جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الامن السيبراني
Department of Cyber Security

Subject:

Digital Computer

Class:

THIRD

Lecturer:

Asst. raed alshmary

Lecture: (2)

Introduction

1.Digital Computer

The digital computer is a digital system that performs various computational tasks. Digital computers use the binary number system, which has two digits: 0 and 1. A binary digit is called a bit.
Information is represented in digital computers in groups of bits. By using various coding techniques, groups of bits can be made to represent not only binary numbers but also other discrete symbols, such as decimal digits or letters of the alphabet. By judicious use of binary arrangements and by using various coding techniques, the groups of bits are used to develop complete sets of instructions for performing various types of computations.
A computer system is sometimes subdivided into two functional entities

1- The hardware of the computer consists of all the electronic components and electromechanical devices that comprise the physical entity of the device.
2- Computer software consists of the instructions and data that the computer manipulates to perform various data-processing tasks.

The system software of a computer consists of a collection of programs whose purpose is to make more effective use of the computer. The programs included in a systems software package are referred to as the operating system.

2.Computer Hardware

The hardware of the computer is usually divided into three major parts, as shown in Fig(1

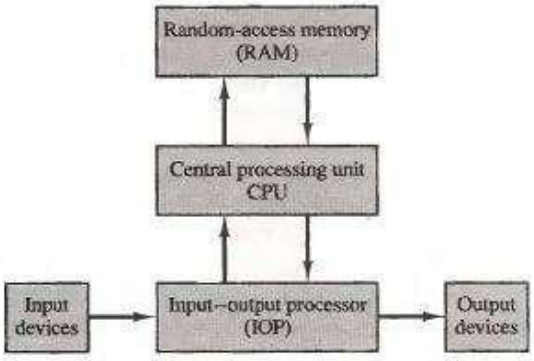


Figure 1 Block diagram of a digital computer.

The central processing unit (CPU) contains arithmetic and logic unit for manipulating data, a number of registers for storing data, and control circuits for fetching and executing instructions. The memory of a computer contains storage for instructions and data.

It is called ar andom- access memory (RAM) because the CPU can access any location in memory at random and retrieve the binary information within a fixed interval of time. The input and output processor (IOP) contains electronic circuits for communicating and controlling the transfer of information between the computer and the outside world. The input and output devices connected to the computer include keyboards, printers, terminals, magnetic disk drives, and other communication devices.

3. Computer Organization
Computer organization is concerned with the way the hardware components operate and the way they are connected together to form the computer system. The various components are assumed to be in place and the task is to investigate the organizational structure to verify that the computer parts operate as intended.

4. Computer Design
Computer design is concerned with the hardware design of the computer. Once the computer specifications are formulated, it is the task of the designer to develop hardware for the system. Computer design is concerned with the determination of what hardware should be used and how the parts should be connected.
This aspect of computer hardware is sometimes referred to as computer implementation.

5. Computer Architecture
Computer architecture is concerned with the structure and behavior of the computer as seen by the user. It includes the information formats, the instruction set, and techniques for addressing memory. The architectural design of a computer system is concerned with the 20 specifications of the various functional modules, such as processors and memories, and structuring them together into a computer system.

Instruction Set Architecture

1. Opcodes:
Consists of operate instructions: as logical and arithmetical instructions, Data movement instructions and Control instructions

2. Data types: consists of 8, 16, 32, and 64 bits

3. Addressing modes: consists of:
- Operands specified
- Next instruction to execute is specified
- Architecture-specific
- An instruction can use several addressing modes

6.1. Register Transfer Language

Digital systems vary in size and complexity from a few integrated circuits to a complex of interconnected and interacting digital computers. Digital system design invariably uses a modular approach.
The modules are constructed from such digital components as registers, decoders, arithmetic elements, and control logic. The various modules are interconnected with common data and control paths to form a digital computer system.
Digital modules are best defined by the registers they contain and the operations that are performed on the data stored in them. The operations executed on data stored in registers are called micro operations(MO).
A micro operation is an elementary operation performed on the information stored in one or more registers. The result of the operation may replace the previous binary information of a register or may be transferred to another register. Examples of micro operations are shift, count, dear, and load.

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

1- The set of registers it contains and their function.

2- The sequence of micro operations performed on the binary information stored in the registers.

3- The control that initiates the sequence of micro operations.
The symbolic notation used to describe the micro operation transfers among registers is called a *register transfer language*.

The term "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 register.

The word "language" is borrowed from programmers, who apply this term to programming languages.

A *register transfer language* is a system for expressing in symbolic form the micro operation sequences among the registers of a digital module.

6.2. Register Transfer

Computer registers are designated by capital letters (sometimes followed by numerals) to denote the function of the register.

For example:
MAR: memory address register
PC: program counter
IR: instruction register
R1: processor register
The representation of registers in block diagram form is shown in Fig(2):

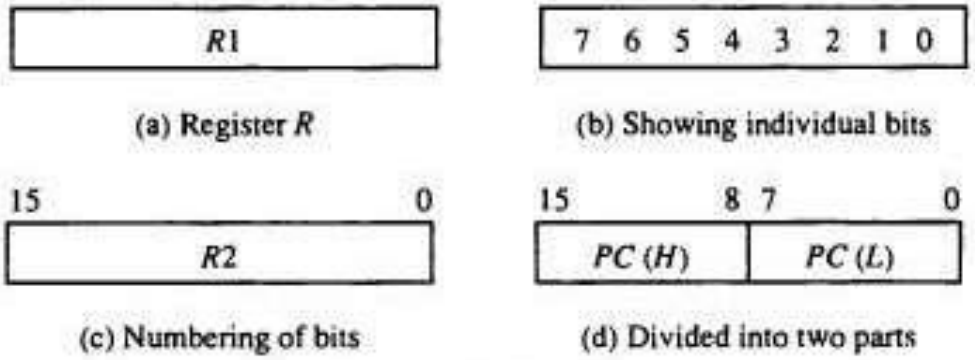


Figure 2 Block diagram of register.

a- Rectangular box with the name of the register inside.

b- The individual bits.

c- The numbering of bits in a 16-bit register can be marked on top of the box.
d- 16-bit register is partitioned into two parts.
Bits 0 through 7 are assigned the symbol L (for low byte) and bits 8 through 15 are assigned the symbol H (for high byte).

d- The name of the 16-bit register is PC. The symbol PC (0-7) or PC(L) refers to the low order byte and PC (8-15) or PC(H) to the high-order byte.

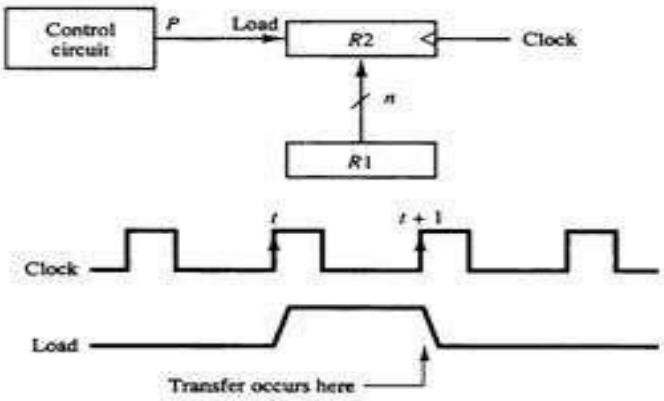
Information transfer from one register to another is designated in symbolic form by means of a *replacement* operator. The statement:

$R2 \leftarrow R1$

Denotes a transfer of the content of register Rl into register R2. It designates a replacement of the content of R2 by the content of Rl. By definition, the content of the source register Rl does not change after the transfer.
If we want the transfer to occur only under a predetermined control condition.
This can be shown by means of an if-then statement.
$If\ (P = 1)\ then\ (R2 \leftarrow R1)$

where P is a control signal generated in the control section. It is sometimes convenient to separate the control variables from the register transfer operation control function by specifying a control function.
$P: R2 \leftarrow R1$

The control condition is terminated with a colon. It symbolizes) (the requirement that the transfer operation be executed by the hardware only if P = 1.



Transfer from R1 to R2 when $P = 1$.

To separate two or more operations that is executed at the same time by using the *comma* as the statement:

$T: R2 \leftarrow R1, R5 \leftarrow R3$
The basic symbols of the register transfer notation are listed in Table (1) Registers are denoted by capital letters, and numerals may follow the letters. Parentheses are used to denote a part of a register by specifying the range of bits or by giving a symbol name to a portion of a register.

TABLE 1 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	$R2 \leftarrow R1$
Comma ,	Separates two microoperations	$R2 \leftarrow R1, R1 \leftarrow R2$

6.3. Bus and Memory Transfers

A typical digital computer has many registers, and paths must be provided to transfer information from one register to another. The number of wires will be excessive if separate lines are used between each register and all other registers in the system.

A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during each particular register transfer. The multiplexers select the source register whose binary information is then placed on the bus. For example, the construction of a bus system for four registers is shown in Fig (3) Each register has four bits, numbered 0 through 3.

The bus consists of four 4x1(4-input-one output) multiplexers each having four data inputs, 0 through 3, and two selection inputs, S1 and S0. (00,01,10,11)

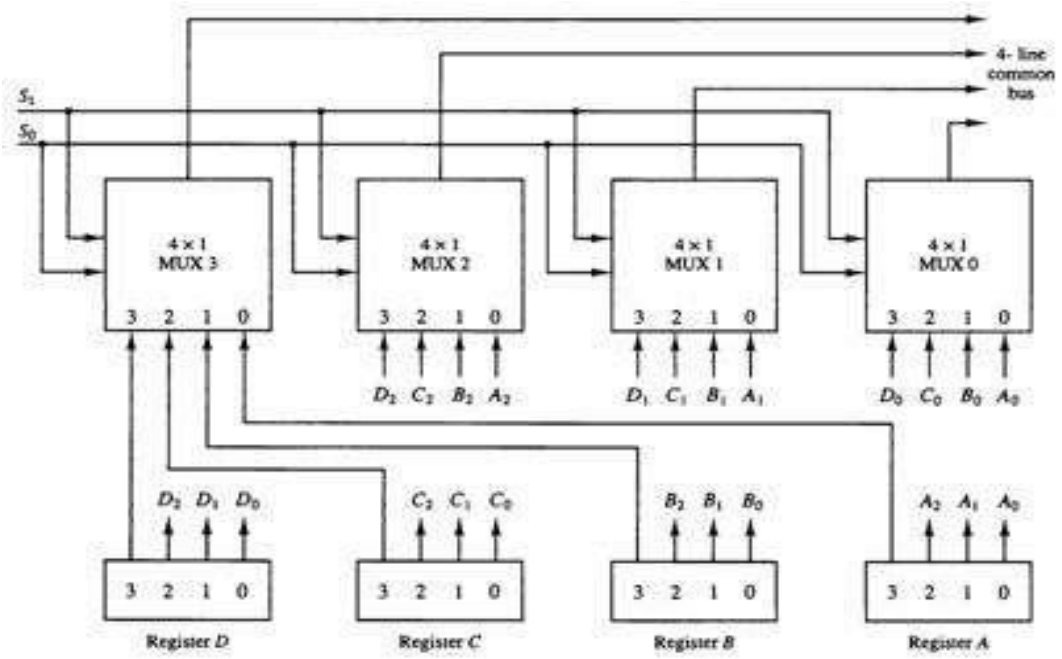


Fig 3 Bus system for four registers

The table (2) shows the register that is selected by the bus for each of the four possible binary values of the selection lines.

Table 2 Function for Bus of Fig

S_1	S_0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

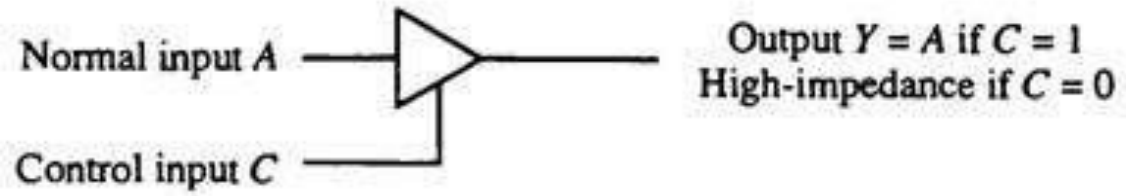
The symbolic statement for a bus transfer may mention the bus or its presence may be implied in the statement. When the bus is including in the statement, the register transfer is symbolized as follows:

$Bus \leftarrow C$, $R1 \leftarrow Bus$

The content of register C is placed on the bus, and the content of the bus is loaded into register Rl by activating its load control input. If the bus is known to exist in the system, it may be convenient just to show the direct transfer.

$R1 \leftarrow C$

A bus system can be constructed with three-state gates. The graphic symbol of a three state buffer gate is shown:



Graphic symbols for three-state buffer.

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 as shown in Fig (4). Each group of four buffers receives one significant bit from the four registers.

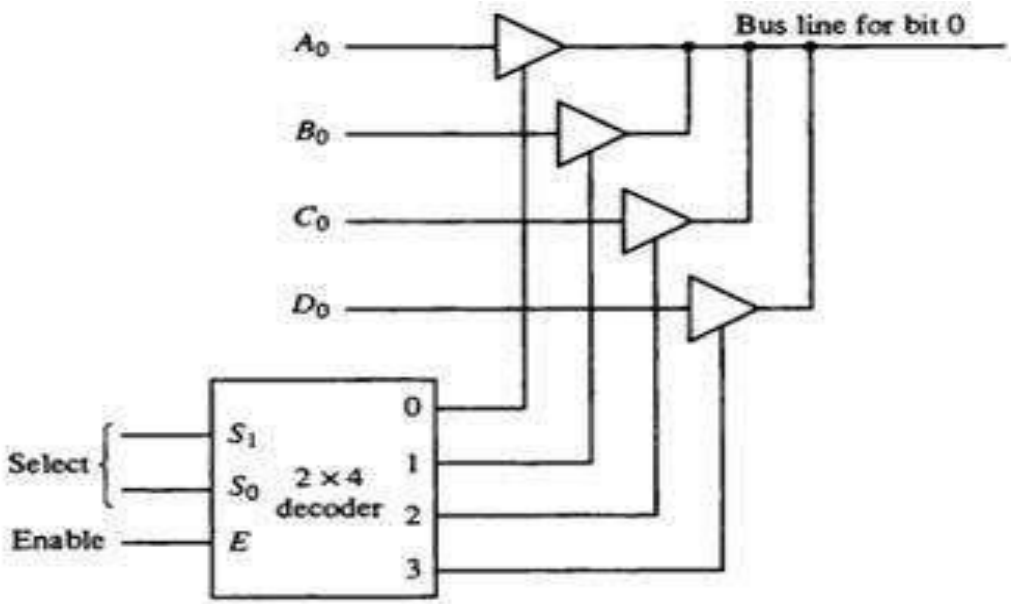


Figure 4 Bus line with three state-buffers.

The transfer of information from a memory word to the outside environment is called a read operation. The transfer of new information to be stored into the memory is called a write operation. Consider a memory unit that receives the address from a register, called the Address Register, symbolized by AR. The data are transferred to another register, called the Data Register, symbolized by DR.

$Read: DR \leftarrow M[AR]$

The write operation transfers the content of a data register to a memory word M selected by the address.

$Write: M[AR] \leftarrow DR$

6.4. Arithmetic Microo perations

The arithmetic operations are listed in the Table(3):

TABLE 3 Arithmetic Microoperations

Symbolic designation	Description
$R3 \leftarrow R1 + R2$	Contents of $R1$ plus $R2$ transferred to $R3$
$R3 \leftarrow R1 - R2$	Contents of $R1$ minus $R2$ transferred to $R3$
$R2 \leftarrow \overline{R2}$	Complement the contents of $R2$ (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement the contents of $R2$ (negate)
$R3 \leftarrow R1 + \overline{R2} + 1$	$R1$ plus the 2's complement of $R2$ (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of $R1$ by one
$R1 \leftarrow R1 - 1$	Decrement the contents of $R1$ by one

The multiply and divide are not listed in Table (3), these two operations are valid arithmetic operations but are not included in the basic set of micro operations. In most computers, the multiplication operation is implemented with a sequence of add and shift micro operations. Division is implemented with a sequence of subtract and shift micro operations.

The digital circuit that generates the arithmetic sum of two binary numbers of any length is called a binary adder as shown in Fig (5).

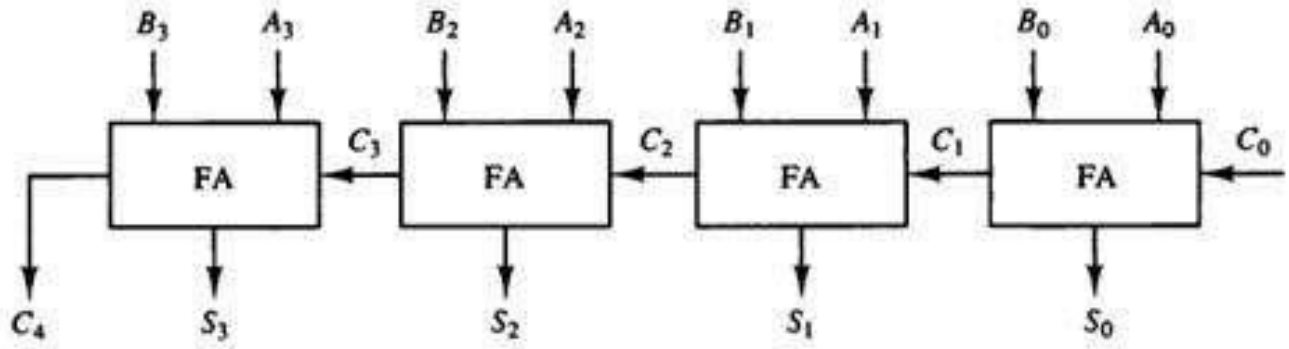


Figure 5 4-bit binary adder.

The *addition and subtraction operations* can be combined into one common circuit by including an exclusive-OR gate with each full-adder as shown in Fig (6).

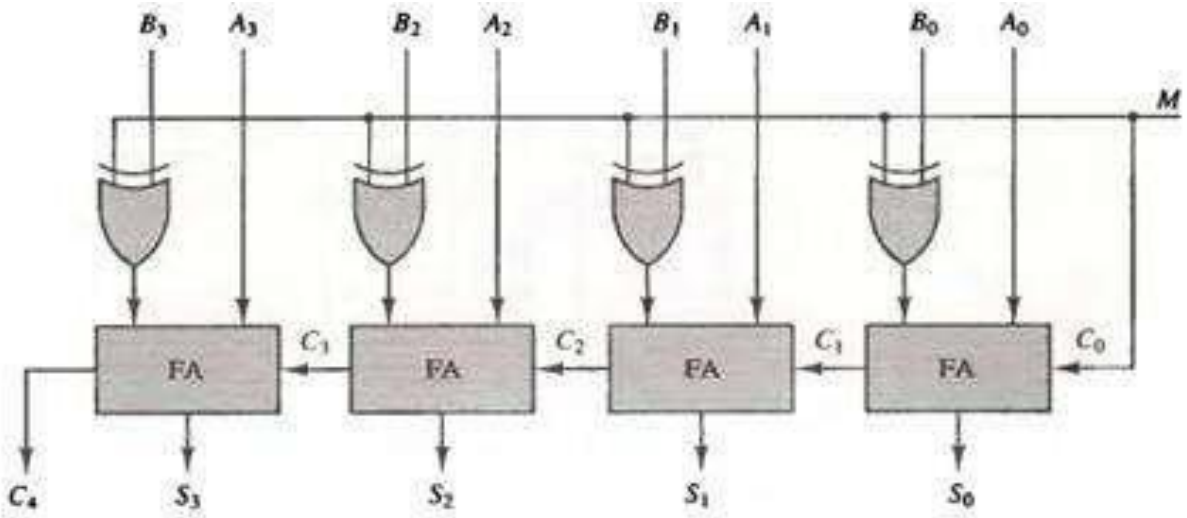


Figure 6 4-bit adder-subtractor.

The increment micro operation adds one to a number in a register. For example, if a 4-bit register has a binary value 0110, it will go to 0111 after it is incremented. The diagram of a 4bit combinational circuit incremented is shown in Fig(7):
(HA means Half Adder)

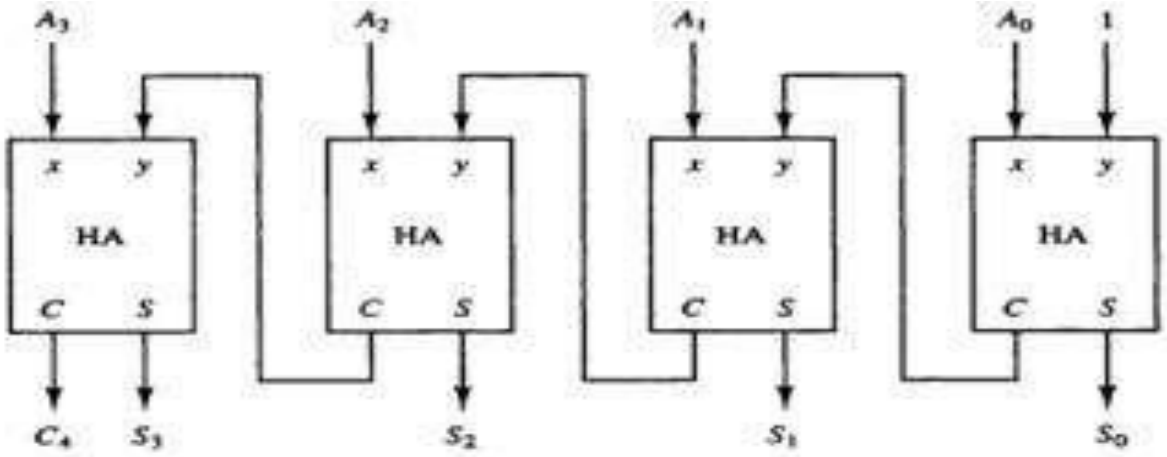


Figure 7 4-bit binary incrementer.

The arithmetic micro operations listed in the Table 3 can be implemented in one composite arithmetic circuit. The basic component of an arithmetic circuit is the parallel adder. By controlling the data inputs to the adder, it is possible to obtain different types of arithmetic operations as shown in Fig (8).

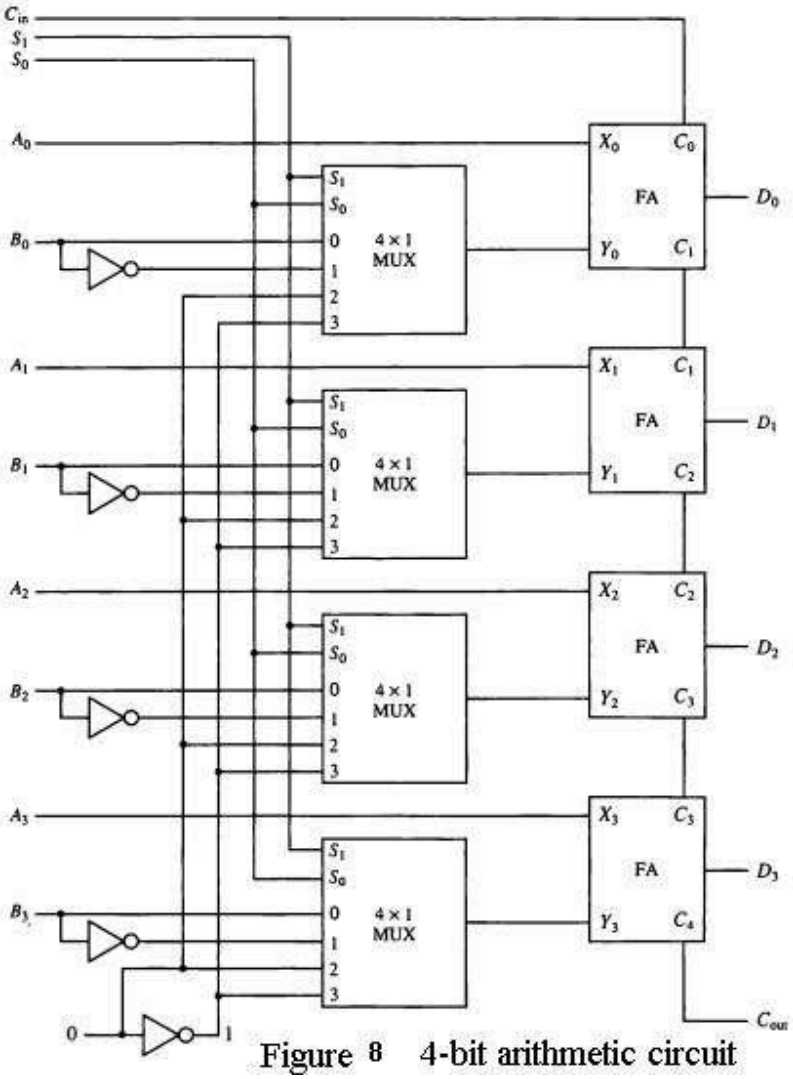


Figure 8 4-bit arithmetic circuit

It is possible to generate the eight arithmetic micro operations listed in Table (4):

TABLE 4 Arithmetic Circuit Function Table

Select			Input	Output	
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	Microoperation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

6.5. Logic Micro Operations

Logic micro operations specify binary operations for strings of bits stored in registers.
These operations consider each bit of the register separately and treat them as binary variables.
For example, the exclusive-OR micro operation with the contents of two registers Rl and R2 is symbolized by the statement:

$P: R1 \leftarrow R1 \oplus R2$

It specifies a logic micro operation to be executed on the individual bits of the registers provided that the control variable P = 1. As a numerical example, assume that each register has four bits.
Let the content of Rl be 1010 and the content of R2 be 1100. The exclusive-OR micro operation stated above symbolizes the following logic computation:

$$\begin{array}{ll} 1010 & \text{Content of } R1 \\ \underline{1100} & \text{Content of } R2 \\ 0110 & \text{Content of } R1 \text{ after } P = 1 \end{array}$$

There are 16 different logic operations that can be performed with two binary variables.
They can be determined from all possible truth tables obtained with two binary variables as shown in Table (5):

TABLE 5 Truth Tables for 16 Functions of Two Variables

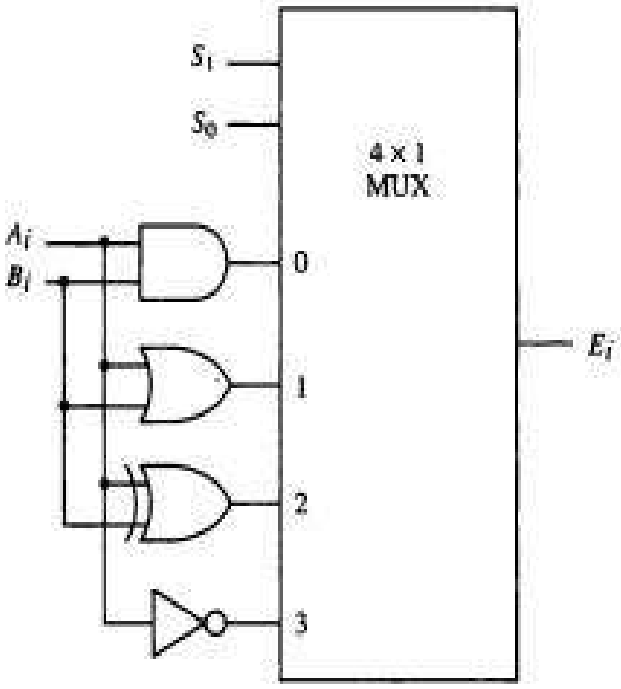
x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

The 16 Boolean functions of two variables x and y are expressed in algebraic form in the first column of Table (6):

TABLE 6 Sixteen Logic Microoperations

Boolean function	Microoperation	Name
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \overline{B}$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow \overline{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	Transfer B
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	Exclusive-OR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x + y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	Exclusive-NOR
$F_{10} = y'$	$F \leftarrow \overline{B}$	Complement B
$F_{11} = x + y'$	$F \leftarrow A \vee \overline{B}$	
$F_{12} = x'$	$F \leftarrow \overline{A}$	Complement A
$F_{13} = x' + y$	$F \leftarrow \overline{A} \vee B$	
$F_{14} = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow$ all 1's	Set to all 1's

The diagram shows (Fig 9-a) one typical stage with subscript i. For a logic circuit with n bits, the diagram must be repeated n times for i = 0,1, 2,..., n - 1. The selection variables are applied to all stages.
The function table in Fig.(9-b) lists the logic micro operations obtained for each combination of the selection variables.



(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Function table

Figure 9 One stage of logic circuit.

6.6. Shift Micro operations

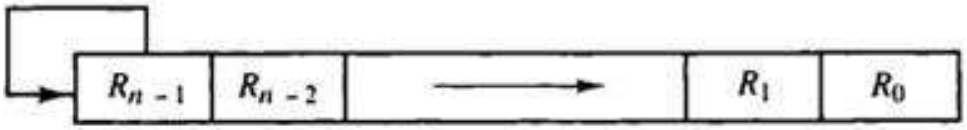
Shift micro operations are used for serial transfer of data. The contents of a register can be shifted to the left or the right. There are three types of shifts: logical, circular, and arithmetic.

The symbolic notation for the shift micro operations is shown in Table (7):

TABLE 7 Shift Microoperations

Symbolic designation	Description
$R \leftarrow$ shl R	Shift-left register R
$R \leftarrow$ shr R	Shift-right register R
$R \leftarrow$ cil R	Circular shift-left register R
$R \leftarrow$ cir R	Circular shift-right register R
$R \leftarrow$ ashl R	Arithmetic shift-left R
$R \leftarrow$ ashr R	Arithmetic shift-right R

An arithmetic shift is a micro operation that shifts a signed binary number to the left or right.
The arithmetic shift-left inserts a 0 into R0, and shifts all other bits to the left. The initial bit of Rn-1 is lost and replaced by the bit from Rn-2 . A sign reversal occurs if the bit in Rn-1 changes in value after the shift and caused an overflow.
The arithmetic shift-right leaves the sign bit unchanged and shifts the number (including the sign bit) to the right.

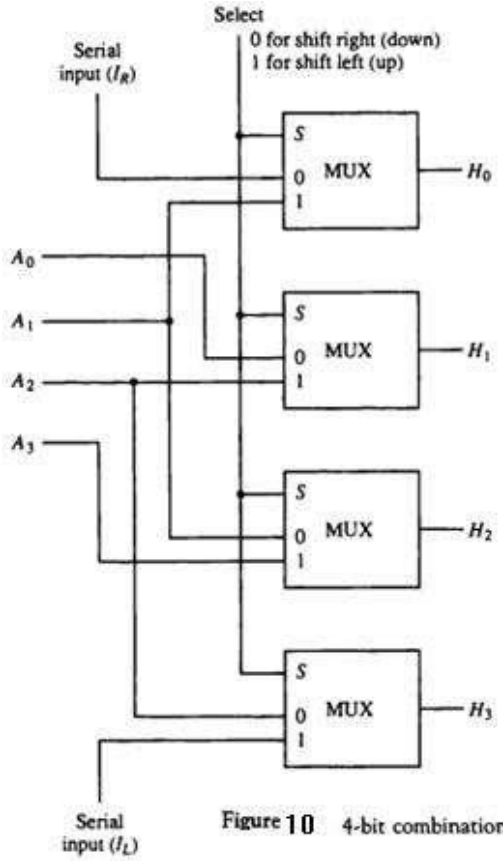


Sign bit

Arithmetic shift right.

Ex: If the content of 8 bits register is (10100011). What is the result of the operation after executing to the register:
a. shl R: shift left register by 3. b. cil R : circular shift left register by 3.
c. ashl R: arithmetic shift left register by 3. d. ashr R: arithmetic shift right register by 3.
Ans:
(a) 00011000. (b) 00011101. (c) 00011000. Overflow (d) 11110100
A combinational circuit shifter can be constructed with multiplexers as shown in Fig (10).
The 4-bit shifter has four data inputs, A0 through A3, and four data outputs, H0 through H3.

There are two serial inputs, one for shift left (IL) and the other for shift right (IR).



Function table

Select	Output			
S	H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

Figure 10 4-bit combinational circuit shifter.

6.7. Arithmetic Logic

Arithmetic Logic Shift Unit

Computer systems employ a number of storage registers connected to a common
operational unit called an arithmetic logic unit, abbreviated ALU.

The arithmetic, logic, and shift circuits introduced in previous sections can be combined into one ALU with common selection variables.
One stage of an arithmetic logic shift unit is shown in Fig (11) with the functional
table(8):

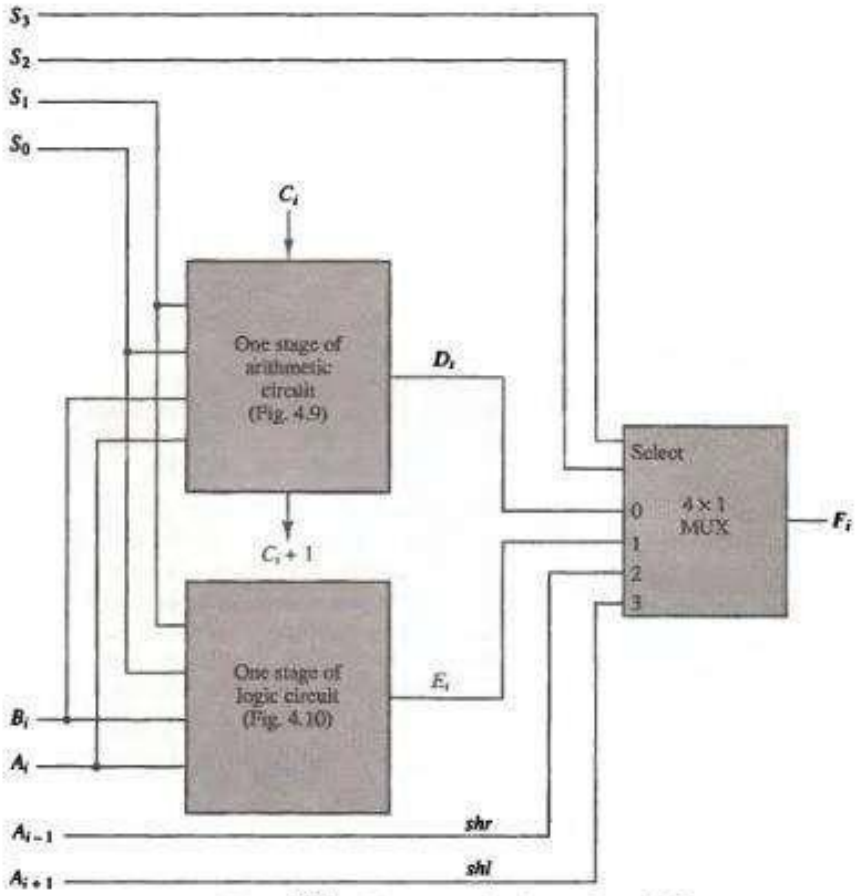


Figure 11 One stage of arithmetic logic shift unit.

TABLE 8 Function Table for Arithmetic Logic Shift Unit

Operation select						
S_3	S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A + 1$	Increment A
0	0	0	1	0	$F = A + B$	Addition
0	0	0	1	1	$F = A + B + 1$	Add with carry
0	0	1	0	0	$F = A + \overline{B}$	Subtract with borrow
0	0	1	0	1	$F = A + \overline{B} + 1$	Subtraction
0	0	1	1	0	$F = A - 1$	Decrement A
0	0	1	1	1	$F = A$	Transfer A
0	1	0	0	×	$F = A \wedge B$	AND
0	1	0	1	×	$F = A \vee B$	OR
0	1	1	0	×	$F = A \oplus B$	XOR
0	1	1	1	×	$F = \overline{A}$	Complement A
1	0	×	×	×	$F =$ shr A	Shift right A into F
1	1	×	×	×	$F =$ shl A	Shift left A into F

The way that the interrupt is handled by the computer can be explained by means of the flowchart of Fig(19). An interrupt flip-flop R is included in the computer.

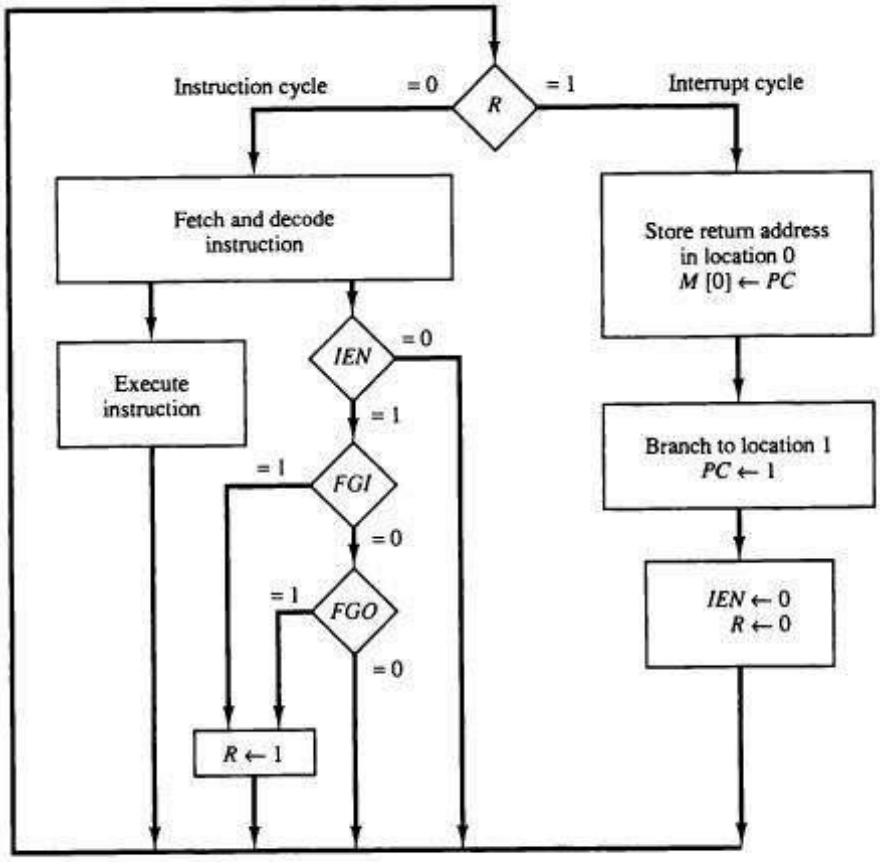


Figure 19 Flowchart for interrupt cycle.

7. Design of Basic Computer

The basic computer consists of the following hardware components:

1. **A memory unit with 4096 words of 16 bits each**
2. **Nine registers: AR, PC, DR, AC, IR, TR, OUTR, INPR, and SC**
3. **Seven flip-flops: I, S, E, R, IEN, FGI, and FGO**
4. **Two decoders: a 3 x 8 operation decoder and a 4 x 16 timing decoder**
5. **A 16-bit common bus**
6. **Control logic gates**
7. **Adder and logic circuit connected to the input of AC The outputs of the control logic circuit**

are:

1. Signals to control the inputs of the nine registers
2. Signals to control the read and write inputs of memory
3. Signals to set, clear, or complement the flip-flops
4. Signals for S2, S1, and S0 to select a register for the bus
5. Signals to control the AC adder and logic circuit.

8. Design of Accumulator Logic

The circuits associated with the AC register are shown in Fig(20). The adder and logic circuit has three sets of inputs.

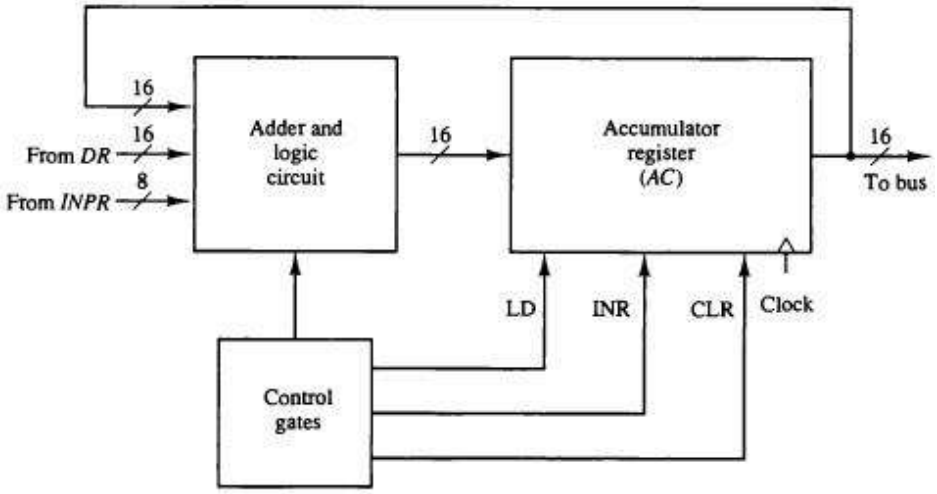


Figure 20 Circuits associated with AC.

In order to design the logic associated with AC, it is necessary to go over the register transfer statements and extract all the statements that change the content of AC.

$AC \leftarrow AC \wedge DR$	AND with *DR*
$AC \leftarrow AC + DR$	Add with *DR*
$AC \leftarrow DR$	Transfer from *DR*
$AC(0\text{–}7) \leftarrow INPR$	Transfer from *INPR*
$AC \leftarrow \overline{AC}$	Complement
$AC \leftarrow \text{shr } AC, \quad AC(15) \leftarrow E$	Shift right
$AC \leftarrow \text{shl } AC, \quad AC(0) \leftarrow E$	Shift left
$AC \leftarrow 0$	Clear
$AC \leftarrow AC + 1$	Increment