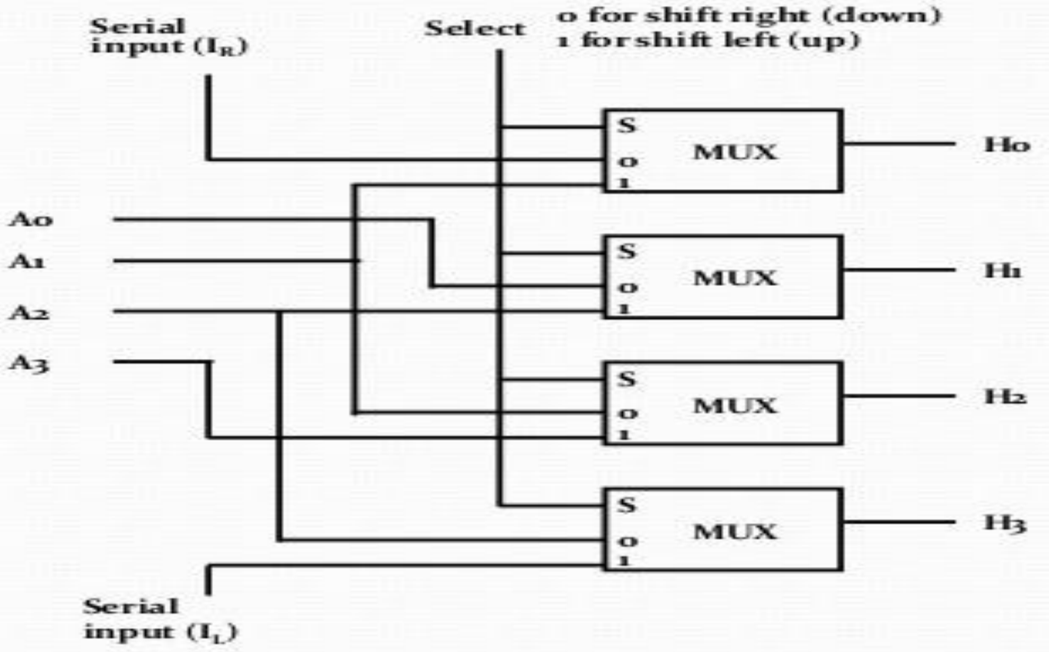
- An left arithmetic shift operation must be checked for the overflow



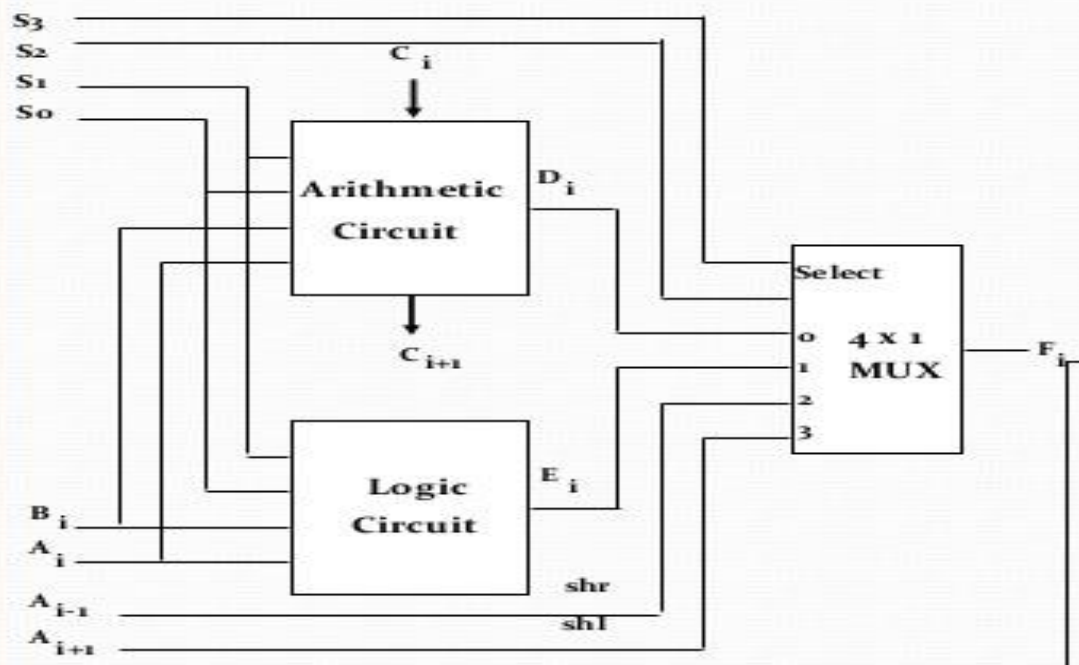
Before the shift, if the leftmost two bits differ, the shift will result in an overflow

- In a RTL, the following notation is used
  - *ashl* for an arithmetic shift left
  - *ashr* for an arithmetic shift right
  - Examples:
    - » R2 ← *ashr* R2
    - » R3 ← *ashl* R3

Hardware Implementation of Shift Microoperation



Arithmetic Logic and Shift Unit



S_3	S_2	S_1	S_0	Cin	Operation
0	0	0	0	0	F = A
0	0	0	0	1	F = A + 1
0	0	0	1	0	F = A + B
0	0	0	1	1	F = A + B + 1
0	0	1	0	0	F = A + B'
0	0	1	0	1	F = A + B'+1
0	0	1	1	0	F = A - 1
0	0	1	1	1	F = A
0	1	0	0	X	$F = A \wedge B$
0	1	0	1	X	$F = A \vee B$
0	1	1	0	X	$F = A \oplus B$
0	1	1	1	X	F = A'
1	0	X	X	X	F = shr A
1	1	X	X	X	F = shl A

MICRO COMPUTER

History of Micro computer :

Early microcomputers In late 1972, a French team headed by François Gernelle within a small company, Réalisations & Etudes Electroniqes (R2E), developed and patented a computer based on a microprocessor – the Intel 8008 8-bit microprocessor.

- Intel’s first microcomputer add appeared in November 1971

“Announcing a new era in integrated electronics.”

Definition :

Micro Computer is a small computer. Your personal computers are equivalent to the microcomputer. Mainframe and Mini Computer is ancestor of microcomputer.

A microcomputer is the **smallest general purpose processing system**.

Micro Computer:

Microcomputers are designed to serve only one user at a time, although they can often be modified with software or hardware to concurrently serve more than one user.

The term *microcomputer* came into popular use after the introduction of the minicomputer.

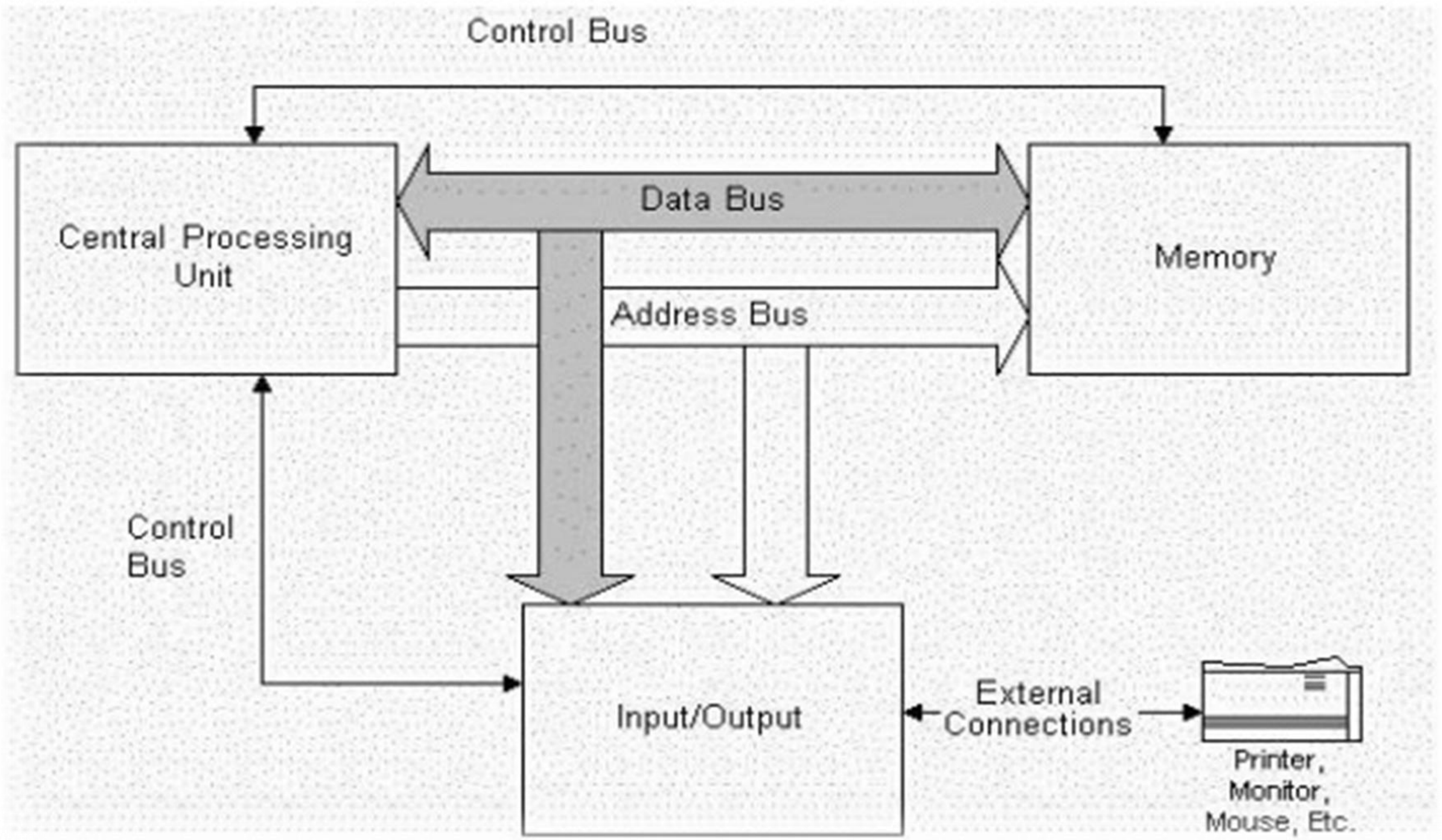
Three microcomputer systems frequently associated with the first wave of commercially successful 8-bit home computers.

- ❑The Commodore PET 2001, the Apple II, and the TRS-80 Model 1.

Some many types of micro computers:

- Desktop computer
- Notebook or Laptop
- Tablet
- Smartphone
- Personal digital assistant
- server

Architecture of microcomputer



Instructions:

A binary code used for specifying micro-operations for the computer.

Instruction Code:

- Instructions are encoded as binary *instruction codes*.
- Each instruction code contains of an *operation code*, or *opcode*, which designates the overall purpose of the instruction.
- The number of bits allocated for the opcode determined how many different instructions the architecture supports.

Different types of Instruction formats

- Zero address instruction format
- One address instruction format
- Two address instruction format Two address instruction format
- Three address instruction format

Instruction Format

An instruction consists of bits and these bits are grouped up to make fields.

Some fields in instruction format are as follows

1. Opcode which tells about the operation to be performed.
2. Address field designating a memory address or a processor register.
3. Mode field specifying the way the operand or effective address is determined.

Addressing Modes

- The term addressing modes refers to the way in which the operand of an instruction is specified.
- The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually executed.

Addressing modes are divided into two categories:

* Addressing modes for data

* Addressing modes for branch

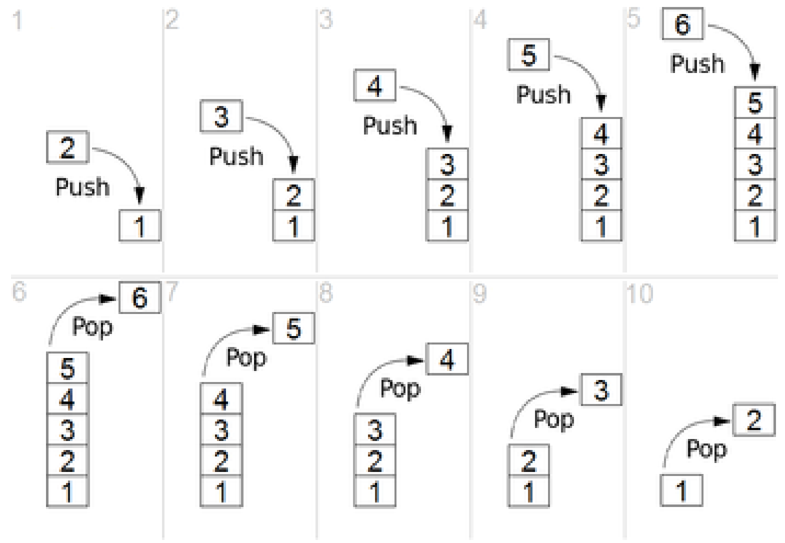
- The 8086 memory addressing modes provide flexible access to memory, allowing you to easily access variables, arrays, records, pointers, and other complex data types.

- The key to good assembly language programming is the proper use of memory addressing modes.

Stack:

The stack is a **list of data words**. It uses Last In First Out (LIFO) access method which is the most popular access method in most of the CPU.

A register is used to store the address of the topmost element of the stack which is known as Stack pointer (SP). In this organisation, ALU operations are performed on stack data.



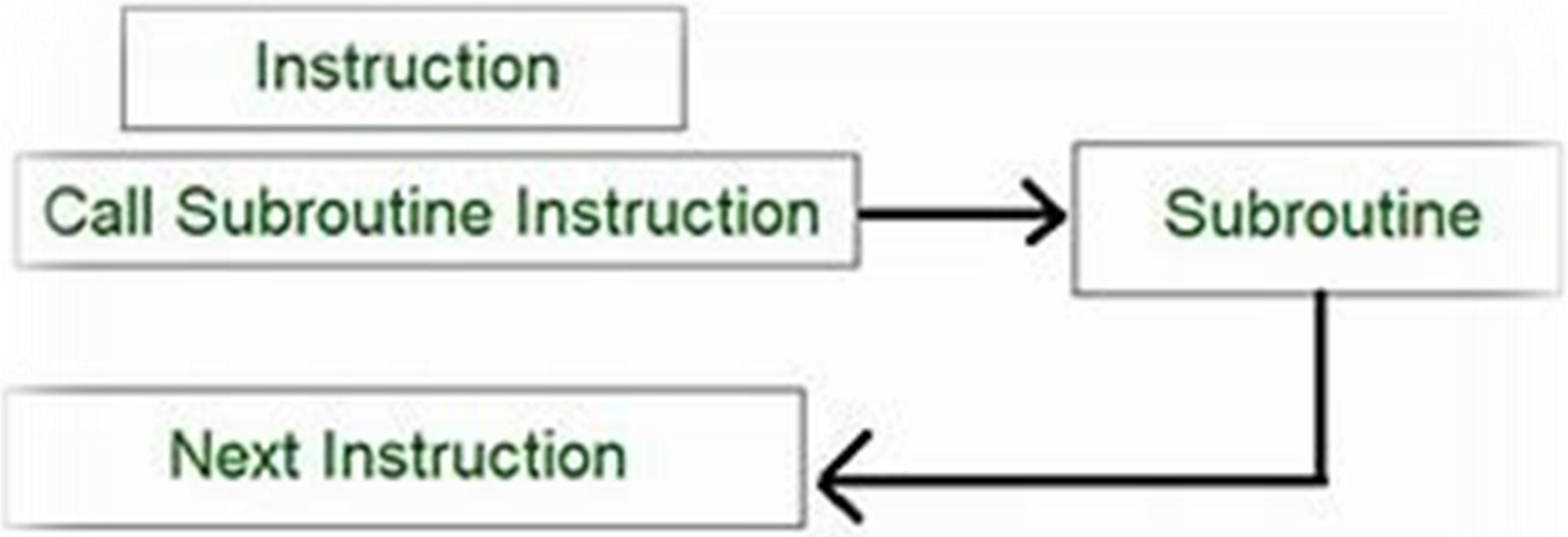
Subroutine:

In computer programming, a **subroutine** is a sequence of program instructions that performs a specific task, packaged as a unit. This unit can then be used in programs wherever that particular task should be performed.

In different programming languages, a subroutine may be called a **routine**, **subprogram**, **function**, **method**, or **procedure**. Technically, these terms all have different definitions.

The generic, umbrella term **callable unit** is sometimes used.

Subroutine function



Interrupt

Interrupt is the mechanism by which modules like I/O or memory may interrupt the normal processing by CPU.

Program interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request.

The concept of program interrupt is used to handle a variety of problems that arise out of normal program sequence.

Types of Interrupts :

1. External interrupts
2. Internal interrupts
3. Software interrupts

External interrupts:

An external interrupt is a computer system interrupt that happens as a result of outside interference, whether that’s from the user, from peripherals, from other hardware devices or through a network.

Internal interrupts

An internal interrupt is a specific type of interrupt that is caused by instructions embedded in the execution instructions of a program or process.

Typically, internal interrupts resist changes by users, and happen "naturally" or "automatically" as a processor works through program instructions, rather than being caused by external events or network connections.

Software interrupts

A software interrupt is a type of interrupt that is caused either by a special instruction in the instruction set or by an exceptional condition in the processor itself.

A software interrupt is invoked by software, unlike a hardware interrupt, and is considered one of the ways to communicate with the kernel or to invoke system calls, especially during error or exception handling.

I/O BUS AND INTERFACE MODULES :

- The data bus, address bus and control bus that arise out of the processor and are intended to communicate with I/O devices are called I/O bus.
- The I/O bus is connected to all peripheral interfaces. To communicate with a particular device, the processor places a device address on the address bus.
- Each interface attached to the I/O bus contains an address decoder that monitors the address lines. When
  the interface detects an address to be its own, it activates the path between the bus and the device that it controls. All other peripherals are disabled. At the same time, a function code is provided to the control bus which is called I/O command.

MEMORYMAPPED I/O

- Memory and I/O have separate address space
- All address can be used by the memory
- Separate instruction control read and write operation in I/O and Memory
- In this I/O address are called ports.
- More efficient due to separate buses
- Larger in size due to more buses
- It is complex due to separate

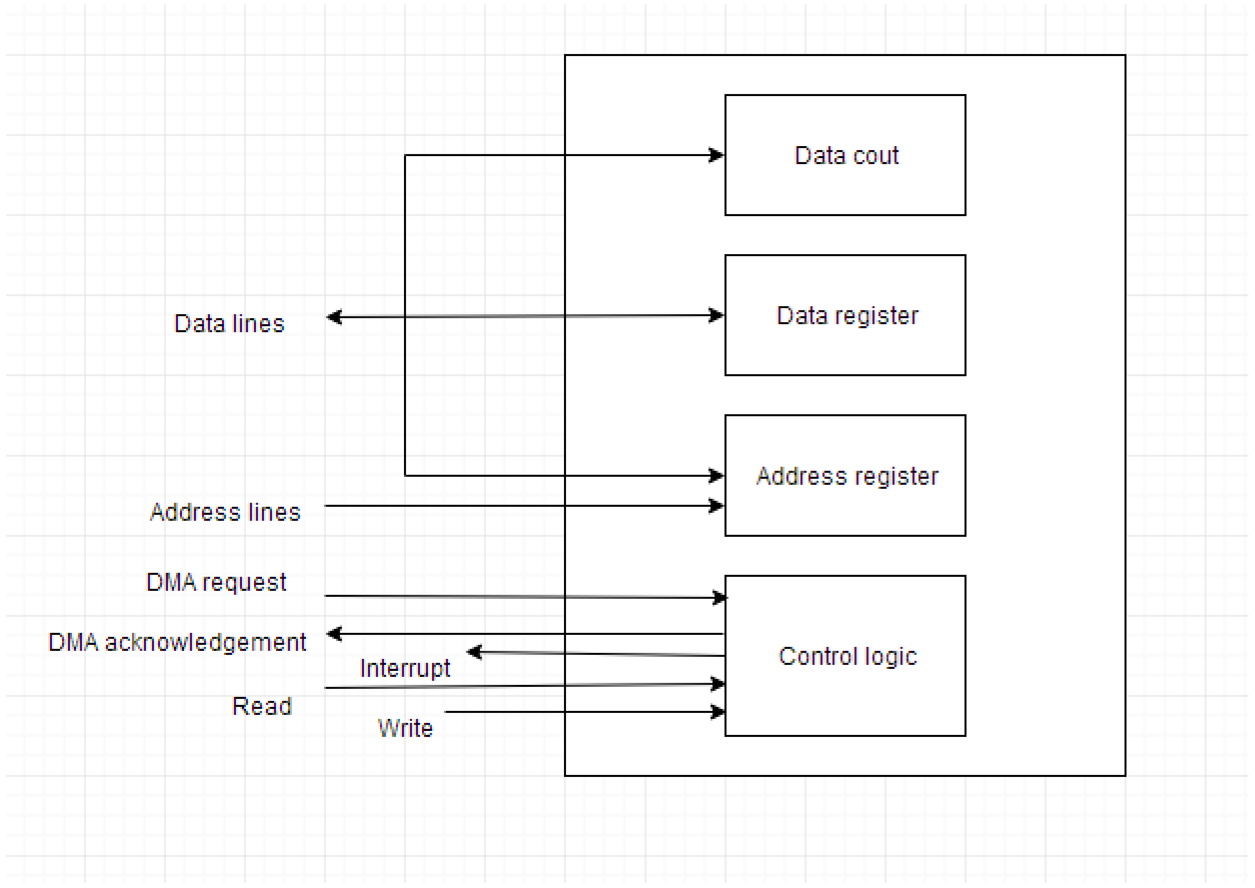
ISOLATED I/O

- Both have same address space
- Due to addition of I/O addressable memory become less for memory
- Same instructions can control both I/O and Memory
- Normal memory address are for both
- Lesser efficient
- Smaller in size
- Simpler logic is used as I/O is also treated as memory only.

DIRECT MEMORY ACCESS:

- Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This technique is known as **DMA**.
- In this, the interface transfer data to and from the memory through memory bus. A DMA controller manages to transfer data between peripherals and memory unit.
- Many hardware systems use DMA such as disk drive controllers, graphic cards, network cards and sound cards etc. It is also used for intra chip data transfer in multicore processors. In DMA, CPU would initiate the transfer, do other operations while the transfer is in progress and receive an interrupt from the DMA controller when the transfer has been completed.

BLOCK DIAGRAM OF DMA :



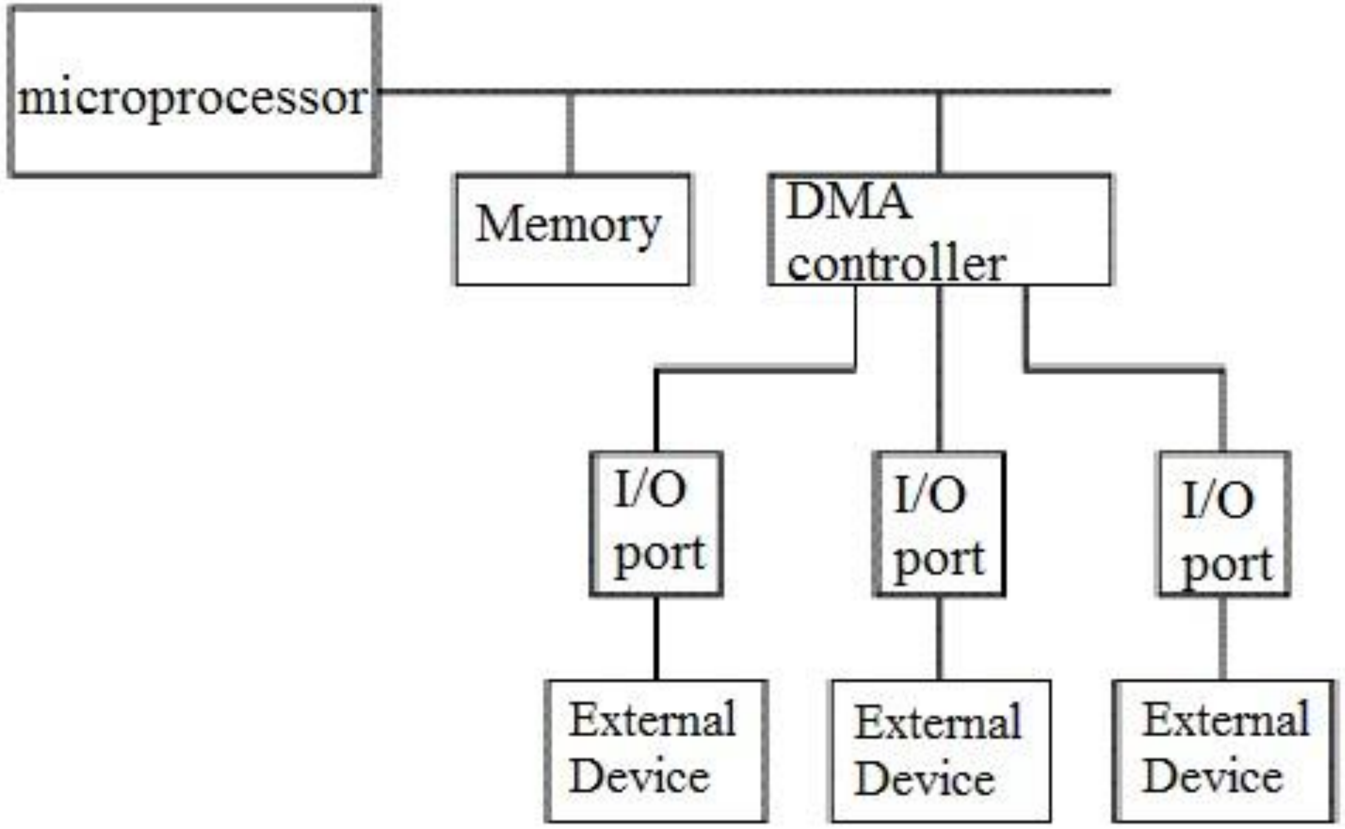
DMA CONTROLLER:

- The term DMA stands for direct memory access. The hardware device used for direct memory access is called the DMA controller. DMA controller is a control unit, part of I/O device's interface circuit, which can transfer blocks of data between I/O devices and main memory with minimal intervention from the processor. DMA controller provides an interface between the bus and the input-output devices. Although it transfers data without intervention of processor, it is controlled by the processor.

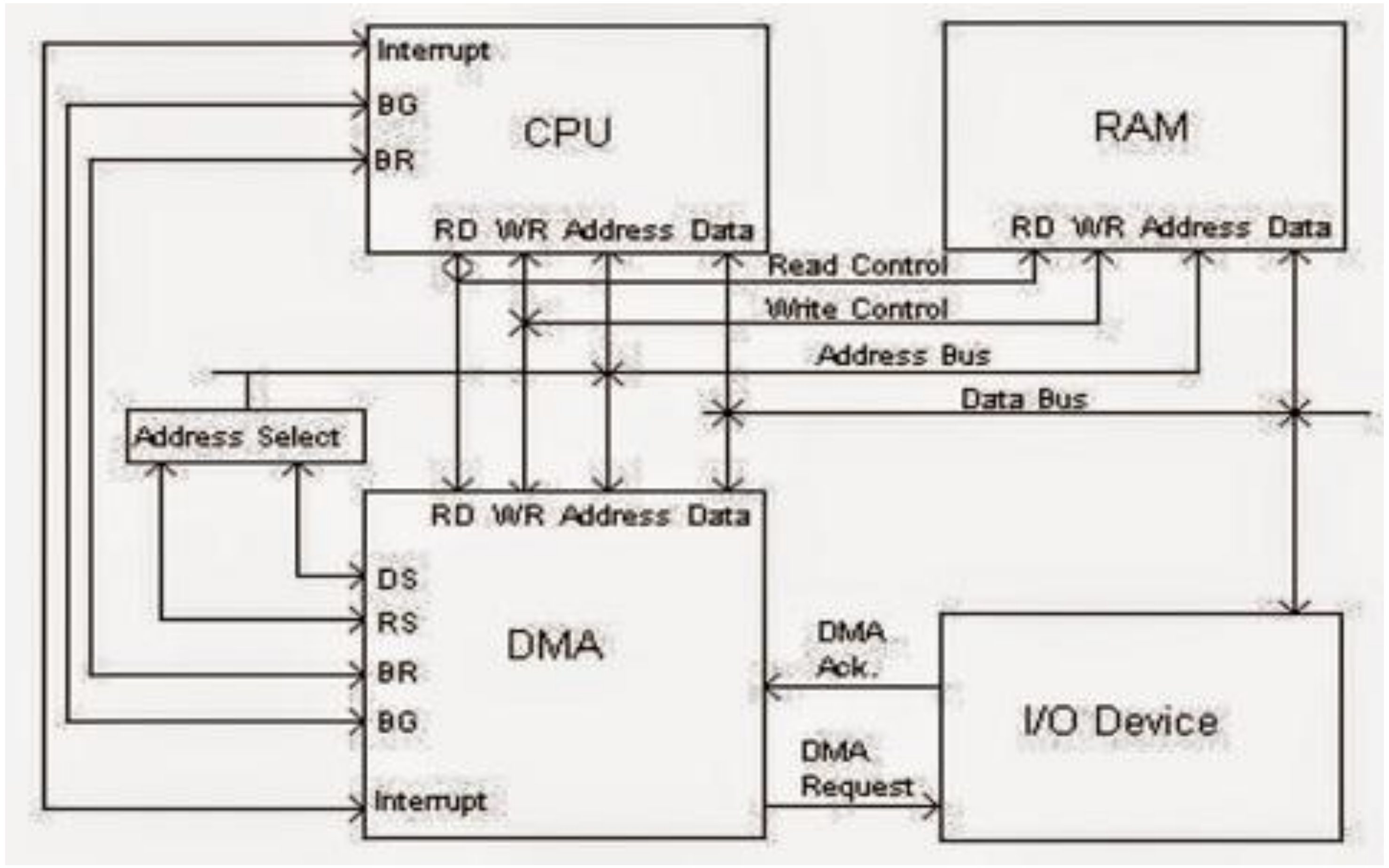
DMA CONTROLLER CONT.

- The processor initiates the DMA controller by sending the starting address, Number of words in the data block and direction of transfer of data .i.e. from I/O devices to the memory or from main memory to I/O devices. More than one external device can be connected to the DMA controller.

DMA CONTROLLER DIAGRAM



WORKING OF DMA CONTROLLER:



WORKING OF DMA CONTROLLER :

- DMA controller has to share the bus with the processor to make the data transfer. The device that holds the bus at a given time is called bus master. When a transfer from I/O device to the memory or vice verse has to be made, the processor stops the execution of the current program, increments the program counter, moves data over stack then sends a DMA select signal to DMA controller over the address bus.
- If the DMA controller is free, it requests the control of bus from the processor by raising the bus request signal. Processor grants the bus to the controller by raising the bus grant signal, now DMA controller is the bus master. The processor initiates the DMA controller by sending the memory addresses, number of blocks of data to be transferred and direction of data transfer. After assigning the data transfer task to the DMA controller, instead of waiting ideally till completion of data transfer, the processor resumes the execution of the program after retrieving instructions from the stack.

THANK YOU

This content is taken from the text books and reference books prescribed in the syllabus.