PPT ON
COMPUTER ORGANIZATION
V SEM (IARE-R16)

UNIT I
INTRODUCTION

CLO's	Course Learning outcomes
CLO1	Describe the various components like input/output units, memory unit, control unit, arithmetic logic unit connected in the basic organization of a computer.
CLO2	Understand the concepts associated with the computer organization.
CLO3	Describe various data representations and explain how arithmetic and logical operations are performed by computers.
CLO4	Understand instruction types, addressing modes and their formats in the assembly language programs.

Computing and Computers

•The word computer is derived from the word compute. Compute means to calculate. The computer was originally defined as a super fast calculator. It had the capacity to solve complex arithmetic and scientific problems at very high speed.

•The information in one form which is presented to the computer is the input
information or **input data**.

•Information in another form is presented by the computer after performing a process on it. This information is the output information or **output data.**

Elements of a computer

Human	Computer
Like human beings has ears, nose, eyes etc.	Computers have input devices such as keyboard, scanner, touch screen, mouse etc to get information.
Like we remember things	Computer also stores information
We recollect certain information as required	The computer also retrieves information when times
We express ourselves by speech, writing etc	Computer expresses through screen, Printouts etc which We call as output
When we watch, hear, learn certain things and analyze	with the help of software, computer also can analyze Information and draw conclusions
The place where we store, analyze	The computer brain is known as CPU conclude information is known as the brain (Central Processing Unit) where it analyses information.

Limitations of Computers

- Although the computers of today are highly intelligent and sophisticated they have their own limitations. The computer cannot think on its own, since it does not have its own brain.

- It can only do what is has been programmed to do. It can execute only those jobs that can be expressed as a finite set of instructions to achieve a specific goal. Each of the steps has to be clearly defined.

- The computers do not learn from previous experience nor can they arrive at a conclusion without going through all the intermediate steps. However the impact of computers on today‘s society in phenomenal and they are today an important part of the society.

Evolution of computers

First Generation Computers

1. Mark I
2. ENIAC
3. EDVAC
4. UNIVAC

Evolution of computers

Mid-1950s: Transistor Computers (Second Generation)

- The development of **transistors** led to the replacement of vacuum tubes, and resulted in significantly smaller computers.
- In the beginning, they were less reliable than the **vacuum tubes** they replaced, but they also consumed significantly less power. **IBM 350 RAMAC** used disk drives.

- These transistors also led to developments in **computer peripherals**. The first disk drive, the **IBM 350 RAMAC**, was the first of these introduced in 1956.

Evolution of computers

1960s: The Microchip and the Microprocessor (Third Generation Computers)

- The microchip (or integrated circuit) is one of the most important advances in computing technology. Many overlaps in history existed between microchip-based computers and transistor-based computers throughout the 1960s. Microchips allowed the manufacturing of smaller computers.
- The microchip spurred the production of minicomputers and microcomputers, which were small and inexpensive enough for small businesses and even individuals to own.
- The microchip also led to the microprocessor, another breakthrough technology that was important in the development of the personal computer.

Evolution of computers

1970s: Personal Computers (Fourth Generation)

- The first personal computers were built in the early 1970s. Most of these were runs, and worked based on small-scale integrated circuits and multi-chip CPUs.
- The **Commodore PET** was a personal computer in the 70s. The Altair 8800 was the first popular computer using a single-chip microprocessor. Clones of this machine quickly cropped up, and soon there was an entire market based on the design and architecture of the 8800.
- It also spawned a club based around hobbyist computer builders, the **Homebrew Computer Club**. 1977 saw the rise of the "**Trinity**" the **Commodore PET**, the **Apple II**, and the **Tandy Corporation's TRS-80**. These three computer models eventually went on to sell millions.

Evolution of computers

2000s: The Rise of Mobile Computing (Present and Beyond)

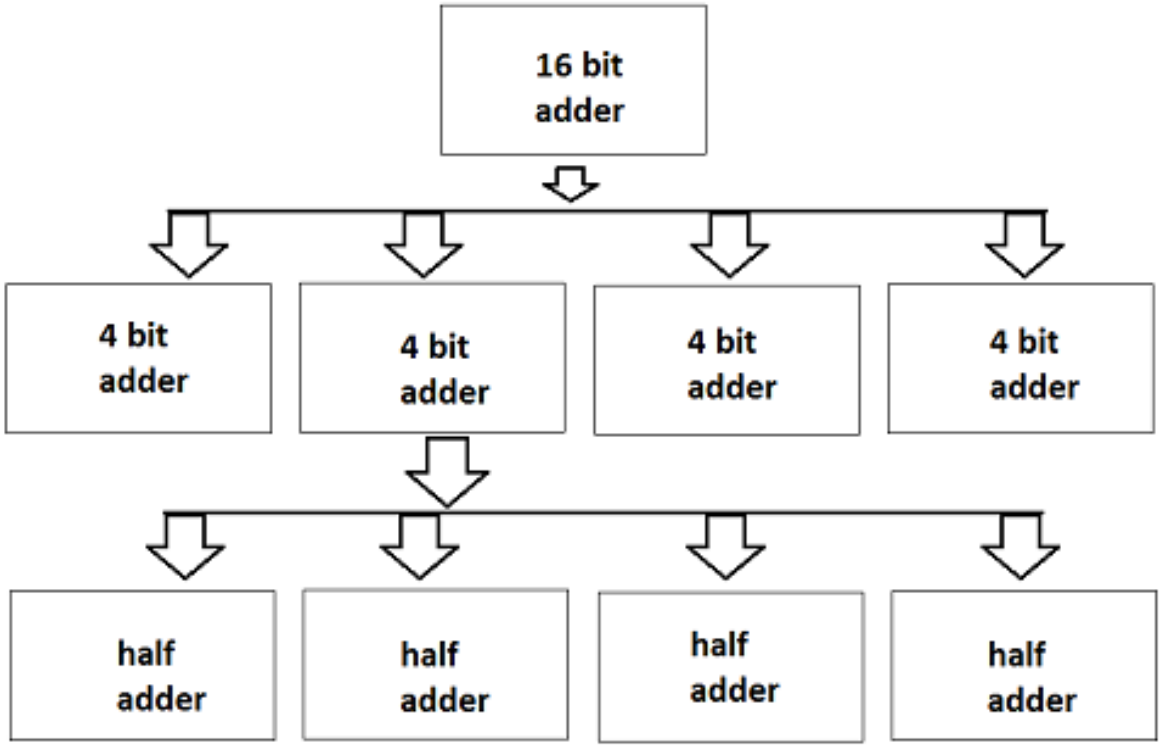
Mobile computing

- It is one of the most recent major milestones in the history of computers. Many smart phones today have higher processor speeds and more memory than desktop **PCs** had even ten years ago.
- With phones like the **iPhone** and the **Motorola Droid**, it's becoming possible to perform most of the functions once reserved for desktop PCs from anywhere. The **Droid** is a smart phone capable of basic computing tasks such as emailing and web browsing.

Design Hierarchy-Structural

- The design hierarchy involves the principle of "Divide and Conquer." It is nothing but dividing the task into smaller tasks until it reaches to its simplest level.
- This process is most suitable because the last evolution of design has become so simple that its manufacturing becomes easier.
- We can design the given task into the design flow process's domain (Behavioral, Structural, and Geometrical).
- To understand this, let’s take an example of designing a 16-bit adder, as shown in the figure below.

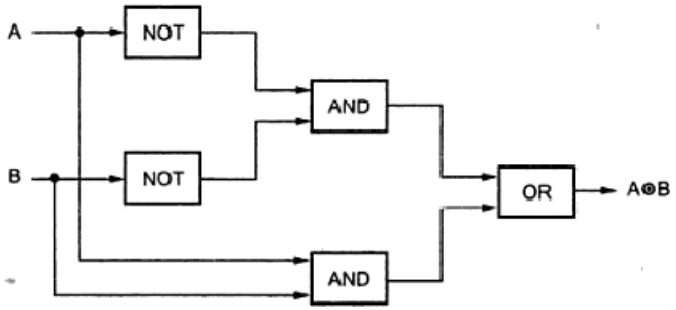
Design Hierarchy-Structural



System design

System Representation

- We can represent a system using a graph or a block diagram. A computer system is usually represented by a block diagram.
- A system has its own structure and behavior. The structure and behavior are the two properties of the system.
- We can define the structure of a system as the abstract graph consisting of its block diagram with no functional information, as shown in Fig. below



A block diagram representing EX-NOR logic circuit

System design

- As shown in figure, the structure gives the components and their interconnection. A behavioral description, on the other hand, describes the function of each component and thus the function of the system.
- The behavior of the system may be represented by Boolean function or by truth table in case of logic circuit.

-The behavior of logic circuits can also be described by Hardware description language such as VHDL. They can provide precise, technology-independent descriptions of digital circuits at various levels of abstraction, primarily the gate and register levels.

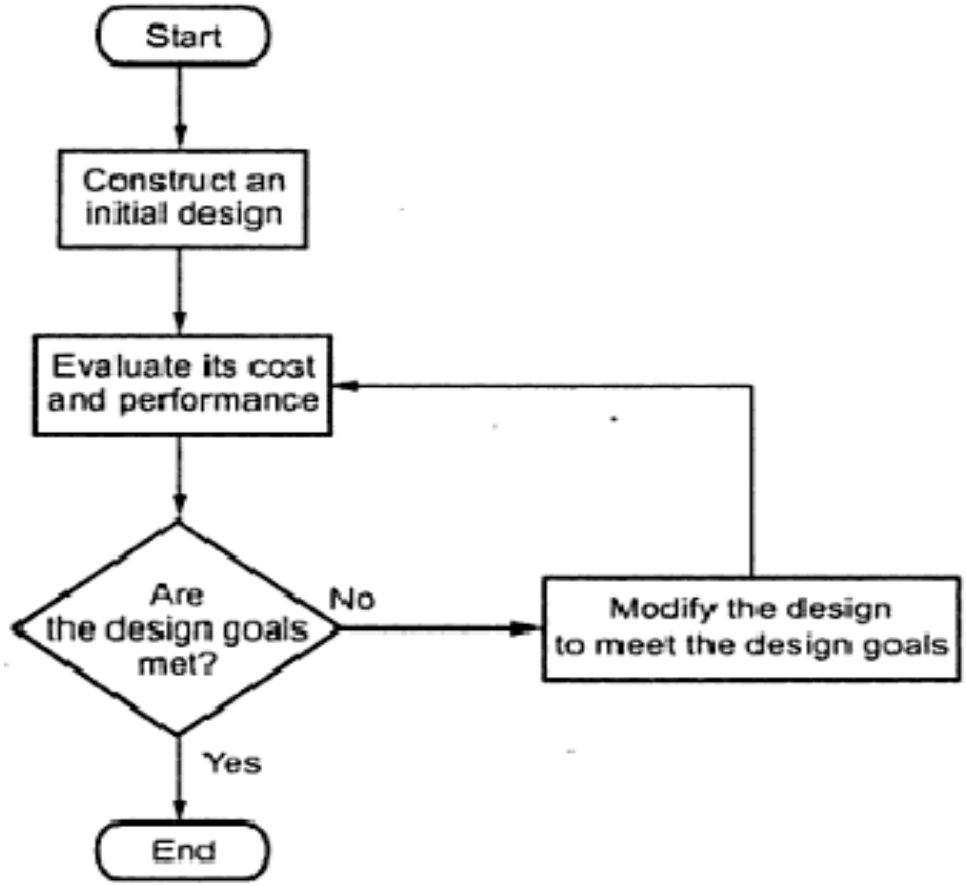
Design Process

- For a given system's structure, the task of determining its function or behavior is termed analysis. On the other hand, the problem of determining a system structure that exhibits a given behavior is design or synthesis.

- The design process starts with the construction of initial design. In this process, with given a desired range of behavior and set of available components we have to determine a structure (Design).

- The next step is to evaluate its cost and performance. The cost and performance should be in the acceptable range. Then we have to confirm that whether the formed structure achieves the desired behavior. If not we have to modify the design to meet the design goals. The Fig. below illustrates the design process.

Design Process



Design process

Computer-aided Design

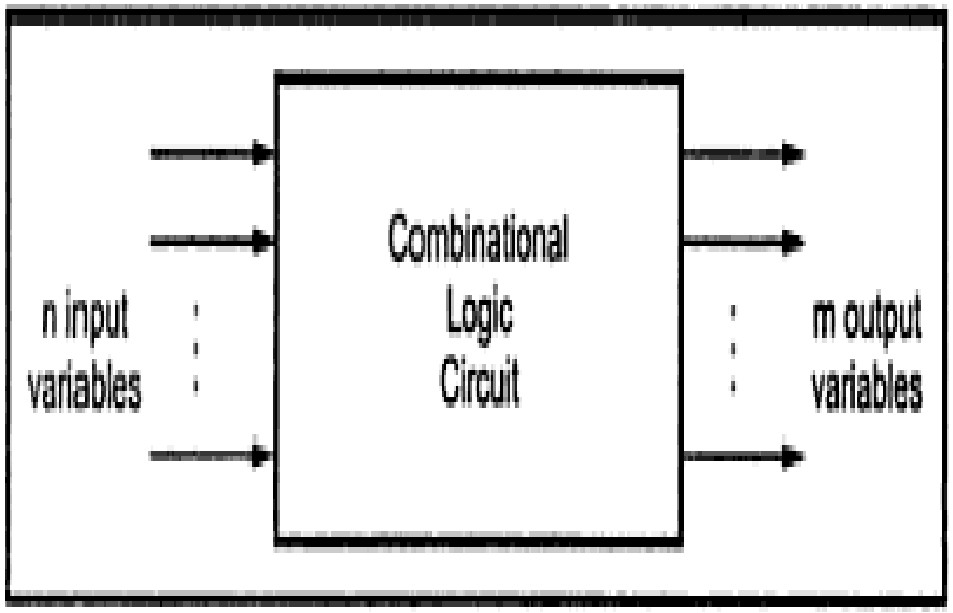
- The computer-aided design (CAD) tools provides designers with a range of programs to support their design goals. They are used to automate fully or party the more tedious design and evaluate its steps. They contribute mainly in three important ways to the overall design process.
    - CAD editors or translators convert design data into forms such as HDL descriptions or schematic diagrams, which can be efficiently processed by the humans, computers or both.
    - Simulators create the computer model for the design and can mimic the design's behavior. It helps designer to determine how well the design meets various performance and cost goals.
    - Synthesizers derive structures that implement all or part of some design step.

Gate Level Design

- Gate level design concerned with processing binary variables: 0 and 1. In this level, the design components are logic gates and flip-flops. Logic gates are memoryless elements; however flip-flops are bit storage devices. Using these elements gate level design is used to build the combinational and sequential circuits.
- When logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting circuit is called combinational logic. In combinational logic, the output variables are at all times dependent on the combination of input variables.
- A combinational circuit consists of input variables, logic gates, and output variables. The logic gates accept signals from the input variables and generate output signals. This process transforms binary information from the given input data to the required output data. Fig. below shows the block diagram of a combinational circuit. As shown in figure, the combinational Circuit accepts n-input binary variables and generates output variables depending on the logical combination of gates.

z

Gate Level Design



Block diagram of a combinational circuit

Gate Level Design

The design of combinational circuits starts from the outline of the problem statement and ends in a logic circuit diagram or a set of Boolean functions from which the logic diagram can be easily obtained. The design procedure of the combinational circuit involves following steps :

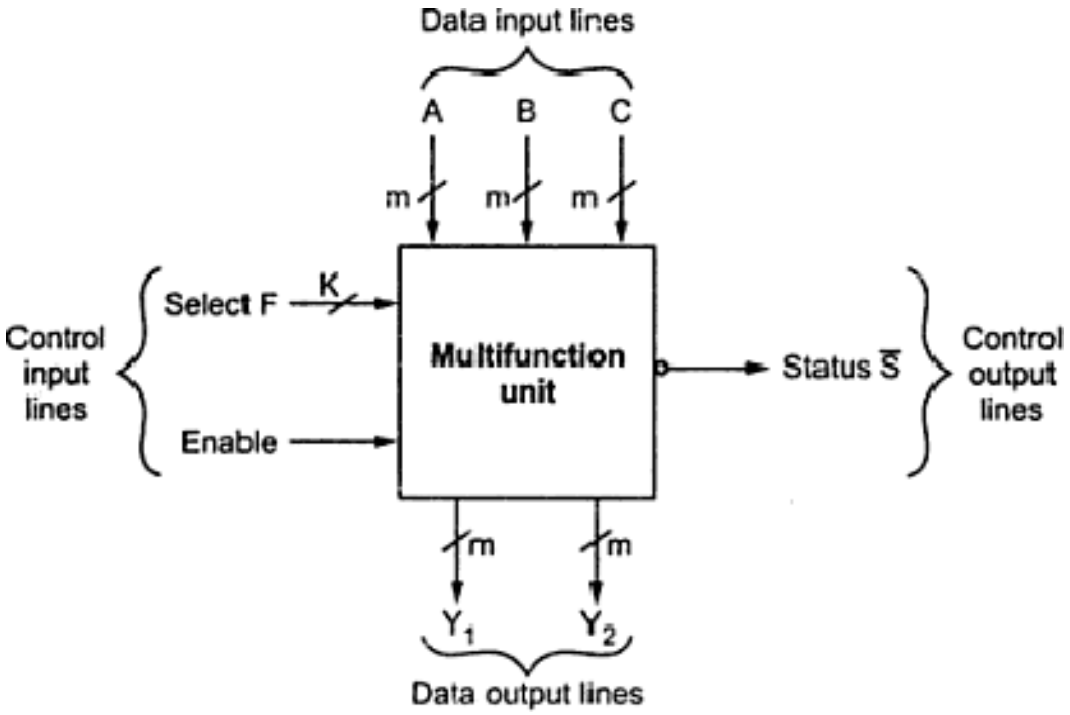
1.The problem definition.
2.The determination of number of available input variables and required output variables.
3.Assigning letter symbols to input and output variables.
4.The derivation of truth table indicating the relationships between input and output variables.
5.Obtain simplified Boolean expression for each output.
6.Obtain the logic diagram.

Register Level Design

Type	Component	Function
Combinational	Word gates.	Boolean operations.
	Multiplexers and Demultiplexers.	Data routing; general combinational functions.
	Decoders and encoders.	Code checking and conversion.
	Adders and subtractors.	Addition and subtraction.
	Arithmetic logic units.	Numerical and logical operations.
	Programmable logic devices.	General combinational functions.
Sequential	Registers	Information storage
	Shift register	Information storage; serial parallel conversion.
	Counters	Control/timing signal generation.
	Programmable logic devices.	General sequential functions.

Commonly used register level components

Register Level Design



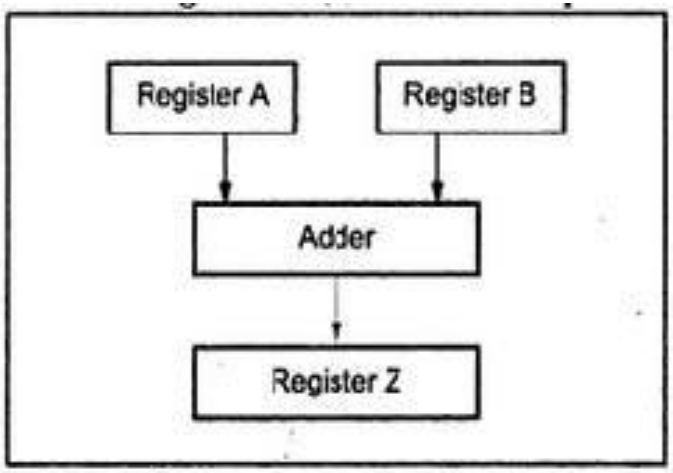
Generic block representation of a register level component

Register Level Design

- The Fig. above shows the generic block representation of a register- level component.
  - The "/m" on the input lines indicate it is a m-bit input bus.
  - A slash'/' with number or letter next to it indicates the multi-bit bus.
  - A bubble on the start or end of the line indicates an active low signal; otherwise it is an active high signal.
  - The input and output data lines are shown separately.
  - Similarly the input and output control lines are also shown separately.
  - The input control lines associated with a multifunction block fall into two broad categories: select lines and enable lines. The select lines specify one of several possible operations that the unit is to perform and enable lines specify the time or condition for a selected operation to be performed.
  - The output control signals, if any, indicate when or how the unit completes its processing.

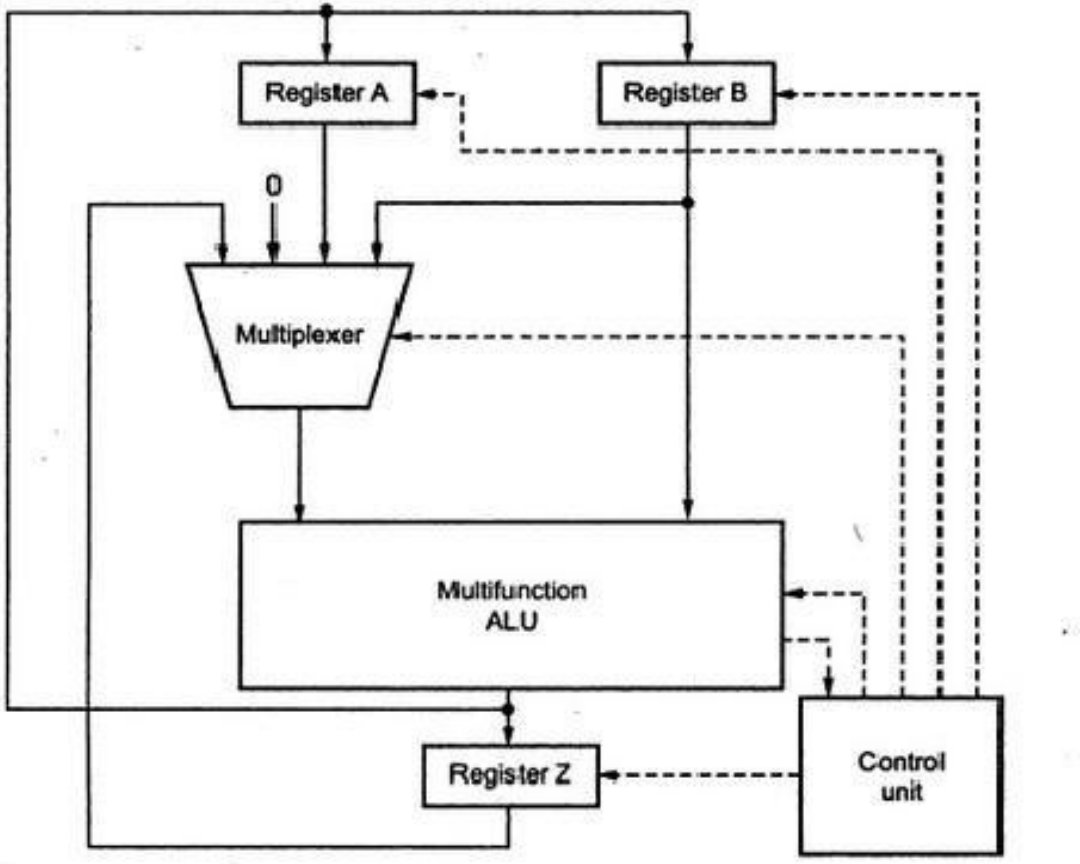
Register Level Design

Data and Control



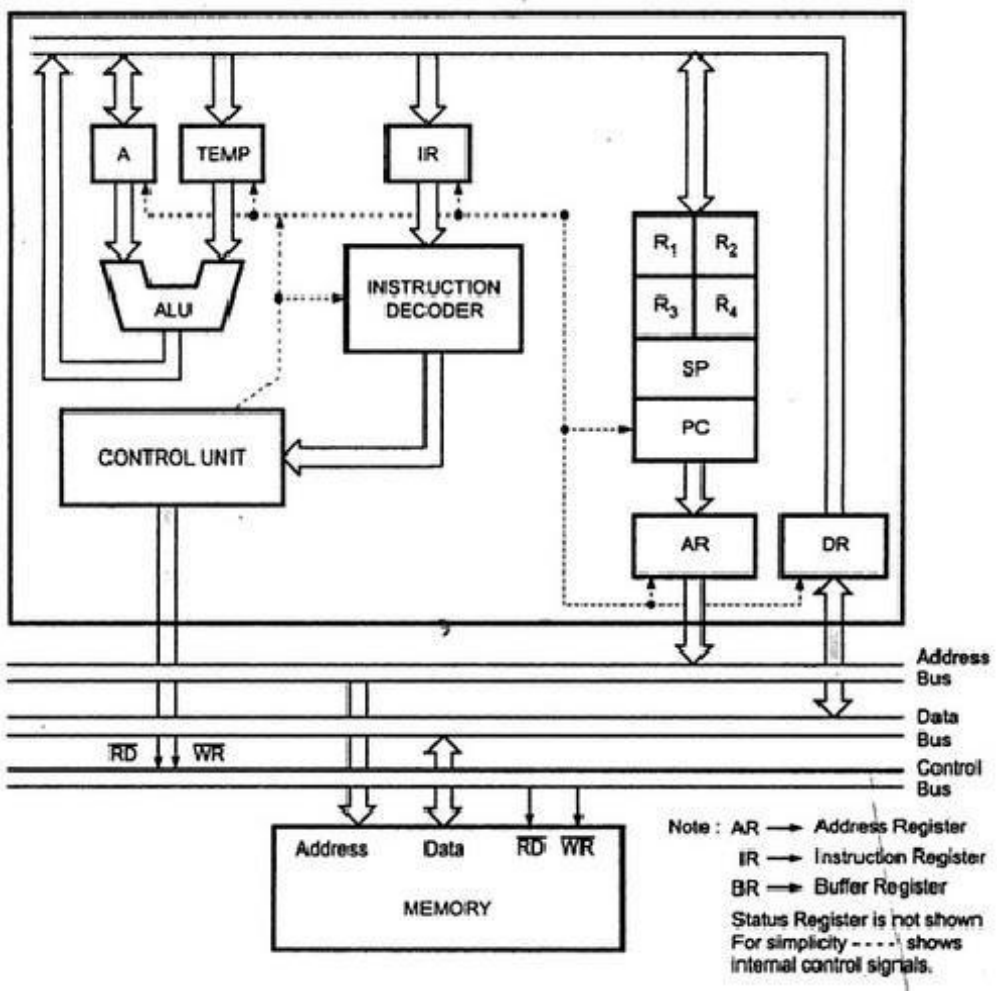
Simple register level system

Register Level Design



Multifunction register level system

Processor-Level Design



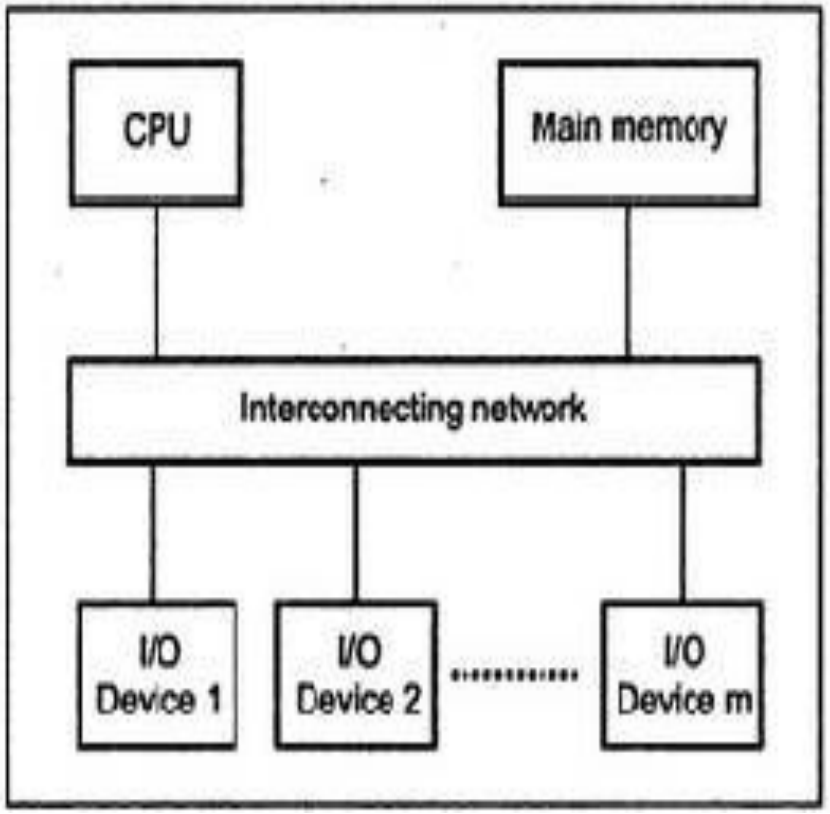
Typical CPU structure

Processor-Level Design

- **Prototype Structures**

1. The processor-level design using prototype-structures involve following steps in the design process.
2. First select a prototype design as per the system requirements and adapt it to satisfy the given performance constraints.
3. Determine the performance of proposed system.
4. If the performance is unsatisfactory, design is to be modified. Repeat step 1.
5. The above steps are to be continued until the acceptable design is obtained and the desired performance constraints are achieved.

Processor-Level Design



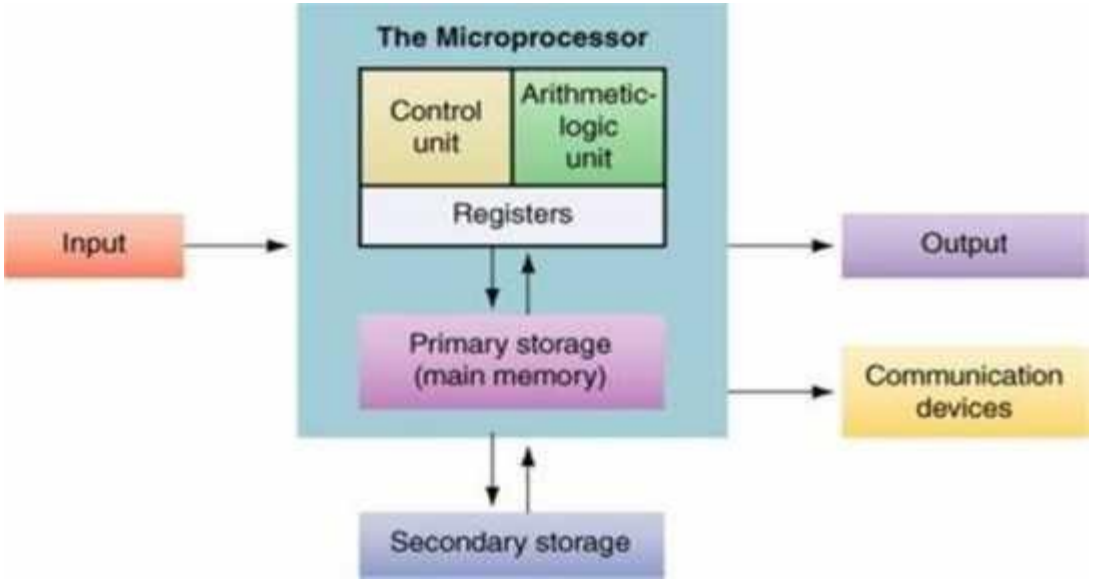
Basic computer structure

Processor-Level Design

-Figure above shows the structure of first generation computers.

-This is the basic and computer structure. The second subsequent generations of computer involve special-purpose 10 processors and cache memory in addition to basic components used within the basic system.

CPU ORGANIZATION



CPU Organization

CPU ORGANIZATION

- Central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.
- In the computer all the all the major components are connected with the help of the system bus.
- Data bus is used to shuffle data between the various components in a computer system.
- To differentiate memory locations and I/O devices the system designer assigns a unique memory address to each memory element and I/O device. When the software wants to access some particular memory location or I/O device it places the corresponding address on the address bus

CPU ORGANIZATION

- The **control bus** is an eclectic collection of signals that control how the processor communicates with the rest of the system.
- The **read** and **write** control lines control the direction of data on the data bus. When both contain logic one the CPU and memory-I/O are not communicates.
- The CPU controls the computer. It **fetches** instructions from memory, supply the address and control signals needed by the memory to access its data.

Instruction Formats

- Layout of bits in an instruction
- Includes opcode
- Includes (implicit or explicit) operand(s)
- Usually more than one instruction format in an instruction set

Instruction Length

- Affected by and affects:
  - Memory size
  - Memory organization
  - Bus structure
  - CPU complexity
  - CPU speed
- Trade off between powerful instruction repertoire and saving space

Allocation of Bits

- Number of addressing modes
- Number of operands
- Register versus memory
- Number of register sets
- Address range
- Address granularity

PDP-8 Instruction Format

Memory Reference Instructions

Opcode			D/I	Z/C	Displacement						
0		2	3	4	5						11

Input/Output Instructions

1	1	0	Device						Opcode		
0		2	3					8	9		11

Register Reference Instructions

Group 1 Microinstructions

1	1	1	0	CLA	CLL	CMA	CML	RAR	RAL	BSW	IAC
0	1	2	3	4	5	6	7	8	9	10	11

Group 2 Microinstructions

1	1	1	1	CLA	SMA	SZA	SNL	RSS	OSR	HLT	0
0	1	2	3	4	5	6	7	8	9	10	11

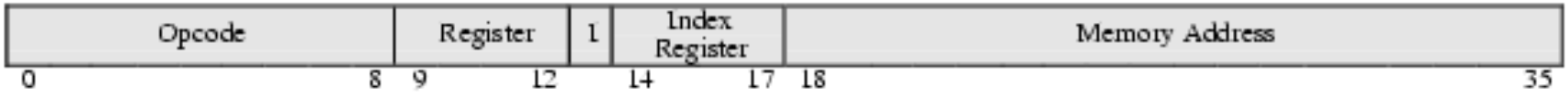
Group 3 Microinstructions

1	1	1	1	CLA	MQA	0	MQL	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11

D/I = Direct/Indirect address
Z/C = Page 0 or Current page
CLA = Clear Accumulator
CLL = Clear Link
CMA = CoMplement Accumulator
CML = CoMplement Link
RAR = Rotate Accumultator Right
RAL = Rotate Accumulator Left
BSW = Byte SWap

IAC = Increment ACcumulator
SMA = Skip on Minus Accumulator
SZA = Skip on Zero Accumulator
SNL = Skip on Nonzero Link
RSS = Reverse Skip Sense
OSR = Or with Switch Register
HLT = HaLT
MQA = Multiplier Quotient into Accumulator
MQL = Multiplier Quotient Load

PDP-10 Instruction Format



I = indirect bit

Instruction types

Assembly languages instructions are grouped together based on the operation they performed.

1. Data transfer instructions
2. Data operational instructions
3. Program control instructions

Data transfer instructions:

Data transfer instructions:

- **Load the data from memory into the microprocessor**: These instructions copy data from memory into a microprocessor register
- **Store the data from the microprocessor into the memory**: This is similar to the load data expect data is copied in the opposite direction from a microprocessor register to memory.
- **Move data within the microprocessor**: These operations copies
data from one microprocessor register to another.
- **Input the data to the microprocessor**: The microprocessor inputs the data from the input devices ex: keyboard in to one of its registers.
- **Output the data from the microprocessor**: The microprocessor copies the data from one of the registers to an input device such as digital display of a microwave oven.

Data operational instructions:

Data operational instructions:

- Data operational instructions do modify their data values. They typically perform some operations using one or two data values (operands) and store result.
- Arithmetic instructions make up a large part of data operations instructions. Instructions that add, subtract, multiply, or divide values fall into this category. An instruction that increment or decrement also falls in to this category.
- Logical instructions perform basic logical operations on data. They AND, OR, or XOR two data values or complement a single value.
- Shift operations as their name implies shift the bits of a data values also comes under this category.

Program control instructions:

Program control instructions:

- Program control instructions are used to control the flow of a program. Assembly language instructions may include subroutines like in high level language program may have subroutines, procedures, and functions.
- A jump or branch instructions are generally used to go to another part of the program or subroutine.
- A microprocessor can be designed to accept interrupts. An interrupt causes the processor to stop what is doing and start other instructions. Interrupts may be software or hardware.
- One final type of control instructions is halt instruction. This instruction causes a processor to stop executing instructions such as end of a program.

Addressing Modes

- Immediate
- Direct
- Indirect
- Register
- Register Indirect
- Displacement (Indexed)
- Stack

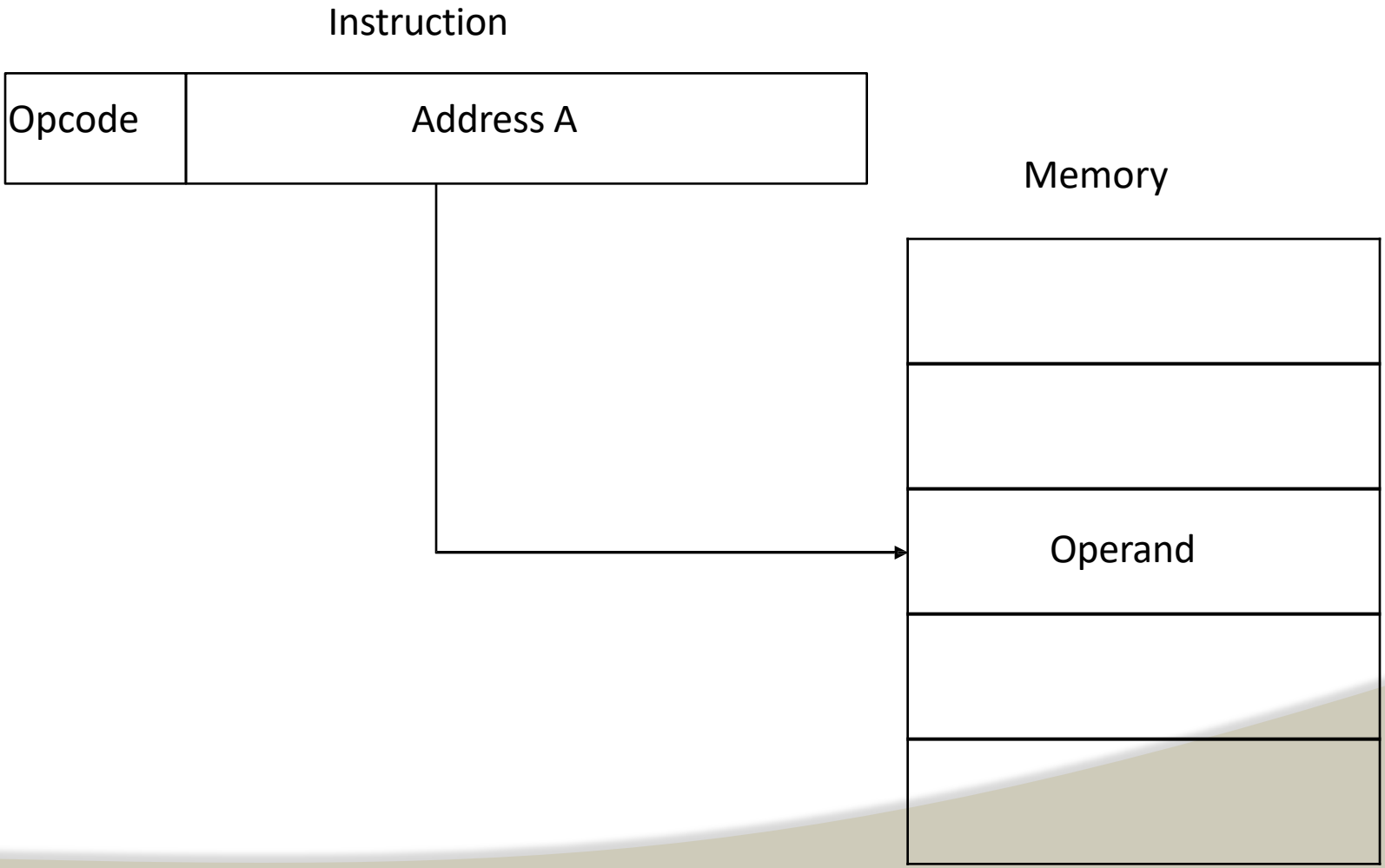
Immediate Addressing

- Operand is part of instruction
- Operand = address field
- e.g. ADD 5
  - Add 5 to contents of accumulator
  - 5 is operand
- No memory reference to fetch data
- Fast
- Limited range

Direct Addressing

- Address field contains address of operand
- Effective address (EA) = address field (A)
- e.g. ADD A
  - Add contents of cell A to accumulator
  - Look in memory at address A for operand
- Single memory reference to access data
- No additional calculations to work out effective address
- Limited address space

Direct Addressing Diagram



Indirect Addressing

- Memory cell pointed to by address field contains the address of (pointer to) the operand
- EA = (A)
  - Look in A, find address (A) and look there for operand
- e.g. ADD (A)
  - Add contents of cell pointed to by contents of A to accumulator

Indirect Addressing

- Large address space
- 2^n where n = word length
- May be nested, multilevel, cascaded
  - e.g. EA = (((A)))
    - Draw the diagram yourself
- Multiple memory accesses to find operand
- Hence slower

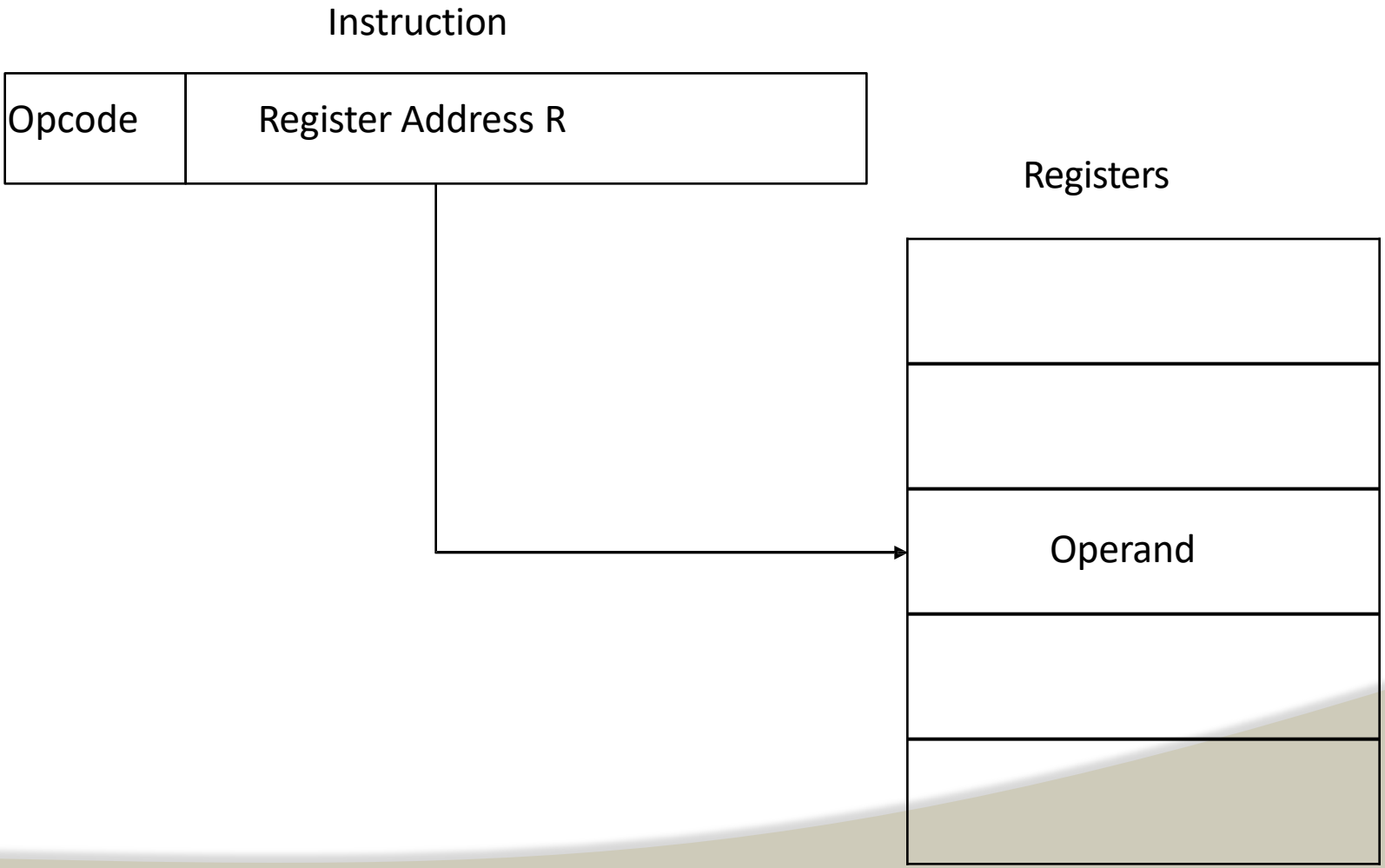
Register Addressing

- Operand is held in register named in address filed
- EA = R
- Limited number of registers
- Very small address field needed
  - Shorter instructions
  - Faster instruction fetch

Register Addressing

- No memory access
- Very fast execution
- Very limited address space
- Multiple registers helps performance
  - Requires good assembly programming or compiler writing
  - N.B. C programming
    - register int a;
- c.f. Direct addressing

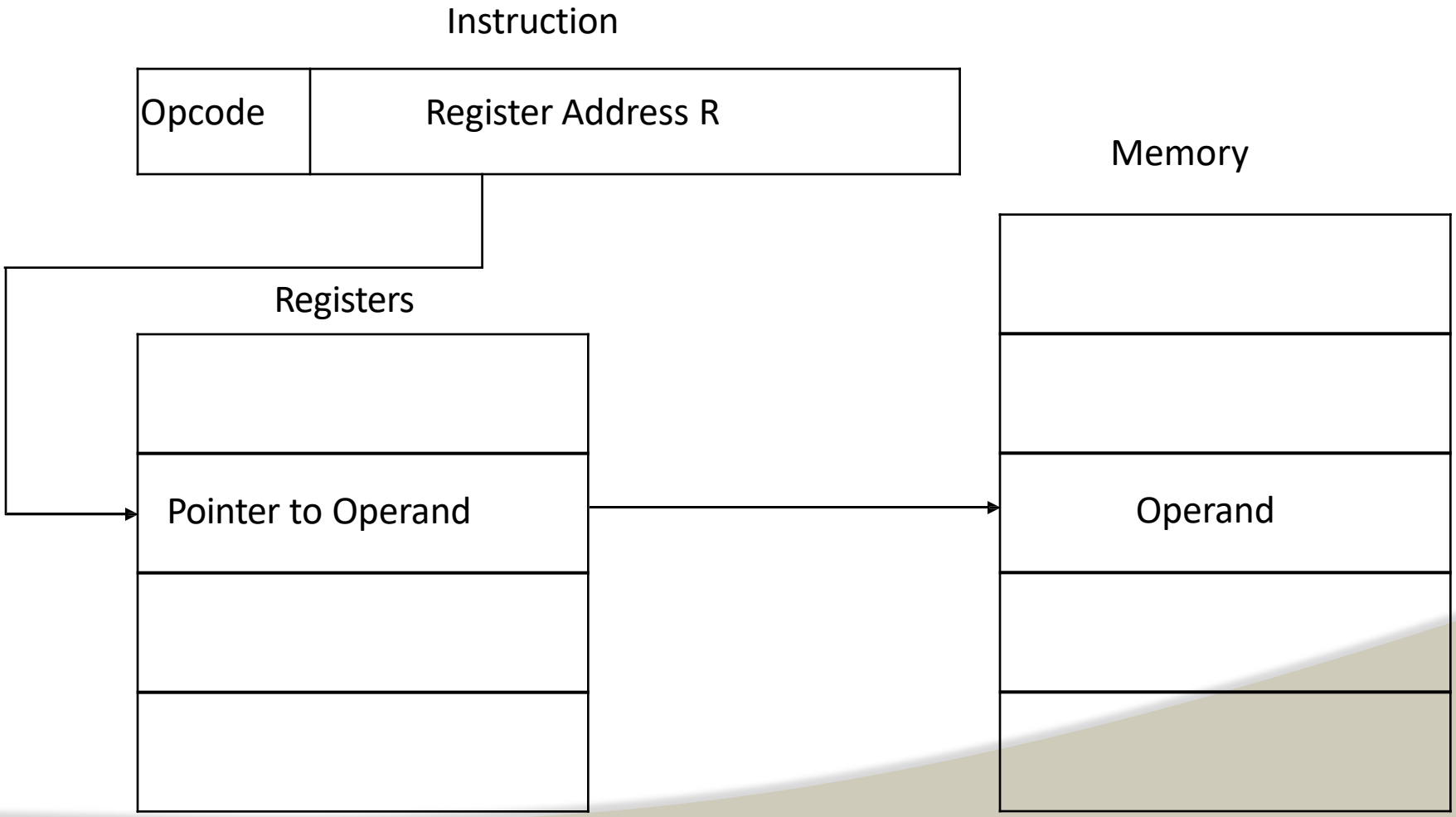
Register Addressing Diagram



Register Indirect Addressing

- C.f. indirect addressing
- EA = (R)
- Operand is in memory cell pointed to by contents of register R
- Large address space (2^n)
- One fewer memory access than indirect addressing

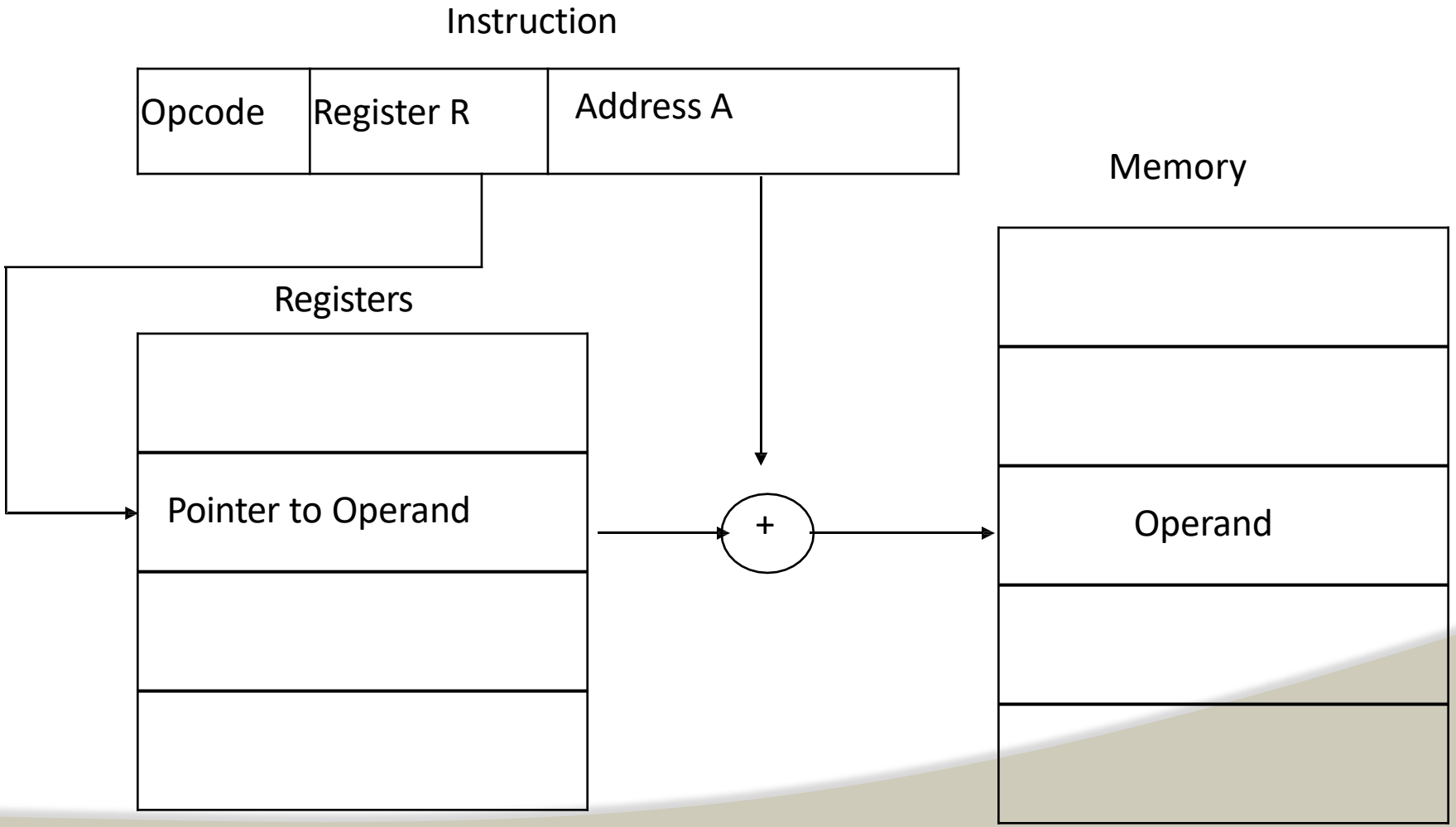
Register Indirect Addressing Diagram



Displacement Addressing

- EA = A + (R)
- Address field hold two values
  - A = base value
  - R = register that holds displacement
  - or vice versa

Displacement Addressing Diagram



Base-Register Addressing

- A holds displacement
- R holds pointer to base address
- R may be explicit or implicit
- e.g. segment registers in 80x86

Indexed Addressing

- A = base
- R = displacement
- EA = A + R
- Good for accessing arrays
  - EA = A + R
  - R++

Combinations

- Postindex
- EA = (A) + (R)

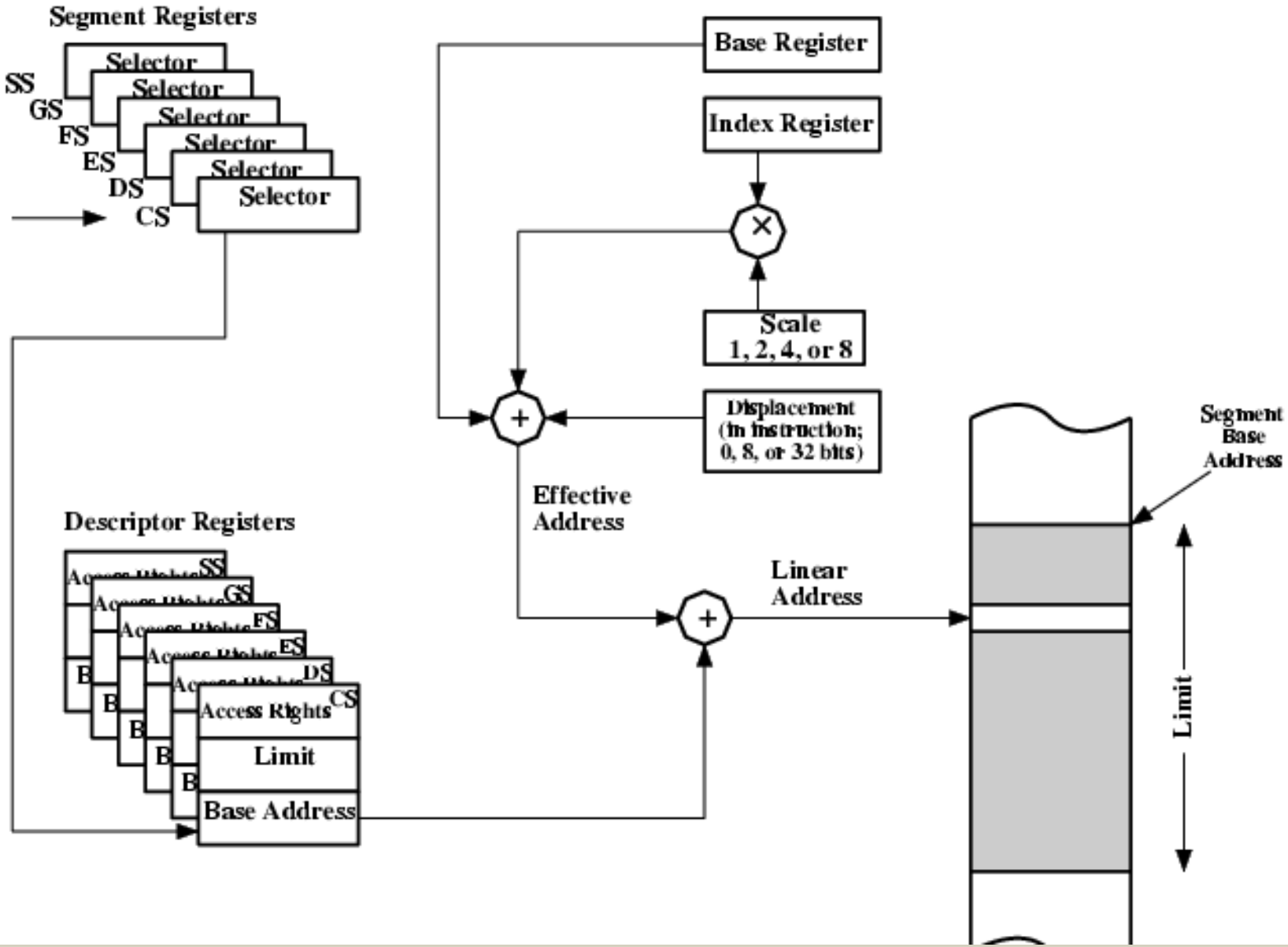
- Preindex
- EA = (A+(R))

- (Draw the diagrams)

Stack Addressing

- Operand is (implicitly) on top of stack
- e.g.
  - ADD Pop top two items from stack and add

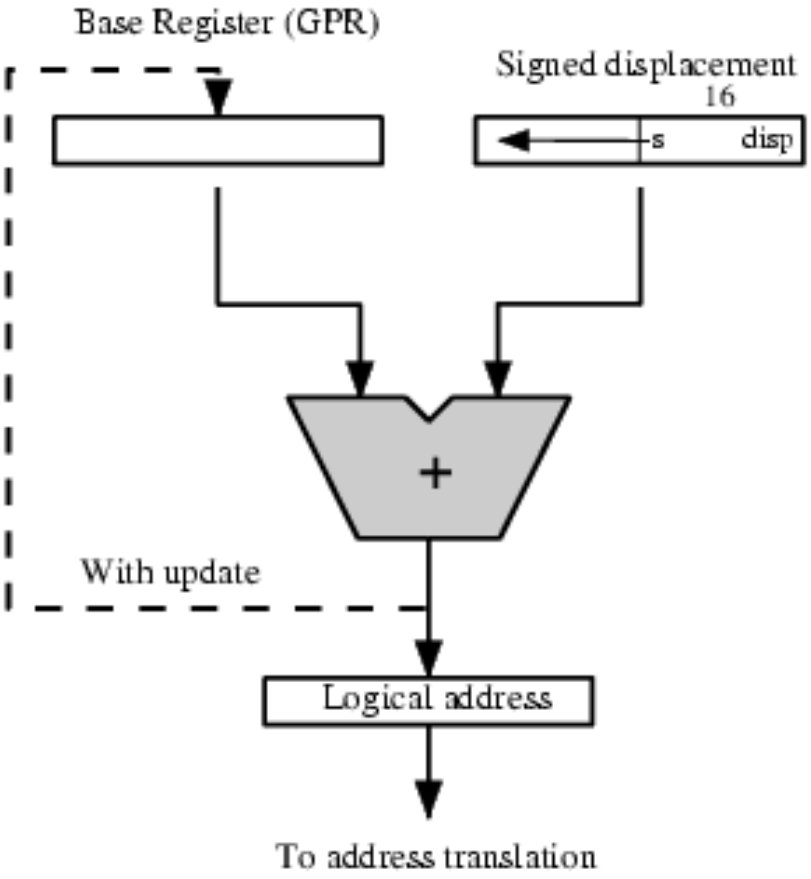
Pentium Addressing Mode Calculation



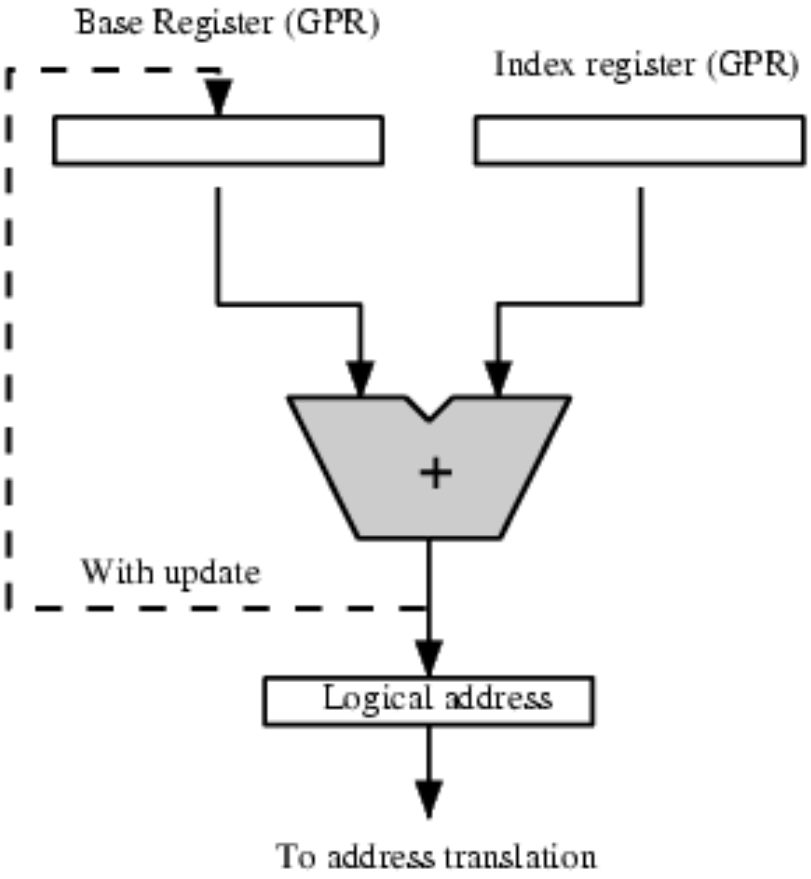
PowerPC Addressing Modes

- Load/store architecture
  - Indirect
    - Instruction includes 16 bit displacement to be added to base register (may be GP register)
    - Can replace base register content with new address
  - Indirect indexed
    - Instruction references base register and index register (both may be GP)
    - EA is sum of contents
- Branch address
  - Absolute
  - Relative
  - Indirect
- Arithmetic
  - Operands in registers or part of instruction
  - Floating point is register only

PowerPC Memory Operand



(a) Indirect Adressing



(b) Indirect Indexed Addressing

UNIT II
DATA PATH DESIGN

CLO's	Course Learning outcomes
CLO1	Describe the implementation of fixed point and floating point addition, subtraction operations.
CLO2	Describe the various major algorithmic techniques (Robertson algorithm, booth's algorithm, non-restoring division algorithm).
CLO3	Describe the pipeline processing concept with multiple functional units.
CLO4	Understand the concept of the modified booth's algorithm.

Fixed point arithmetic

- Floating point (FP) representations of decimal numbers are essential to scientific computation using *scientific notation*. The standard for floating point representation is the IEEE 754 Standard.

- In a computer, there is a tradeoff between range and precision - given a fixed number of binary digits (bits), precision can vary inversely with range. In this section, we overview decimal to FP conversion, MIPS FP instructions, and how registers are used for FP computations.

Fixed point arithmetic

- We have seen that an n-bit register can represent unsigned integers in the range 0 to 2^n-1, as well as signed integers in the range -2^{n-1} to -2^{n-1}-1. However, there are very large numbers (e.g., $3.15576 \cdot 10^{23}$), very small numbers (e.g., 10^{-25}), rational numbers with repeated digits (e.g., 2/3 = 0.666666...), irrationals such as $2^{1/2}$, and transcendental numbers such as e = 2.718..., all of which need to be represented in computers for scientific computation to be supported.

Scientific Notation and FP Representation

Scientific notation has the following configuration:

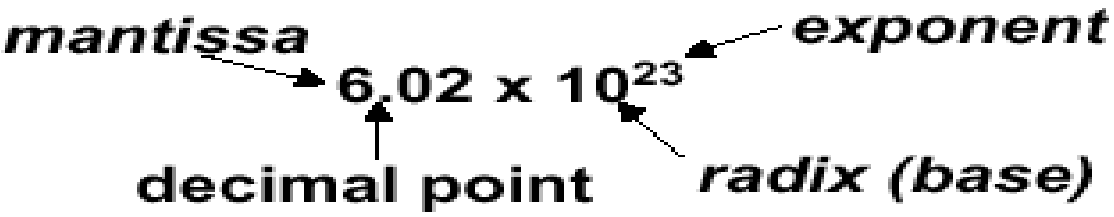


Figure – Scientific Notation

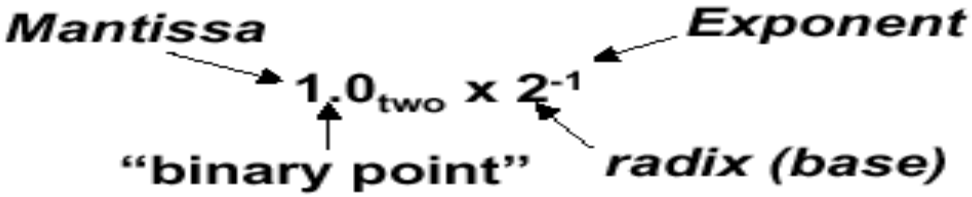


Figure – Binary Scientific Notation

Floating Point

- Zero is represented by a zero significand and a zero exponent - there is no leading value of one in the significand. The IEEE 754 representation is thus computed as:

  FPnumber = $(-1)^S \cdot (1 + \text{Significand}) \cdot 2^{\text{Exponent}}$.

- As a parenthetical note, the significand can be translated into decimal values via the following expansion:

$$1.1001 = (1x2^0) + (1x2^{-1}) + (0x2^{-2}) + (0x2^{-3}) + (1x2^{-4})$$

Floating Point

- Sign: 0 => positive
- Exponent:
  - $0110\ 1000_{two} = 104_{ten}$
  - Bias adjustment: $104 - 127 = -23$
- Significand:
  - $1 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} + 0x2^{-4} + 1x2^{-5} + \ldots$
    $= 1 + 2^{-1} + 2^{-3} + 2^{-5} + 2^{-7} + 2^{-9} + 2^{-14} + 2^{-15} + 2^{-17} + 2^{-22}$
    $= 1.0 + 0.666115$
- Represents: $1.666115 * 2^{-23} \sim 1.986 * 10^{-7}$

FP Arithmetic:

- The preceding example leads to several implementation issues in FP arithmetic. Firstly, *rounding* occurs.
- For example, when multiplying two N-bit numbers, a 2N-bit product results. Since only the upper N bits of the 2N bit product are retained, the lower N bits are *truncated*. This is also called *rounding toward zero.*
- Another type of rounding is called *rounding to infinity*.
- A second implementation issue in FP arithmetic is addition and subtraction of numbers that have nonzero significands and exponents.
- We will review several approaches to floating point operations in MIPS in the following section.

FP Arithmetic:

There are many different criteria's to check when considering the **"best"** scheduling algorithm, they are:

CPU Utilization:

•To make out the best use of CPU and not to waste any CPU cycle, CPU would be working most of the time(Ideally 100% of the time). Considering a real system, CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded.

Throughput:

•It is the total number of processes completed per unit time or rather say total amount of work done in a unit of time. This may range from 10/second to 1/hour depending on the specific processes.

Turnaround Time:

•It is the amount of time taken to execute a particular process, i.e. The interval from time of submission of the process to the time of completion of the process (Wall clock time).

FP Arithmetic:

Waiting Time:

•The sum of the periods spent waiting in the ready queue amount of time a process has been waiting in the ready queue to acquire get control on the CPU.

Load Average:

•It is the average number of processes residing in the ready queue waiting for their turn to get into the CPU.

Response Time:

•Amount of time it takes from when a request was submitted until the first response is produced. Remember, it is the time till the first response and not the completion of process execution (final response).

•In general CPU utilization and Throughput are maximized and other factors are reduced for proper optimization.

Addition, Subtraction:

- The addition and subtraction algorithm for data represented in signed magnitude and again data represented in signed-2‘s complement.
- It is important to realize that the adopted representation for negative numbers refers to the representation of numbers in the register before and after the execution of the arithmetic operations.

Addition and Subtraction with Signed-magnitude Data:

- The representation of numbers in signed-magnitude is familiar because it is used in everyday arithmetic calculation. The procedure for adding or subtracting two signed binary numbers with paper and pencils simple and straight-forward. A review of this procedure will be helpful for deriving the hardware algorithm.

Addition, Subtraction:

- We designated the magnitude of the two numbers by A and B. when the signed numbers are added or subtracted, we find that there are eight different conditions to consider, depending on the sign of the numbers and the operation performed. These conditions are listed in the first column of the table below. The other column in the table shows the actual operation to be performed with the magnitude of the numbers.
- The last column is needed to prevent negative zero. In other words, when two equal numbers are subtracted, the result should be +0 not -0.
- The algorithms for addition and subtraction are derived from the table and can be stated as follows (the words inside parentheses should be used for the subtraction algorithm).

Addition, Subtraction:

- Addition (subtraction) algorithm: when the signs of A and B are identical (different), add the two magnitude and attach the sign of A to the result.
- When the sign of A and B are different (identical), compare the magnitudes.

Addition, Subtraction:

Operation	Add Magnitudes	Subtract Magnitudes		
		When $A > B$	When $A < B$	When $A = B$
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

Figure: Table for Addition and Subtraction of Signed-Magnitude Numbers

Addition, Subtraction:

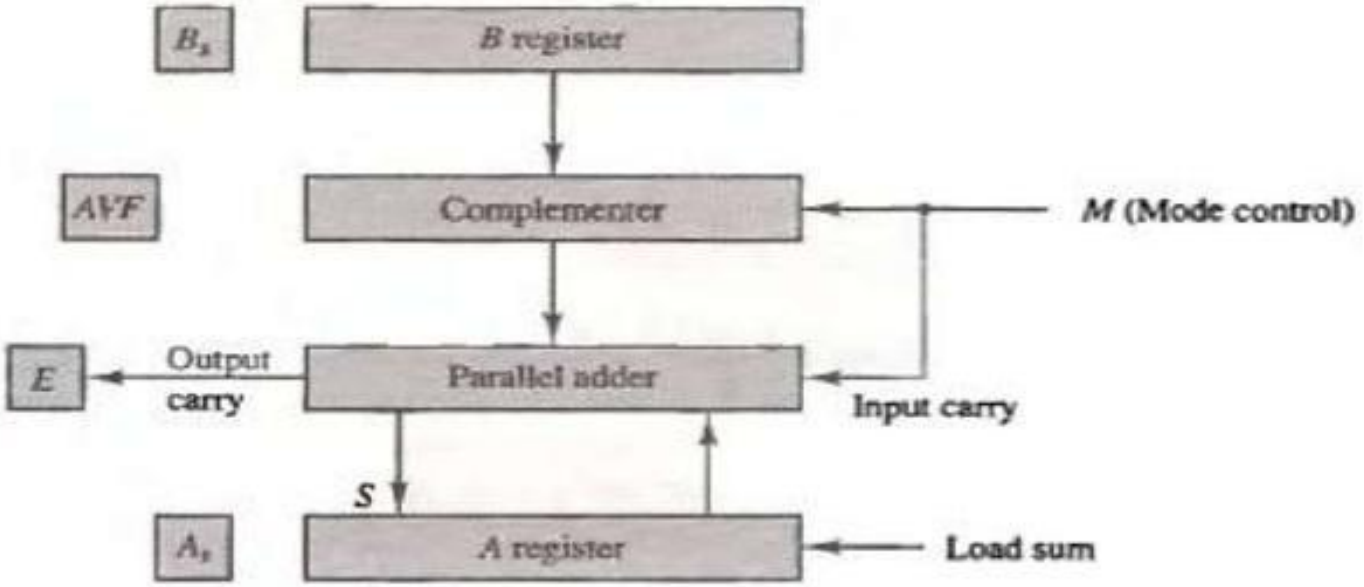
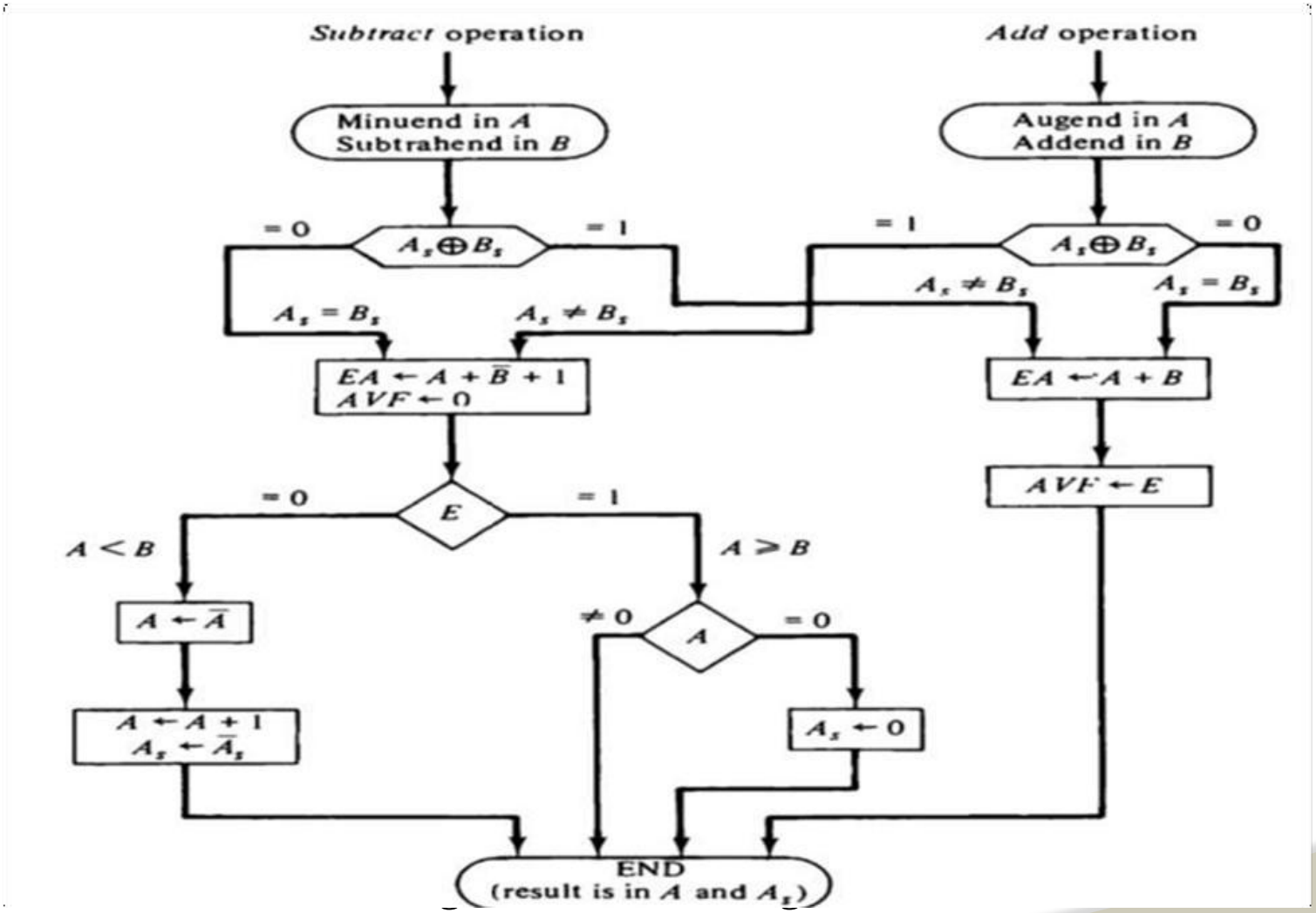


Figure: Hardware for Signed-Magnitude Addition and Subtraction.

- The output carry is transferred to flip-flop E.
- The complementary consists of exclusive-OR gates and the parallel adder consists of full adder circuit

Addition, Subtraction:



Addition, Subtraction:

- The two signs A_s and B_s are compared by an exclusive-OR gate.
- For an add operation, identical signs dictate that the magnitudes be added, for subtract operation different signs dictate that the magnitudes be added. The magnitudes are added with a micro operation E AßA+B.
- Where E A is a register that combines E and A.
- For A 0 indicates that A<B, for this case it is necessary to take the 2‘s compliment of the value in A .this operation can be done with one micro operation AßĀ+1.
- However, we assume that A register as circuits for micro operation compliment and increment, so the 2‘s compliment is obtain from these two micro operations…
- The value in AVF provides an overflow indication.
- The final value of E is immaterial.

Addition, Subtraction:

- Addition and Subtraction with signed2‘s complement data:
- The left most bit of binary number represents the sign bit; 0 for positive and 1 for negative. If the sign bit is 1, the entire the entire number is represented in 2‘s compliment form.
- The addition of two numbers in signed-2‘s complement form consists of adding the number with the sign bits treated the same as the other bits of the number. A carry out of the sign bit position is discarded.
- The subtraction consists of first taking the 2‘s compliment of the subtrahend and then adding it to the minuend
- When two numbers of n digits each are added and the sum occupies n+1 Digits, we say that an overflow occurred.
- When the two carriers are applied to an exclusive-OR gate, the overflow is detected when the output of the gate is equal to 1.

Addition, Subtraction:

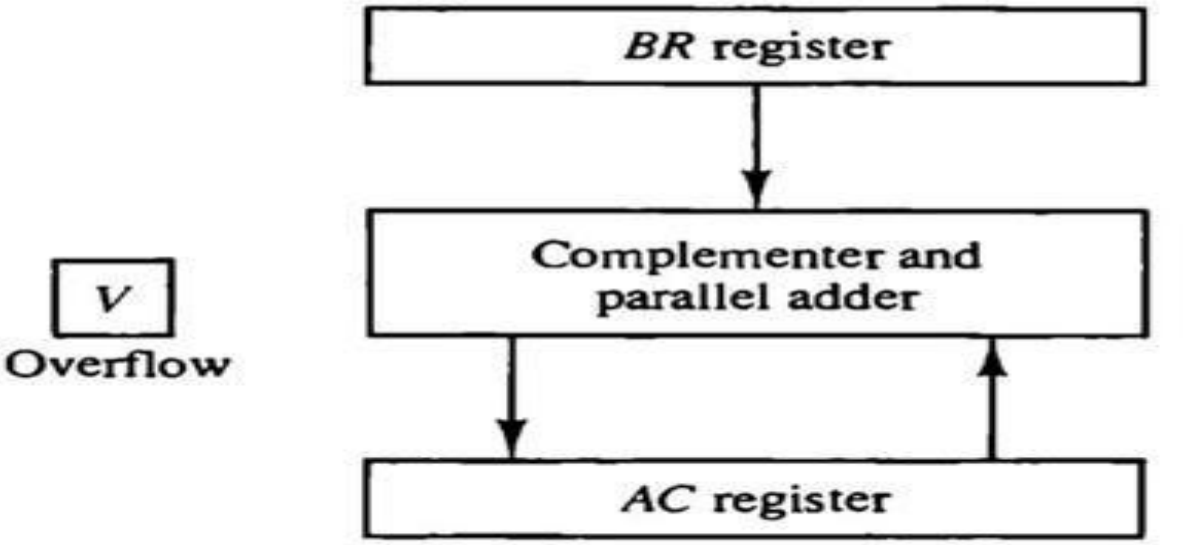
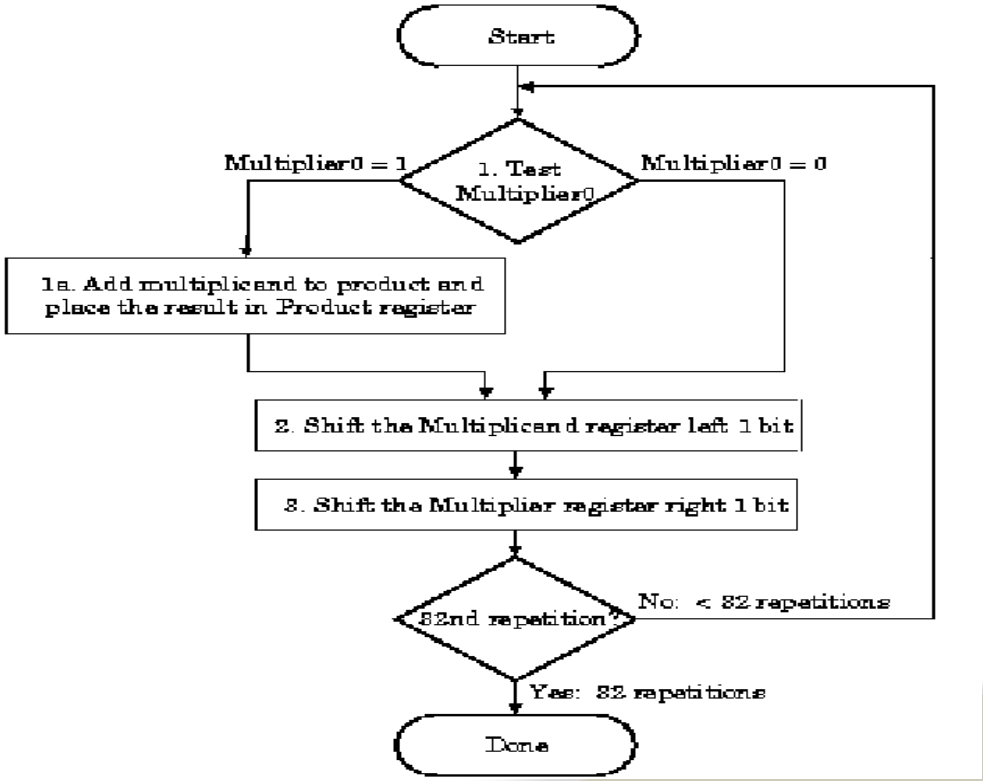


Figure:--Hardware for Signed 2's Compliant Addition and Subtraction

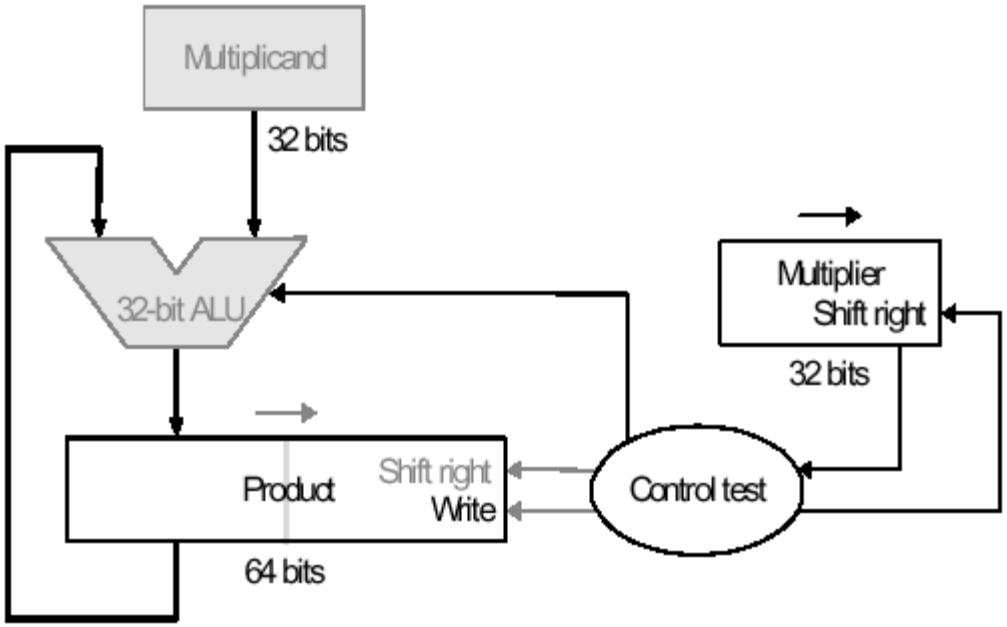
- The left most bit in AC and BR represents the sign bits of the numbers.
- The over flow flip-flops V is set to 1 if there is an overflow.
- The outputs carry in this case is discarded.

Multiplication and Division

- A flowchart of this algorithm, adapted for multiplication of 32-bit numbers, is shown in Figure below; together with a schematic representation of a simple ALU circuit that implements this version of the algorithm.
- Here, the multiplier and the multiplicand are shifted relative to each other, which is more efficient than shifting the partial products alone. Figure (a) Below:



Multiplication and Division



(b)

Figure: Second version of pencil-and-paper multiplication of 32-bit Boolean number representations: (a) algorithm, and (b) schematic diagram of ALU circuitry.

Multiplication and Division

- Thus, we have the following shift-and-add scheme for multiplication:
- The preceding algorithms and circuitry does not hold for signed multiplication, since the bits of the multiplier no longer correspond to shifts of the multiplicand. The following example is illustrative:

	Unsigned	Signed
1011	11	-5
x 1101	13	-3
10001111	143	~~-113~~

- Partial solution for negative multiplicands

1001	(9)	1001	(-7)
x 0011	(3)	x 0011	(3)
00001001	1001×2^0	11111001	$(-7) \times 2^0 = (-7)$
00010010	1001×2^1	11110010	$(-7) \times 2^1 = (-14)$
00011011	(27)	11101011	(-21)

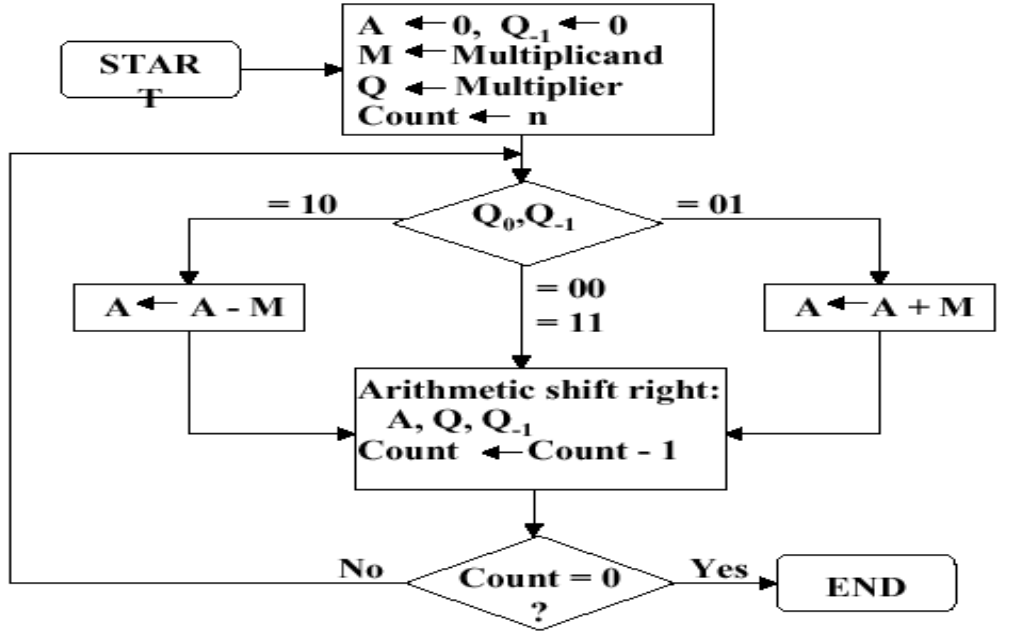
- No straightforward solution if multiplier is negative

Multiplication and Division

- A solution to this problem is Booth's Algorithm, whose flowchart and corresponding schematic hardware diagram are shown in Figure. Here, the examination of the multiplier is performed with lookahead toward the next bit. Depending on the bit configuration, the multiplicand is positively or negatively signed, and the multiplier is shifted or unshifted.

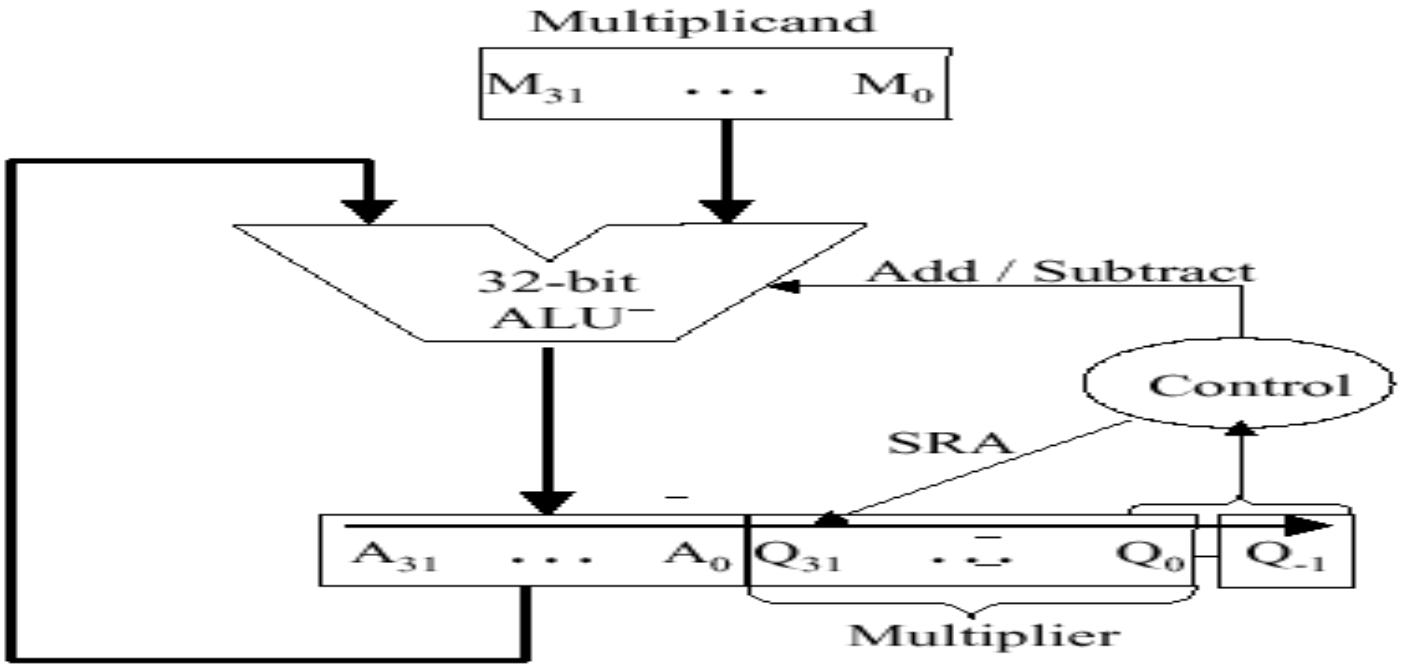
Multiplication and Division

•A solution to this problem is Booth's Algorithm, whose flowchart and corresponding schematic hardware diagram are shown in Figure. Here, the examination of the multiplier is performed with look ahead toward the next bit. Depending on the bit configuration, the multiplicand is positively or negatively signed, and the multiplier is shifted or unshifted.



(a)

Multiplication and Division



(b)

Figure. Booth's procedure for multiplication of 32-bit Boolean number representations: (a) algorithm, and (b) schematic diagram of ALU circuitry.

Multiplication and Division

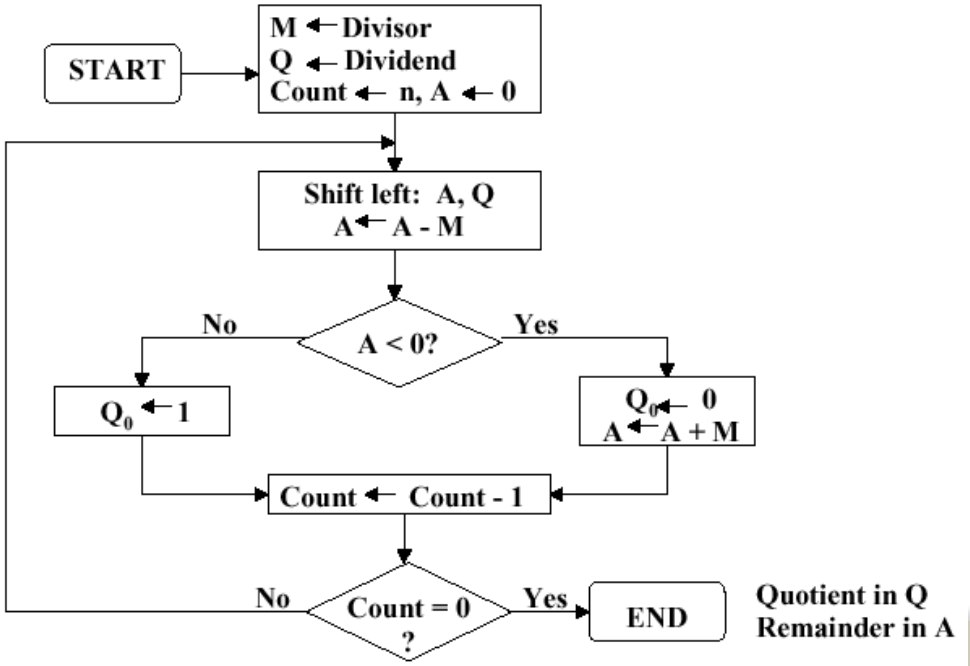
- Observe that Booth's algorithm requires only the addition of a subtraction step and the comparison operations for the two-bit codes, versus the one-bit comparison in the preceding three algorithms. An example of Booth's algorithm follows:

7 (0 1 1 1)
x 3 (0 0 1 1)

	A	Q	Q_{-1}	M		
Initial values	0000	0011	0	0111		
	1001	0011	0	0111	A = A - M	1
	1100	1001	1	0111	Shift	
	1110	0100	1	0111	Shift	2
	0101	0100	1	0111	A = A + M	3
	0010	1010	0	0111	Shift	
	0001	0101	0	0111	Shift	4

Multiplication and Division

•The ALU schematic diagram in given in Figure c. The analysis of the algorithm and circuit is very similar to the preceding discussion of Booth's algorithm.



(a)

Multiplication and Division

Unsigned Division. The unsigned division algorithm that is similar to Booth's algorithm is shown in Figure a, with an example shown in Figure b.

A	Q	M = 0011	
0000	0111	Initial values	
0000	1110	Shift	1
1101		A = A - M	
0000	1110	A = A + M	
0001	1100	Shift	2
1110		A = A - M	
0001	1100	A = A + M	
0011	1000	Shift	3
0000		A = A - M	
0000	1001	$Q_0 = 1$	
0001	0010	Shift	4
1110		A = A - M	
0001	0010	A = A + M	

(b)

Multiplication and Division

•The ALU schematic diagram in given in Figure c. The analysis of the algorithm and circuit is very similar to the preceding discussion of Booth's algorithm.

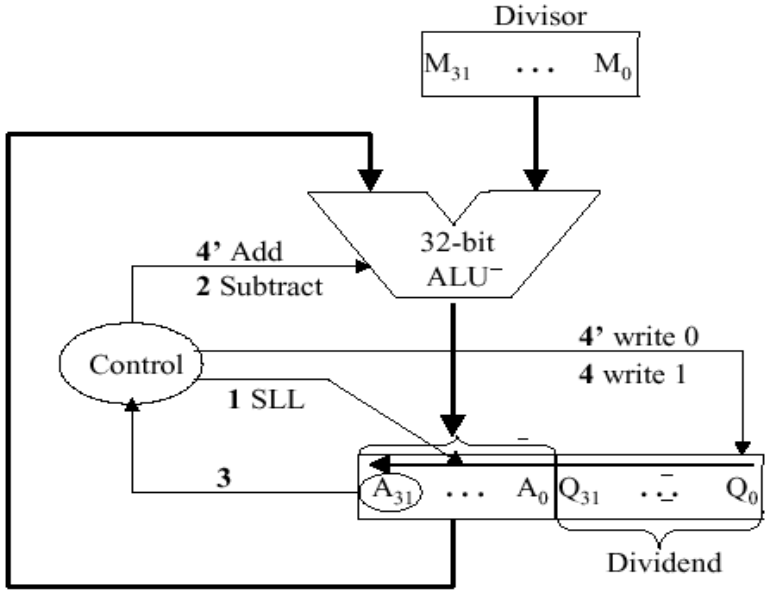


Figure . Division of 32-bit Boolean number representations: (a) algorithm, (b) example using division of the unsigned integer 7 by the unsigned integer 3, and (c) schematic diagram of ALU circuitry.

Multiplication and Division

Signed Division. With signed division, we negate the quotient if the signs of the divisor and dividend disagree. The remainder and the divident must have the same signs. The governing equation is as follows:

- Remainder = Divident - (Quotient · Divisor) ,

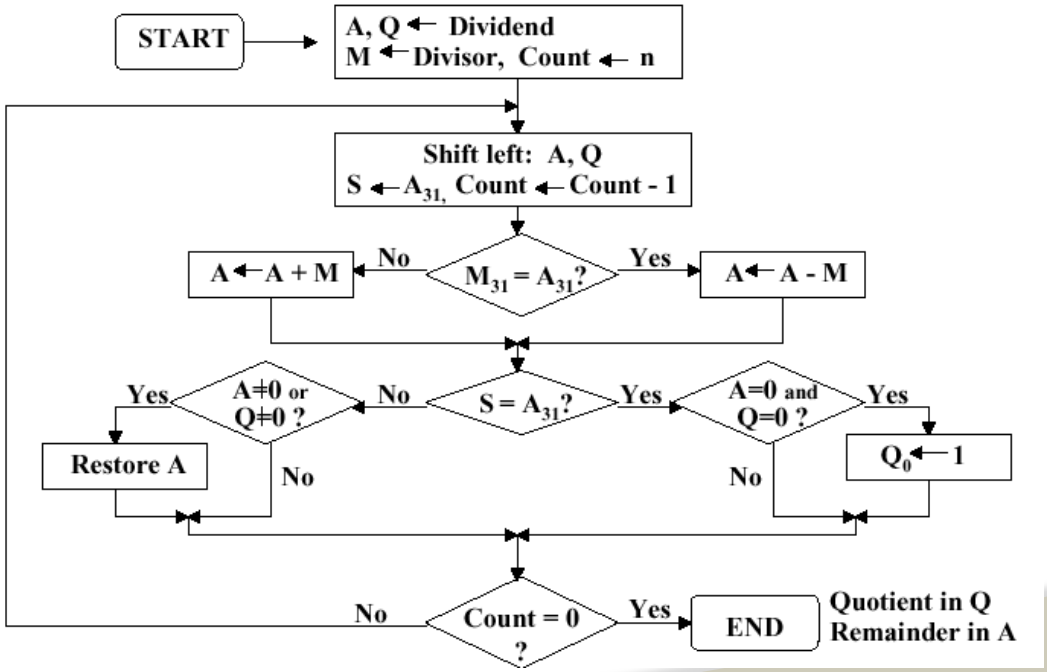
- and the following four cases apply:

(+7) / (+3): Q = 2; R = 1
(-7) / (+3): Q = -2; R = -1
(+7) / (-3): Q = -2; R = 1
(-7) / (-3): Q = 2; R = -1

- We present the preceding division algorithm, revised for signed numbers. Four examples, corresponding to each of the four preceding sign permutations, are given in Figure b and c.

Multiplication and Division

•We present the preceding division algorithm, revised for signed numbers, as shown in Figure a. Four examples, corresponding to each of the four preceding sign permutations, are given in Figure b and c.



(a)

Multiplication and Division

We present the preceding division algorithm, revised for signed numbers, as shown in Figure a. Four examples, corresponding to each of the four preceding sign permutations, are given in Figure b and c.

A	Q	M = 0011	
0000	0111	Initial values	
0000	1110	Shift	1
1101		Subtract	
0000	1110	Restore	
0001	1100	Shift	2
1110		Subtract	
0001	1100	Restore	
0011	1000	Shift	3
0000		Subtract	
0000	1001	$Q_0 = 1$	
0001	0010	Shift	4
1110		Subtract	
0001	0010	Restore	

(7) / (3)

A	Q	M = 1101	
0000	0111	Initial values	
0000	1110	Shift	1
1101		Add	
0000	1110	Restore	
0001	1100	Shift	2
1110		Add	
0001	1100	Restore	
0011	1000	Shift	3
0000		Add	
0000	1001	$Q_0 = 1$	
0001	0010	Shift	4
1110		Add	
0001	0010	Restore	

(7) / (-3)

(b)

Multiplication and Division

•We present the preceding division algorithm, revised for signed numbers, as shown in Figure a. Four examples, corresponding to each of the four preceding sign permutations, are given in

A	Q	M = 0011	
1111	1001	Initial values	
1111	0010	Shift	1
0010		Add	
1111	0010	Restore	
1110	0100	Shift	2
0001		Add	
1110	0100	Restore	
1100	1000	Shift	3
1111		Add	
1111	1001	$Q_0 = 1$	
1111	0010	Shift	4
0010		Add	
1111	0010	Restore	

(-7) / (3)

A	Q	M = 1101	
1111	1001	Initial values	
1111	0010	Shift	1
0010		Subtract	
1111	0010	Restore	
1110	0100	Shift	2
0001		Subtract	
1110	0100	Restore	
1100	1000	Shift	3
1111		Subtract	
1111	1001	$Q_0 = 1$	
1111	0010	Shift	4
0010		Subtract	
1111	0010	Restore	

(-7) / (-3)

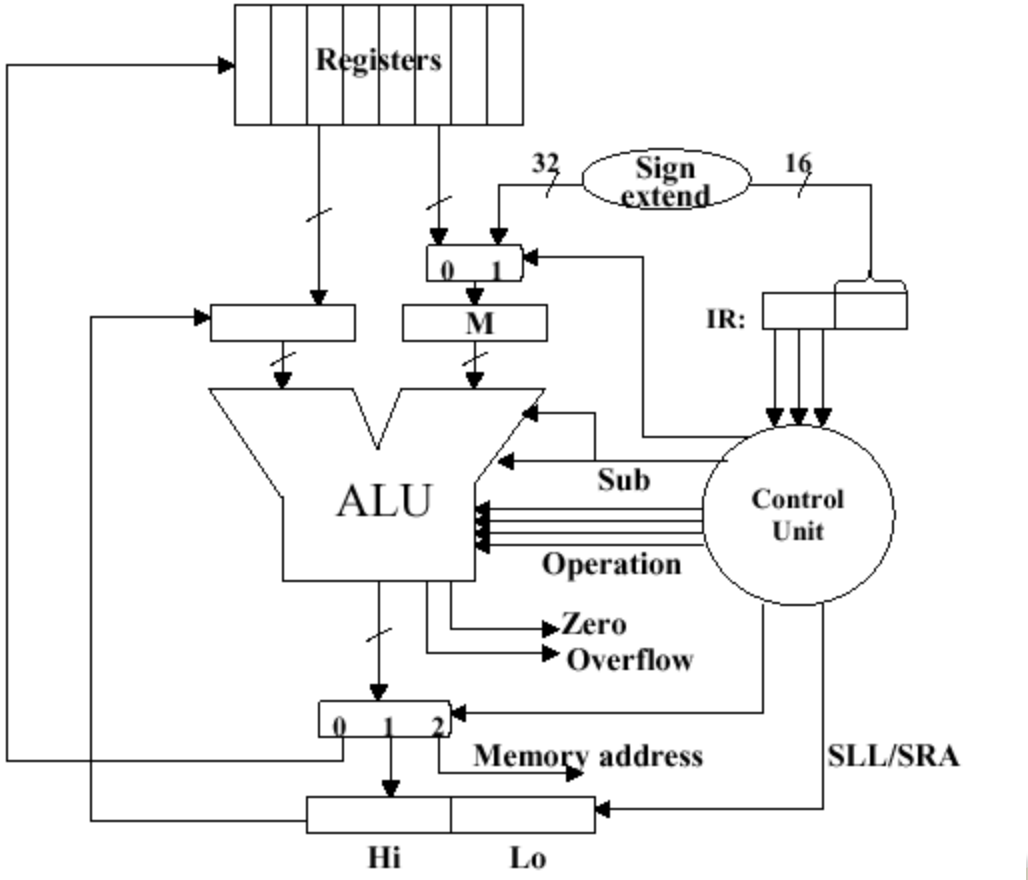
Figure b and c. Figure. Divisio integer +3 or -3;

division of +7 or -7 by the

(c)

Multiplication and Division

Figure : illustrates the MIPS ALU that supports integer arithmetic operations (+,-,x,/).

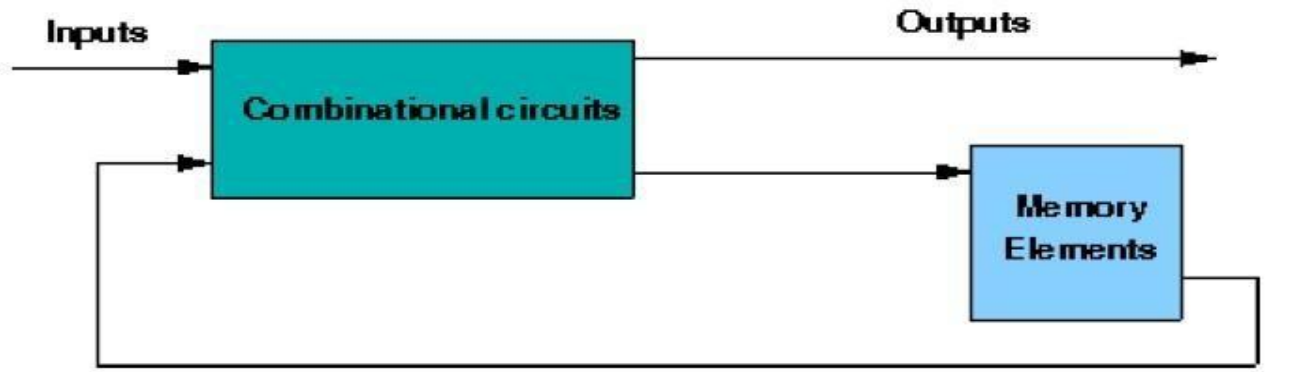


Sequential Logic Circuits:

Sequential Logic Circuits:

1. Made up of combinational circuits and memory elements.
2. These memory elements are devices capable of storing ONE-BIT information.
3. Output depends on input and previous state.
4. Examples of sequential circuits are flip flops, counters, shift registers.

Block Diagram of Sequential Circuits:



Sequential Logic Circuits:

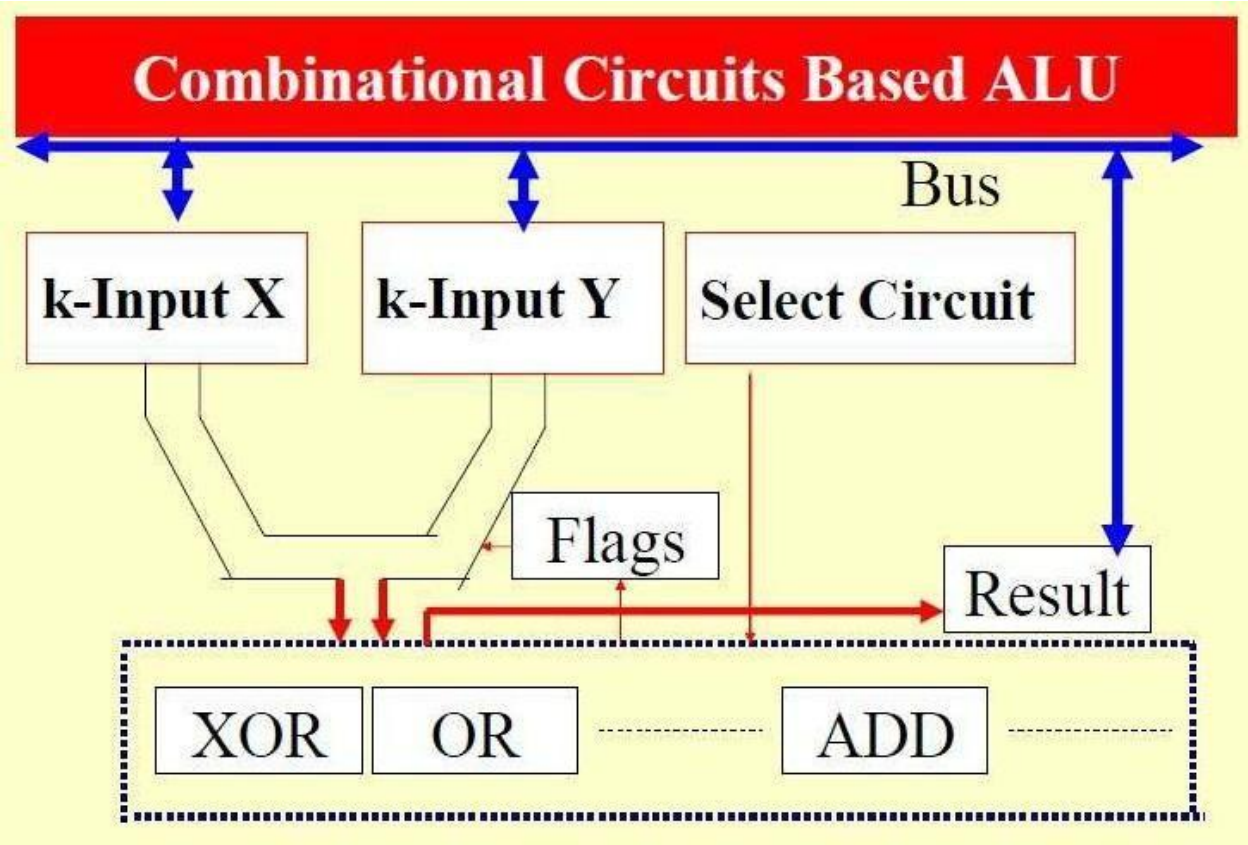
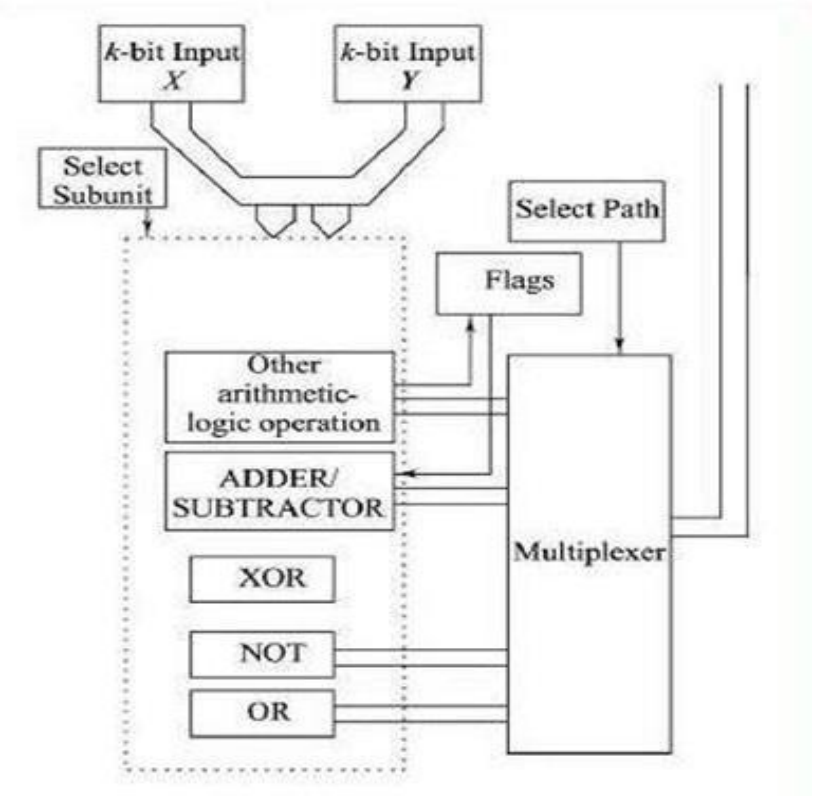


Figure: Sequential Circuits Based ALU

Sequential Logic Circuits:

An ALU using Sequential Circuits:



Sequential Logic Circuits:

Counters-

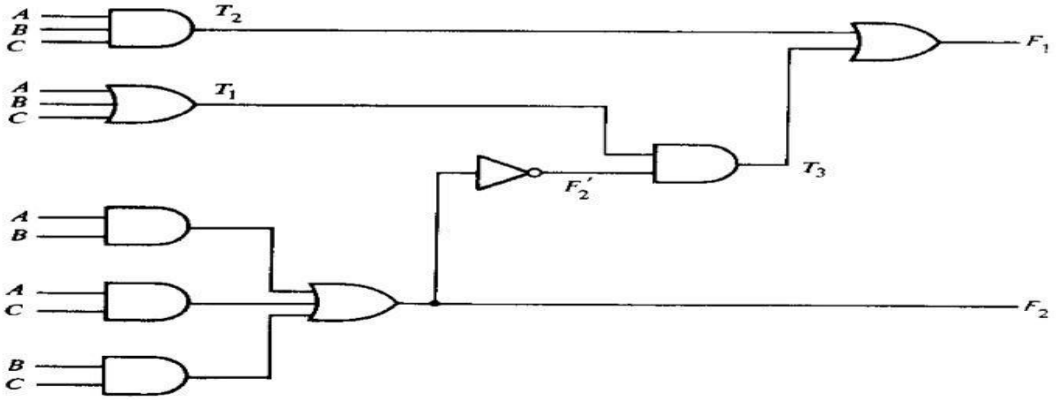
It is essentially a register that goes through a predetermined sequence of states.

- Types of Sequential Circuits:
- Sequential circuits are of two types:

1. Synchronous Sequential Circuits
2. Asynchronous Sequential Circuits

Sequential Logic Circuits:

- Logic Diagram for Analysis Example:



The circuits has 3 inputs A,B,C and 2 outputs F1, F2:

The Boolean function for outputs are:

- T1=A+B+C
- T2=ABC
- T3=F2‘T1

Outputs functions for gates are :

- F1=T3+T2
- F2=AB+AC+BC

Sequential Logic Circuits:

Substituting and Simplifying, we get:

$$F_1 = T_3 + T_2 = F_2' T_1 + ABC = (AB + AC + BC)'(A + B + C) + ABC$$

$$= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC$$

$$= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC$$

$$= A'BC' + A'B'C + AB'C' + ABC$$

Sequential Logic Circuits:

- Truth table drawn from the logic diagram:

  - Truth table:

A	B	C	T2	T1	F2	F2'	T3	F1
0	0	0	0	0	0	1	0	0
0	0	1	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1
0	1	1	0	1	1	0	0	0
1	0	0	0	1	0	1	1	1
1	0	1	0	1	1	0	0	0
1	1	0	0	1	1	0	0	0
1	1	1	1	1	1	0	0	1

Boolean functions obtained for output are:

- F2=AB+AC+BC
- F1=A‘BC‘+A‘B‘C+AB‘C‘+ABC

Sequential Logic Circuits:

A	B	C	C +1	Condition
0	0	0	0	
0	0	1	0	No Carry
0	1	0	0	Generate
0	1	1	1	
1	0	0	0	No Carry
1	0	1	1	Propogate
1	1	0	1	Carry
1	1	1	1	Generate

Figure: Sequential Circuits Based ALU

Modified Booth Algorithm Encoder

•The figure shows the functional operation of the radix-4 booth encoder that consists of eight different types of states. The outcomes or multiplication of multiplicand with 0, -1, and -2 are consecutively obtained during these eight states.

Booth recoding table for radix-4

Multiplier Bits Block			Recoded 1-bit pair		2 bit booth	
i+1	i	i-1	i+1	i	Multiplier Value	Partial Product
0	0	0	0	0	0	Mx0
0	0	1	0	1	1	Mx1
0	1	0	1	-1	1	Mx1
0	1	0	1	0	2	Mx2
1	0	0	-1	0	-2	Mx-2
1	0	1	-1	1	-1	Mx-1
1	1	0	0	-1	-1	Mx-1
1	1	0	0	0	0	Mx0

Figure - Booth Recoding Table for Radix-4

Booth's Algorithm

Booth algorithm gives a procedure for **multiplying binary integers** in signed 2's complement representation **in efficient way**, i.e., less number of additions/subtractions required. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as 2^(k+1) to 2^m.

•As in all multiplication schemes, booth algorithm requires examination **of the**

multiplier bits and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to following

Booth‘s Algorithm

Rules:

1.The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1‘s in the multiplier

2.The multiplier is added to the partial product upon encountering the first 0 (provided that there was a previous _1‘) in a string of 0‘s in the multiplier.

3.The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

Booth‘s Algorithm

Example – A numerical example of booth‘s algorithm is shown below for n =

4. It shows the step by step multiplication of -5 and -7.

- MD = -5 = 1011, MD = 1011, MD'+1 = 0101
- MR = -7 = 1001

The explanation of first step is as follows: Qn+1

- AC = 0000, MR = 1001, Qn+1 = 0, SC = 4
- Qn Qn+1 = 10

So, we do AC + (MD)'+1, which gives AC = 0101

- On right shifting AC and MR, we get
- AC = 0010, MR = 1100 and Qn+1 = 1

Booth‘s Algorithm Flowchart

Product is calculated as follows:

- Product = AC MR
- Product = 0010 0011 = 35

OPERATION	AC	MR	Qn+1	SC
	0000	1001	0	4
AC + MD‘ + 1	0101	1001	0	
ASHR	0010	1100	1	3
AC + MD	1101	1100	1	
ASHR	1110	1110	0	2
ASHR	1111	0111	0	1
AC + MD‘ + 1	0010	0011	1	0

Figure – Booth‘s Algorithm Flowchart

Division using Non-restoring Algorithm:

- Assume-- that there is an accumulator and MQ register, each of k –bits.
- MQ0, (lsb of MQ) bit gives the quotient, which is saved after a subtraction or addition.
- Total number of additions or subtractions are k -only and total number of shifts = k plus one addition for restoring remainder if needed.
- Assume –that X register has (2k − 1) bit for dividend and Y has the k -bit divisor.
- Assume – a sign-bit S shows the sign.

Division using Non-restoring Algorithm:

A has the remainder and MQ has the quotient

Step	S-flag *	First Register for *A*	Second Register for MQ	Action Taken	Number of operations (instructions)
Start	0	0b0000	0b0000	Clear S, A, MQ	3 for clearing C, A and M
	0	0b0001	0b1110	Load dividend X (lower *k* bits) in MQ_{k-1} and MQ_0 and dividend higher k–1 bits in *A*	2 for loading A and MQ
Step 0A	1	1110	1110	Subtract *Y* from S-*A*, because S = 0 result in S-*A*	1
Step 0B	1	1110	1110	$MQ_0 = 0$ as S = 1	1
Step 0C	1	1101	1100	Shift left S-A-M	2

Floating Point Arithmetic

We will continue to use the decimal number system for our numerical examples, but the impact of the computer's use of the binary number system will be felt as we discuss the way those numbers are stored in the computer.

Floating Point Formats:

- Over the years, floating point formats in computers have not exactly been

standardized. While the IEEE (Institute of Electrical and Electronics Engineers) has developed standards in this area, they have not been universally adopted.

Floating Point Arithmetic

- This is due in large part to the issue of "backwards compatibility": when a hardware manufacturer designs a new computer chip, they usually design it so that programs which ran on their old chips will continue to run in the same way on the new one.
- Since there was no standardization in floating point formats when the first floating point processing chips (often called "coprocessors" or "FPU"s: "Floating Point Units") were designed, there was no rush among computer designers to conform to the IEEE floating point standards (although the situation has improved with time).

Floating Point Arithmetic

- From the last example, it is easy to see that a 20 bit significand provides just over 6 decimal digits of precision.
- In the other examples, there is more precision than we have indicated.
- For example, a 16 bit significand is certainly sufficient to represent many decimal numbers with more than 4 digits; however, not all 5 digit decimal numbers can be represented in 16 bits, and so the precision of a 16 bit significand is said to be "> 4" (but less than 5).
- Some texts attempt to more accurately describe the precision using fractions, but we do not feel the need to do so.

Floating Point Arithmetic

- The following table describes the IEEE standard formats as well as those used in common Intel processors:

Precision	Sign	Exponent	Significand	Total Length	Decimal digits
	(# of bits)	(# of bits)	(# of bits)	(in bits)	of precision
IEEE / Intel single	1	8	23	32	> 6
IEEE single extended	1	>= 11	>= 32	>= 44	> 9
IEEE / Intel double	1	11	52	64	> 15
IEEE double extended	1	>= 15	>= 64	>= 80	> 19
Intel internal	1	15	64	80	> 19

Coprocessor

History Of Co-Processor:

Co-processor for floating point arithmetic first appeared in desktop computers in 1970s.

The coprocessors become common in 1980s and into the early 1990s.

Early 8_Bit and 16 Bit processor uses software to carry out the floating point arithmetic

operations.

Math co-processor was popular purchase for users of computer-aided design (CAD)

software and scientific and engineering calculations.

OPERATION PERFORMED BY COPROCESSOR

- Floating point arithmetic
- Graphic & Signal processing.
- String processing.
- Encryption
- Coprocessors are Unable to fetch the code from the memory so they work under the control of main processor.

Coprocessor

Architecture of 8087:

INTEL 8087

- Numeric Processor.
- Packed in 40 pin ceramic DIP package.
- Available in 5 MHz, 8MHz, 10MHz versions compatible with 8086, 8088, 80186, 80188.
- It adds 68 new instruction to the instruction set of 8086.

Architecture of 8087

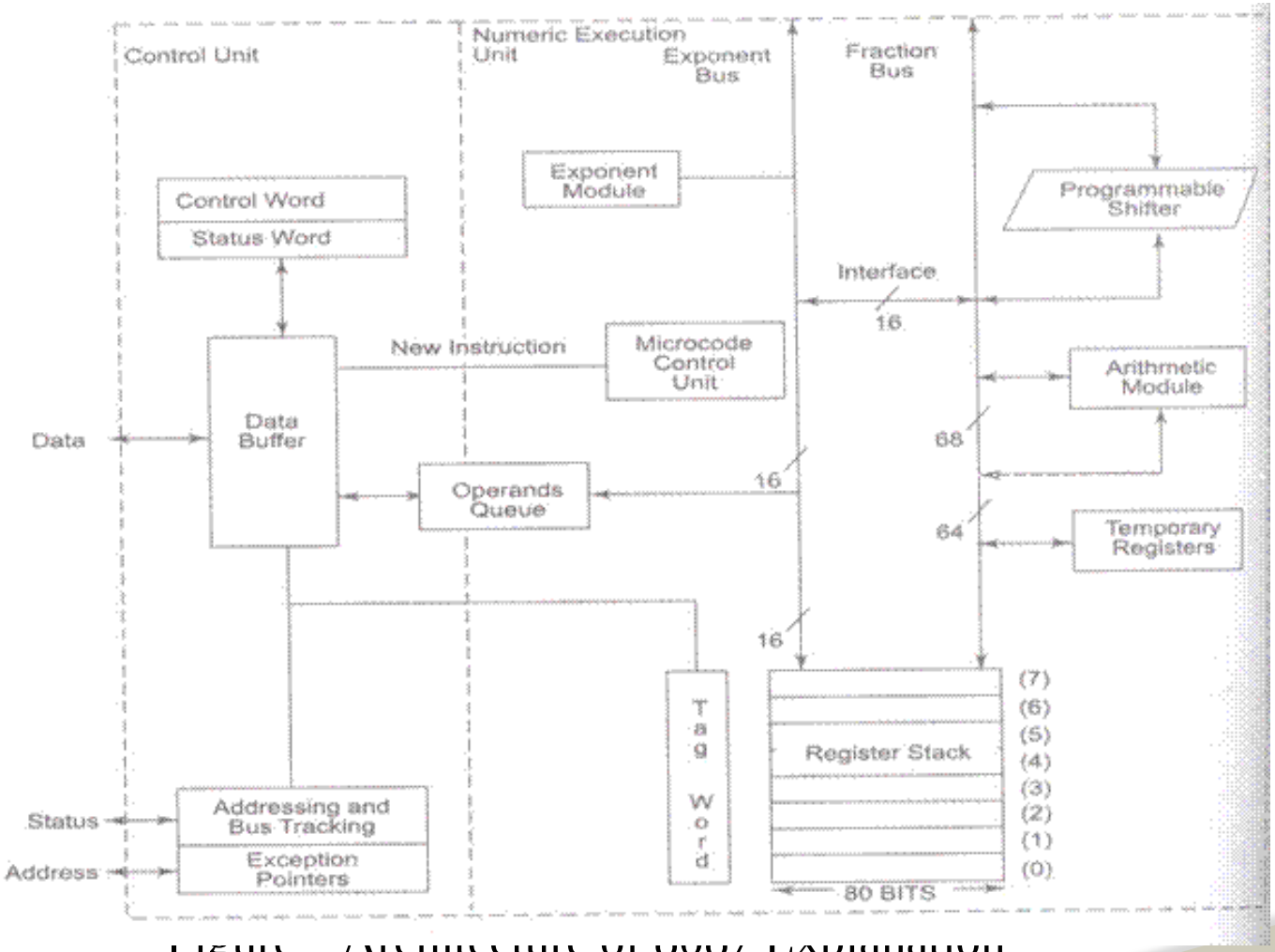


Figure - Architecture of 8087 Explanation

Architecture of 8087

Two major sections:

1)Control unit

2)Numeric Execution unit

Control Unit

- Function:
- It interfaces the coprocessor to the microprocessor – system data bus.
- Monitors the instruction stream. If the instruction is an Escape (coprocessor) instruction, the coprocessor executes it; if not the microprocessor executes it.
- It receives, decodes instructions, read and write memory operands and executes the 8087 instruction

Architecture of 8087

Numeric Execution Unit (NEU)

Functions:

- Execute all the numeric processor instructions.
- It has 8 register (80 bit) stacks that hold the operands for arithmetic instructions & the result.
- Instruction either address data in specific stack data – register or uses push and pop mechanism to store or retrieve data.

Micro instructions

- Thus a microinstruction is in primary control-store memory, it then has the control signals generated for each microinstruction using a secondary control store memory The output word from the secondary memory is called Nano instruction.
- The µCM stores micro instructions whereas nCM stores nano instructions.
- The decoder uses Nano instructions from nCM to generate control signals.
- Thus Nano programming gives an alternative strategy to generate control signals. The process of generation of control signals using nano instructions is shown in Figure

Micro instructions

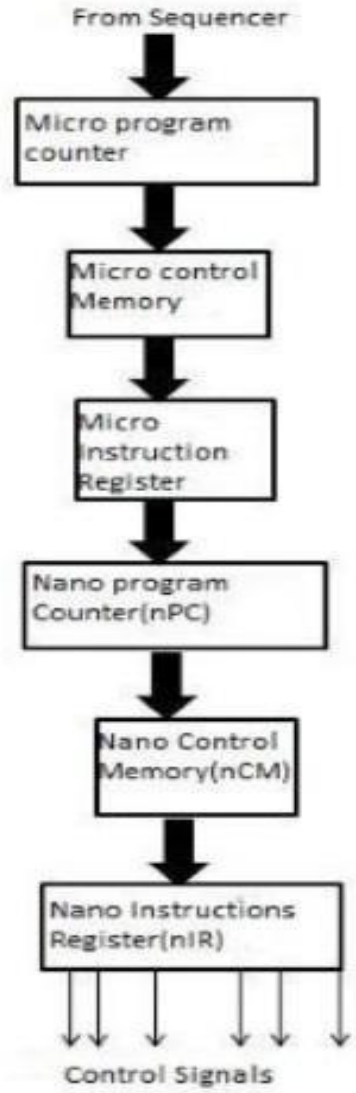


Figure - Nano Programming.

Micro instructions

Advantages of Nano programming

Reduces total size of required control memory

In two level control design technique, the total control memory size S2can be calculated as

*S*2=*HmxWm*+*HnxWn*

Where *H−mn* represents the number of words in the high level memory

Wm

represents the size of word in the high level memory

Hn represents the number of words in the low level memory

Wn represents the size of word in the low level memory

Usually, the micro programs are vertically organized so *Hm* is large and *Wm* is small. In Nano programming, we have a highly parallel horizontal organization, which makes Wn large and Hnis small. This gives the compatible

size for single level control unit as *S*1=*Hmx* Wn which is larger than *S*2. The reduced size of control memory reduces the total chip area.

Micro instructions

Greater design flexibility

Because of two level memories organization more design flexibility exists between instructions and hardware.

Disadvantage of Nano programming

1. Increased memory access time:

The main disadvantage of the two level memory approaches is the loss of speed due to the extra memory access required for Nano control memory.

Modified Booth Algorithm

•Booth multiplication algorithm consists of three major steps as shown in the structure of booth algorithm figure that includes generation of partial product called as recoding, reducing the partial product in two rows, and addition that gives final product.

•For a better understanding of modified booth algorithm & for multiplication, we must know about each block of booth algorithm for multiplication process.

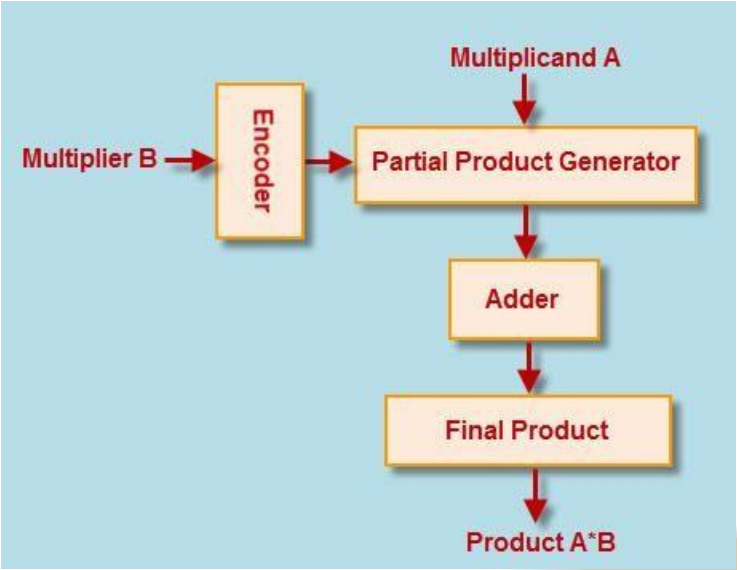


Figure - Modified Booth Algorithm.

Modified Booth Algorithm Encoder

•This modified booth multiplier is used to perform high-speed multiplications using modified booth algorithm. This modified booth multiplier‘s computation time and the logarithm of the word length of operands are proportional to each other.

•We can reduce half the number of partial product. Radix-4 booth algorithm used here increases the speed of multiplier and reduces the area of multiplier circuit. In this algorithm, every second column is taken and multiplied by 0 or

+1 or +2 or -1 or -2 instead of multiplying with 0 or 1 after shifting and adding

of every column of the booth multiplier.

•Thus, half of the partial product can be reduced using this booth algorithm. Based on the multiplier bits, the process of encoding the multiplicand is performed by radix-4 booth encoder.

Modified Booth Algorithm

•The overlapping is used for comparing three bits at a time. This grouping is started from least significant bit (LSB), in which only two bits of the booth multiplier are used by the first block and a zero is assumed as third bit as shown in the figure.

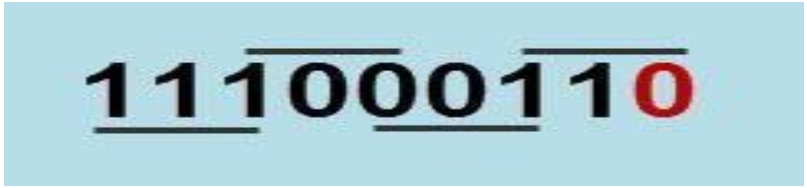


Figure - Bit Pairing as per Booth Recoding

Modified Booth Algorithm

•The figure shows the functional operation of the radix-4 booth encoder that consists of eight different types of states. The outcomes or multiplication of multiplicand with 0, -1, and -2 are consecutively obtained during these eight states.

Booth recoding table for radix-4

Multiplier Bits Block			Recoded 1-bit pair		2 bit booth	
i+1	i	i-1	i+1	i	Multiplier Value	Partial Product
0	0	0	0	0	0	Mx0
0	0	1	0	1	1	Mx1
0	1	0	1	-1	1	Mx1
0	1	0	1	0	2	Mx2
1	0	0	-1	0	-2	Mx-2
1	0	1	-1	1	-1	Mx-1
1	1	0	0	-1	-1	Mx-1
1	1	0	0	0	0	Mx0

Figure - Booth Recoding Table for Radix-4

Modified Booth Algorithm

- The steps given below represent the radix-4 booth algorithm:
- Extend the sign bit 1 position if necessary to ensure that n is even.
- Append a 0 to the right of the least significant bit of the booth multiplier.
- According to the value of each vector, each partial product will be 0, +y, -y, +2y or -2y.

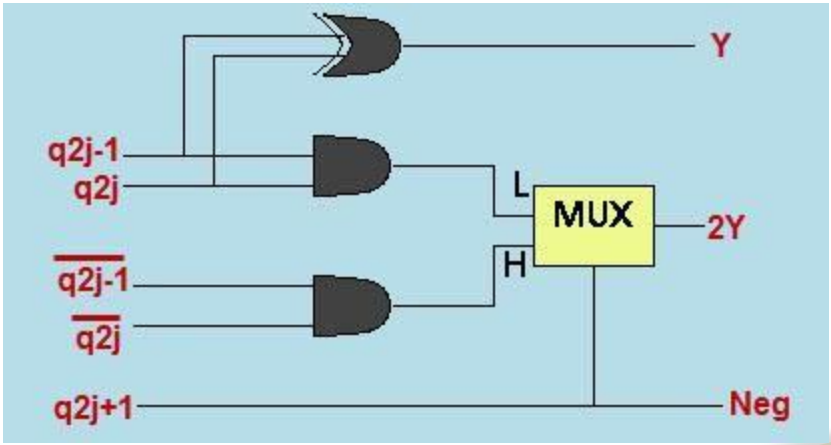


Figure - Booth's Encoder.

Modified Booth Algorithm

- Modified booth multiplier‘s (Z) digits can be defined with the following equation:
- The figure shows the modified booth algorithm encoder circuit.Now, the product of any digit of Z with multiplicand Y may be -2y, -y, 0, y, 2y. But, by performing left shift operation at partial products generation stage, 2y may be generated. By taking 1‘s complement to this 2y, negation is done, and then one is added in appropriate 4-2 compressor. One booth encoder shown in the figure generates three output signals by taking three consecutive bit inputs so as to represent all five possibilities -2X, -X, 0, X, 2X.

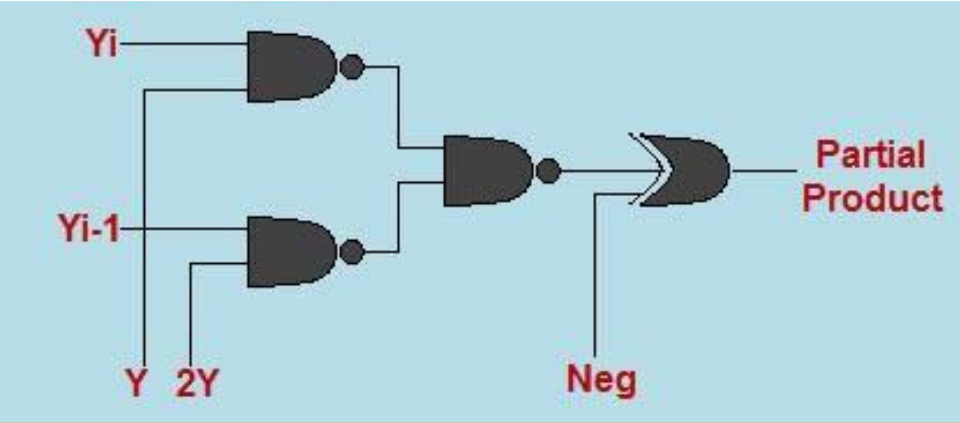


Figure - Partial Product Generator.

Modified Booth Algorithm

•Hence, to design n-bit parallel multipliers only n2 partial products are generated by using booth algorithm. Thus, the propagation delay to run circuit, complexity of the circuit, and power consumption can be reduced. A simple practical example to understand modified booth algorithm is shown in the figure below.

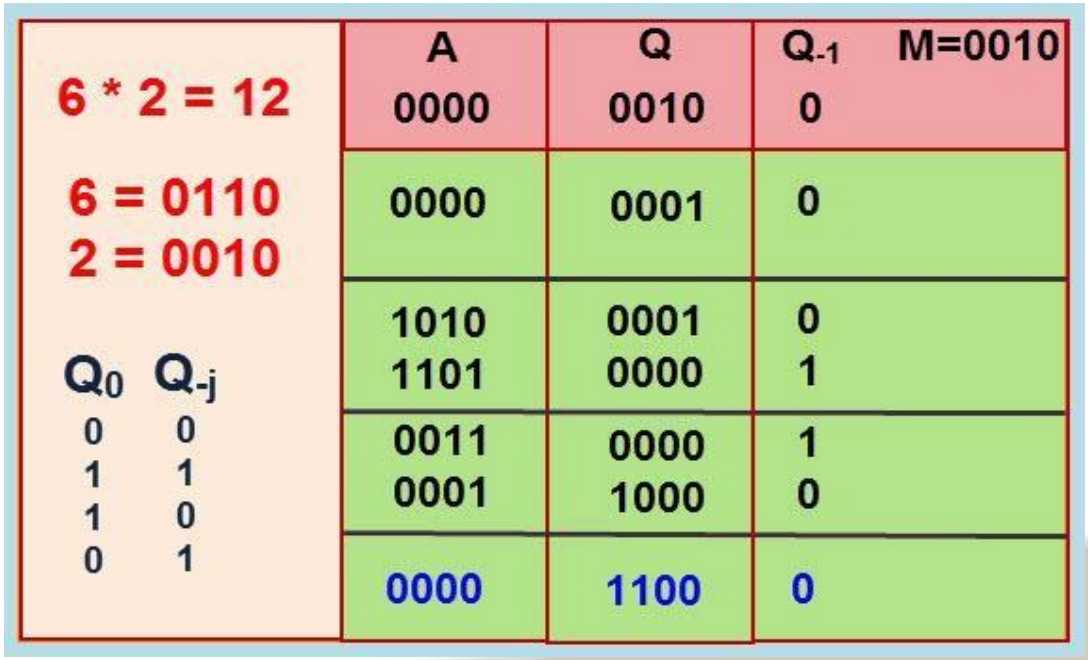


Figure -Practical Multiplication Example using Modified BoothAlgorithm

UNIT III
CONTROL DESIGN

CLO's	Course Learning outcomes
CLO1	Understand the connections among the circuits and the functionalities in the hardwired control unit.
CLO2	Describe the design of control unit with address sequencing and microprogramming Concepts.
CLO3	Describe the concepts CPU control unit, Pipeline control, instruction pipeline.
CLO4	Understand the functionality of super scalar processing and Nano programming.

Hardwired control

There are two major types of control organization:

I.Hardwired control

II.Micro programmed control

In the hardwired organization, the control logic is implemented with gates, flip-flops, decoders, and other digital circuits. It has the advantage that it can be optimized to produce a fast mode of operation. In the microprogrammed organization, the control information is stored in a control memory. The control memory is programmed to initiate the required sequence of microoperations. A hardwired control, as the name implies, requires changes in the wiring among the various components if the design has to be modified or changed. In the microprogrammed control, any required changes or modifications can be done by updating the microprogram in control memory.

Hardwired control

- The block diagram of the control unit is shown in Fig. 5-6. It consists of two decoders, a sequence counter, and a number of control logic gates. An instruction read from memory is placed in the instruction register (IR). The position of this register in the common bus system is indicated . The instruction register is shown again , where it is divided into three parts: the I bit, the operation code, and bits 0 through 1 1 . The operation code in bits 12 through 14 are decoded with a 3 x 8 decoder. The eight outputs of the decoder are designated by the symbols D0 through D7

Hardwired control

- The subscripted decimal number is equivalent to the binary value of the corresponding operation code. Bit 15 of the instruction is transferred to a flip-flop designated by the symbol I. Bits 0 through 11 are applied to the control logic gates. The 4-bit sequence counter can count in binary from 0 through 15. The outputs of the counter are decoded into 16 timing signals T0 through T15

A 2x2 binary multiplier

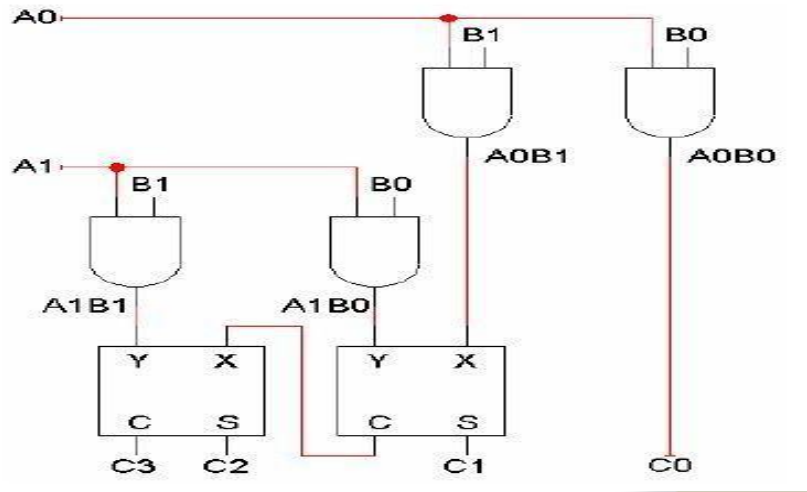
The AND gates produce the partial products.

•For a 2-bit by 2-bit multiplier, we can just use two half adders to sum the partial products

•. In general, though, we'll need full adders.

•Here C3-C0are the product, not carries!

$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & \times & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0B_0 \\
+ & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$



A 2x2 binary multiplier

- Notice that this 4-bit multiplier produces an 8-bit result.

We could just keep all 8 bits.

Or, if we needed a 4-bit result, we could ignore C4-C7, and consider it an overflow condition if the result is longer than 4bits.

- Multipliers are very complex circuits.

In general, when multiplying an m-bit number by an n-bit number:

There are n partial products, one for each bit of the multiplier.

This requires n-1 adders, each of which can add m bits (the size of the multiplicand).

The circuit for 32-bit or 64-bit multiplication would be huge!

A 2x2 binary multiplier

- In decimal, an easy way to multiply by 10 is to shift all the digits to the left, and

  tack a 0 to the right end.

  128x 10 = 1280

- We can do the same thing in binary. Shifting left is equivalent to multiplying by 2:

  11x 10 = 110(in decimal, 3 x 2 = 6)

- Shifting left twice is equivalent to multiplying by 4:

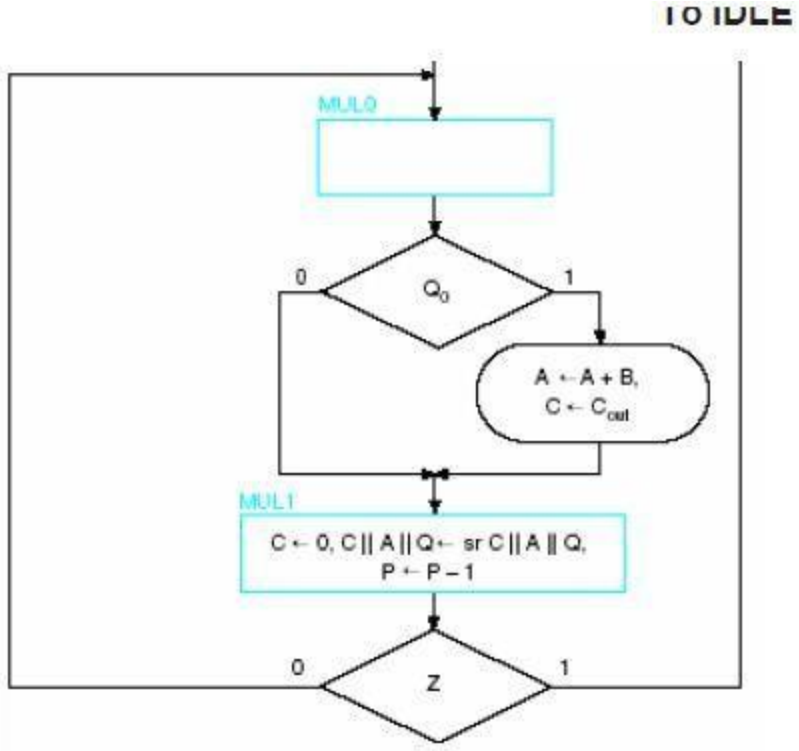
  11x 100 = 1100(in decimal, 3 x 4 = 12)

- As an aside, shifting to the right is equivalent to dividing by 2.

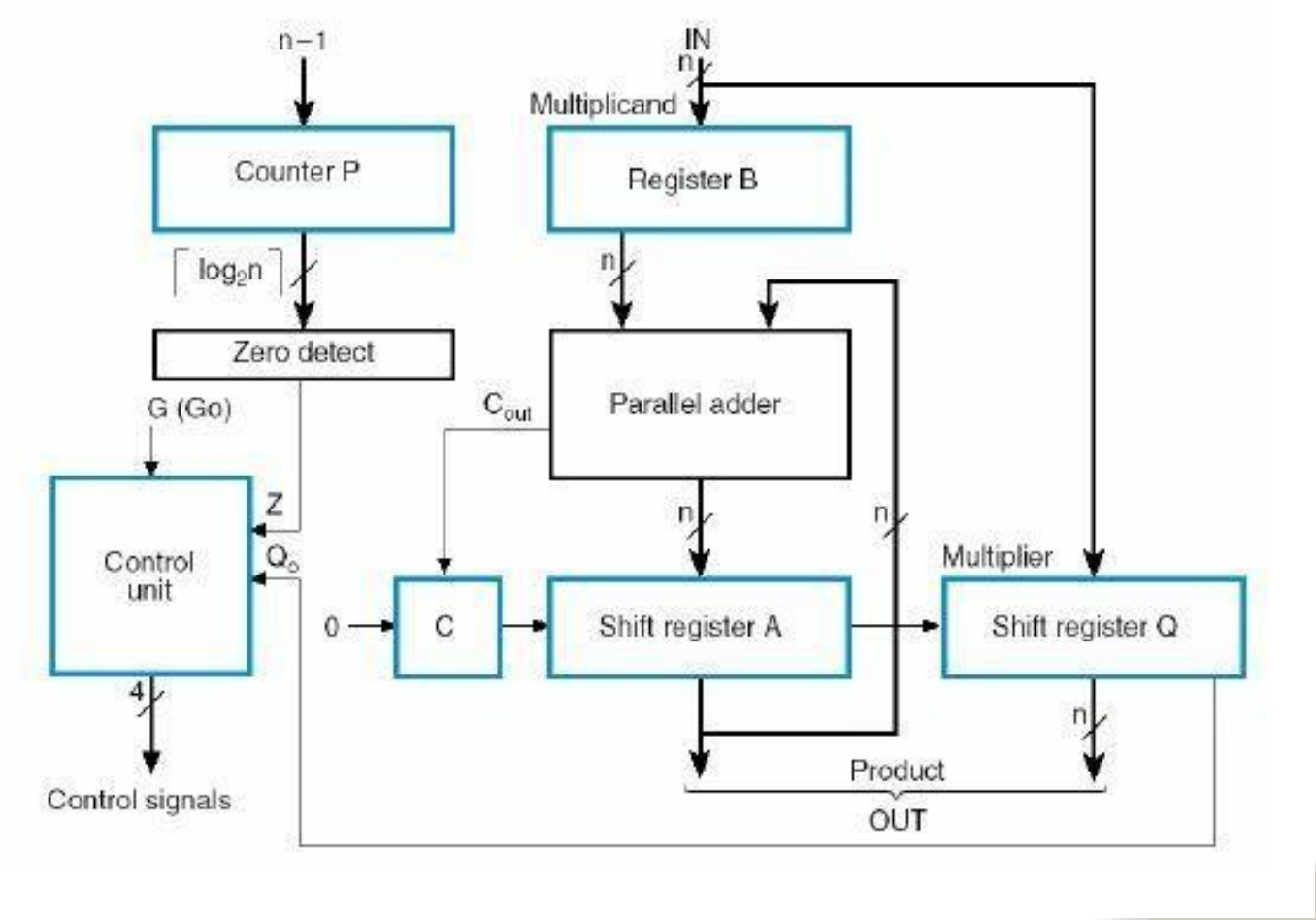
  110’10 = 11(in decimal, 6 ’2 =3)

Multiplication

- Test Q_0
  - If 1, add B
- Recall that MUL1 done all at same time
  - What happens to C?
- Test counter zero



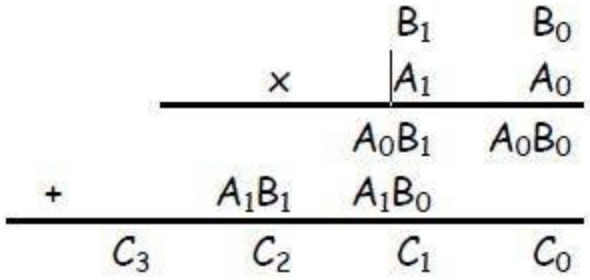
Recall: Multiplier

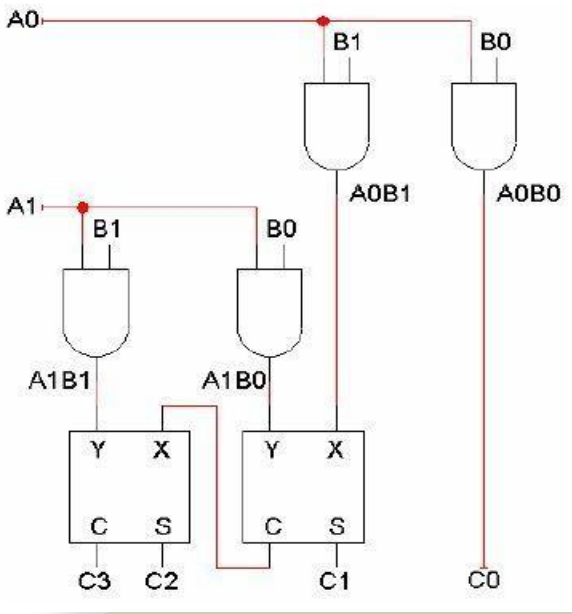


A 2x2 binary multiplier

The AND gates produce the partial products.

- For a 2-bit by 2-bit multiplier, we can just use two half adders to sum the partial products
- In general, though, we'll need full adders.
- Here C3-C0are the product, not carries!





A 2x2 binary multiplier

- Notice that this 4-bit multiplier produces an 8-bit result.

  We could just keep all 8 bits. Or, if we needed a 4-bit result, we could ignore C4-C7, and consider it an overflow condition if the result is longer than 4bits.

- Multipliers are very complex circuits.

  In general, when multiplying an m-bit number by an n-bit number:

  There are n partial products, one for each bit of the multiplier.

  This requires n-1 adders, each of which can add m bits (the size of the multiplicand).

  The circuit for 32-bit or 64-bit multiplication would be huge!

A 2x2 binary multiplier

- In decimal, an easy way to multiply by 10 is to shift all the digits to the left, and tack a 0 to the right end.

  128x 10 = 1280

- We can do the same thing in binary. Shifting left is equivalent to multiplying by 2:

  11x 10 = 110(in decimal, 3 x 2 = 6)

- Shifting left twice is equivalent to multiplying by 4:

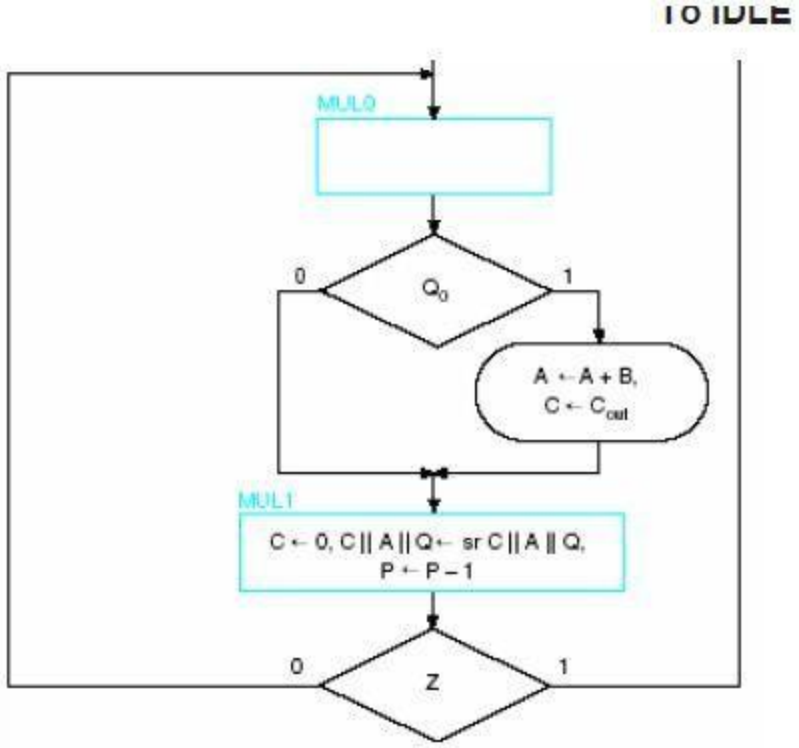
  11x 100 = 1100(in decimal, 3 x 4 = 12)

- As an aside, shifting to the right is equivalent to dividing by 2.

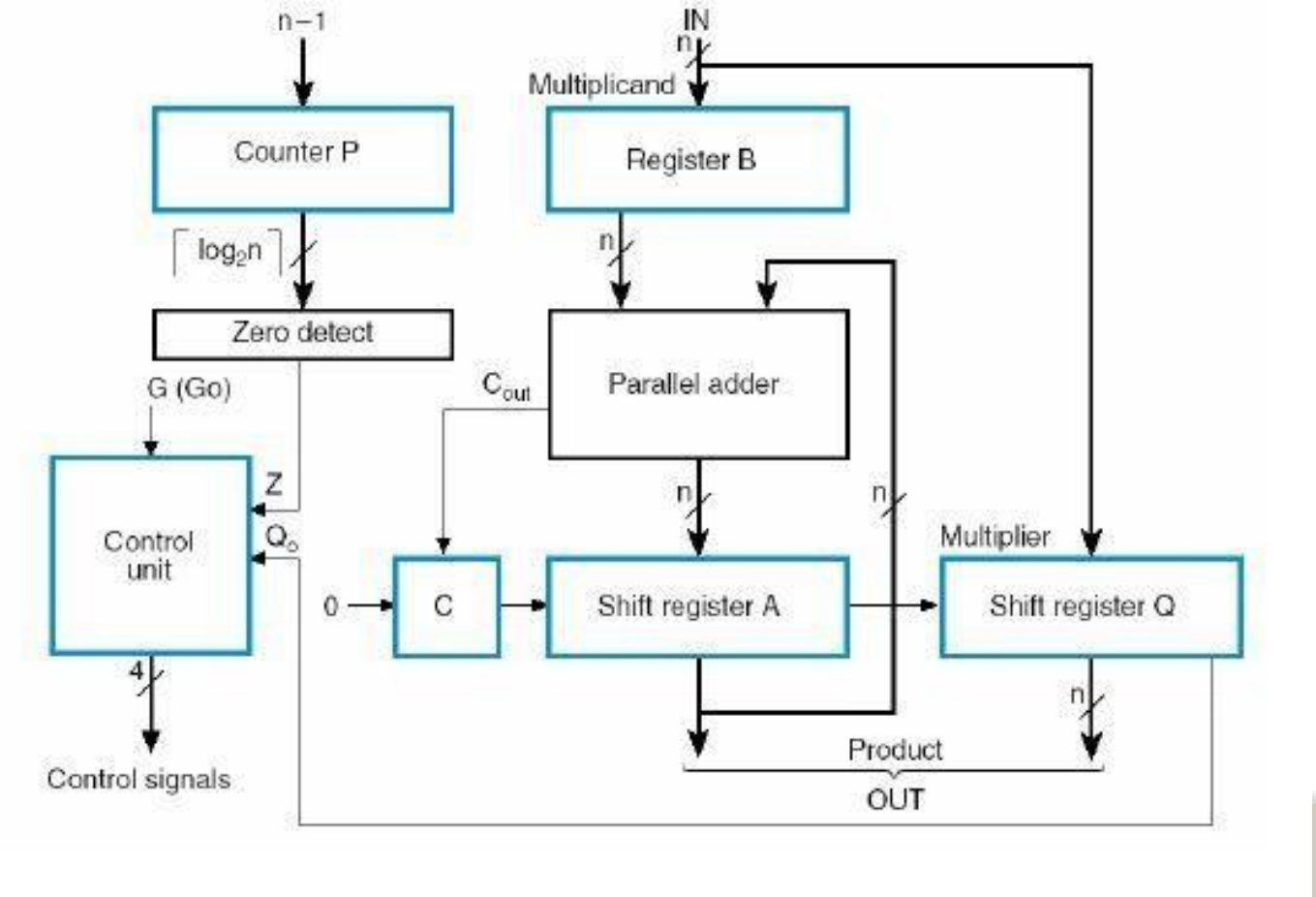
  110÷10 = 11(in decimal, 6 ÷2 = 3)

Multiplication

- Test Q_0
  - If 1, add B
- Recall that MUL1 done all at same time
  - What happens to C?
- Test counter zero



Recall: Multiplier

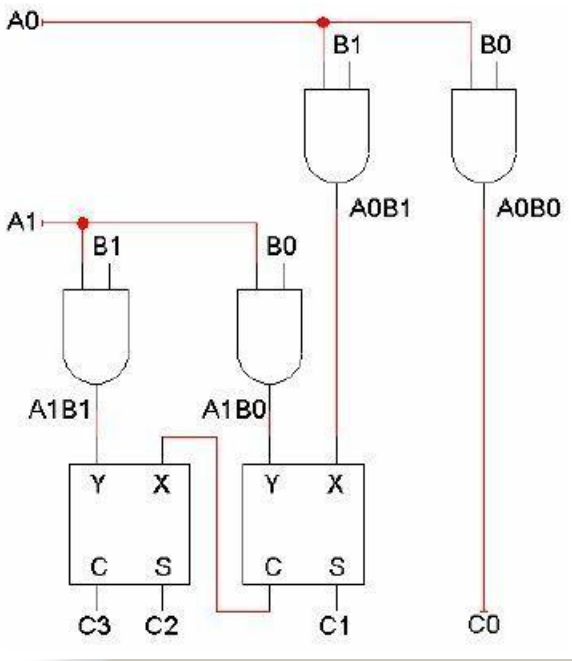


Pipeline Controller

The AND gates produce the partial products.

- For a 2-bit by 2-bit multiplier, we can just use two half adders to sum the partial products
- In general, though, we'll need full adders.
- Here C3-C0are the product, not carries!

$$
\begin{array}{rcccc}
 & & B_1 & B_0 \\
 & \times & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0B_0 \\
+ & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$



Pipeline Controller

- Notice that this 4-bit multiplier produces an 8-bit result.

  We could just keep all 8 bits.

  Or, if we needed a 4-bit result, we could ignore C4-C7, and consider it an overflow condition if the result is longer than 4bits.

- Multipliers are very complex circuits.

  In general, when multiplying an m-bit number by an n-bit number:

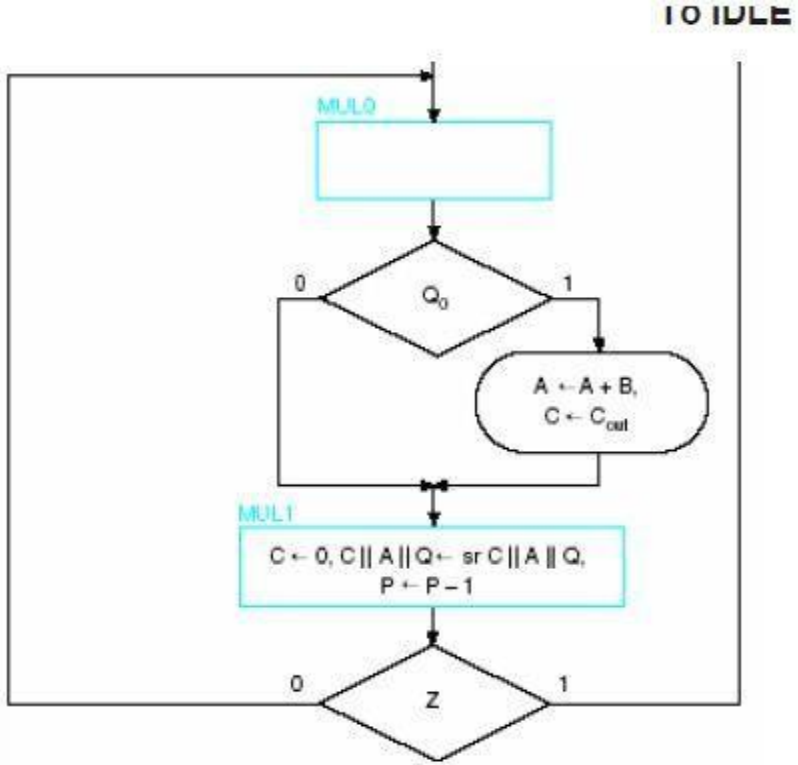
  There are n partial products, one for each bit of the multiplier.

  This requires n-1 adders, each of which can add m bits (the size of the multiplicand).

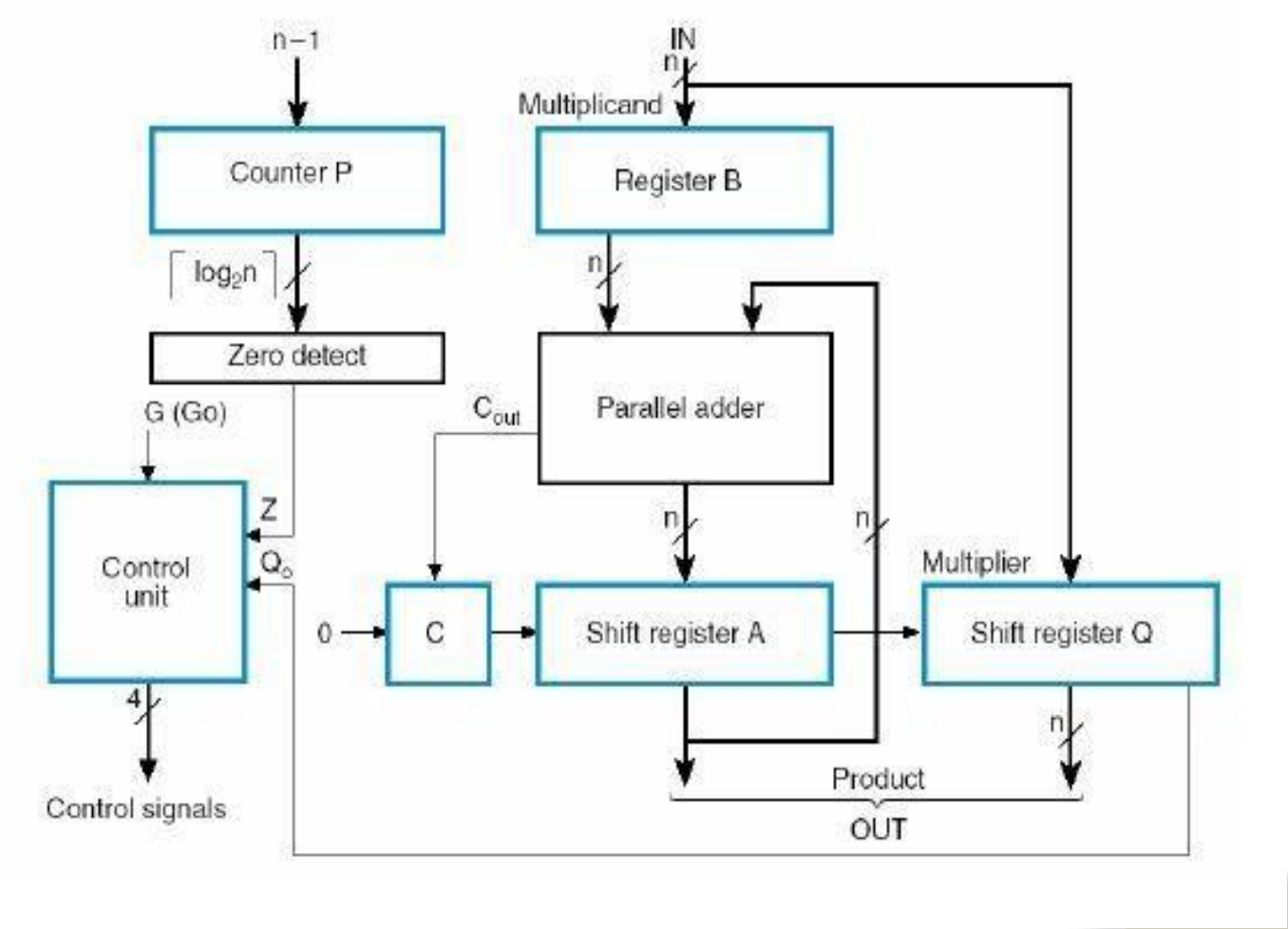
  The circuit for 32-bit or 64-bit multiplication would be huge!

Multiplication

- Test Q_0
  - If 1, add B
- Recall that MUL1 done all at same time
  - What happens to C?
- Test counter zero



Recall: Multiplier

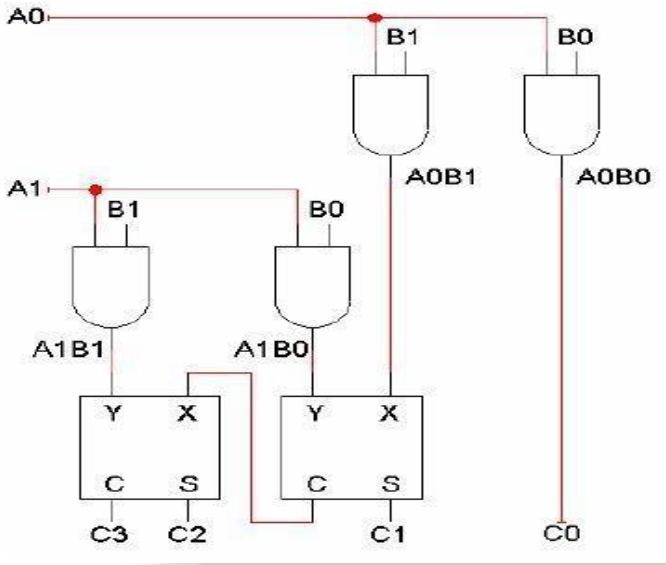


Pipeline Controller

The AND gates produce the partial products:

- For a 2-bit by 2-bit multiplier, we can just use two half adders to sum the partial products
- In general, though, we'll need full adders.
- Here C3-C0are the product, not carries!

$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & \times & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0B_0 \\
+ & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$



Pipeline Controller

- Notice that this 4-bit multiplier produces an 8-bit result.

  We could just keep all 8 bits.

  Or, if we needed a 4-bit result, we could ignore C4-C7, and consider it an overflow condition if the result is longer than 4bits.
- Multipliers are very complex circuits.

  In general, when multiplying an m-bit number by an n-bit number:

  There are n partial products, one for each bit of the multiplier.

  This requires n-1 adders, each of which can add m bits (the size of the multiplicand).

  The circuit for 32-bit or 64-bit multiplication would be huge!

Pipeline Controller

- In decimal, an easy way to multiply by 10 is to shift all the digits to the left, and tack a 0 to the right end.

  128x 10 = 1280

- We can do the same thing in binary. Shifting left is equivalent to multiplying by 2:

  11x 10 = 110(in decimal, 3 x 2 = 6)

- Shifting left twice is equivalent to multiplying by 4:

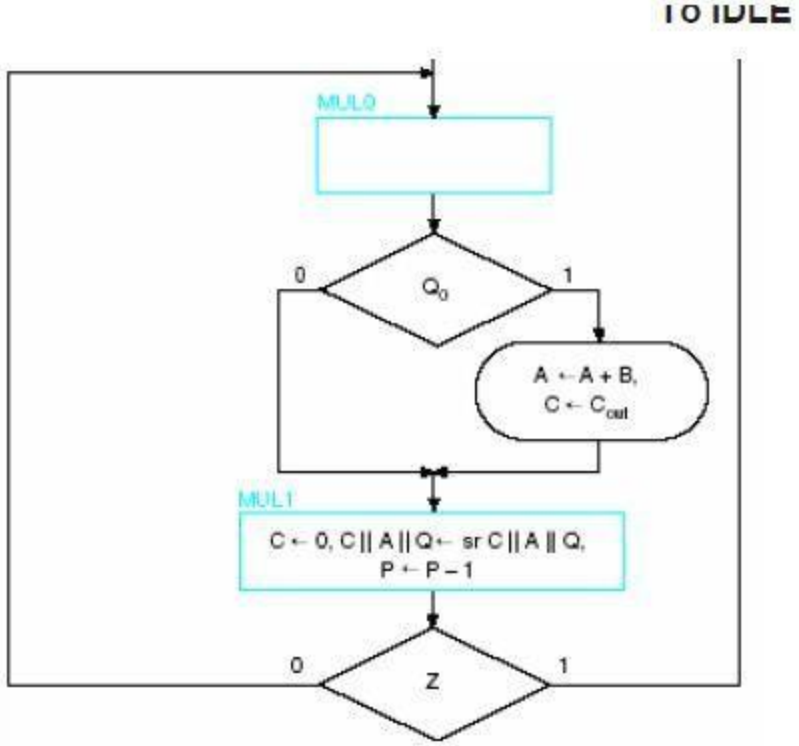
  11x 100 = 1100(in decimal, 3 x 4 = 12)

- As an aside, shifting to the right is equivalent to dividing by 2.

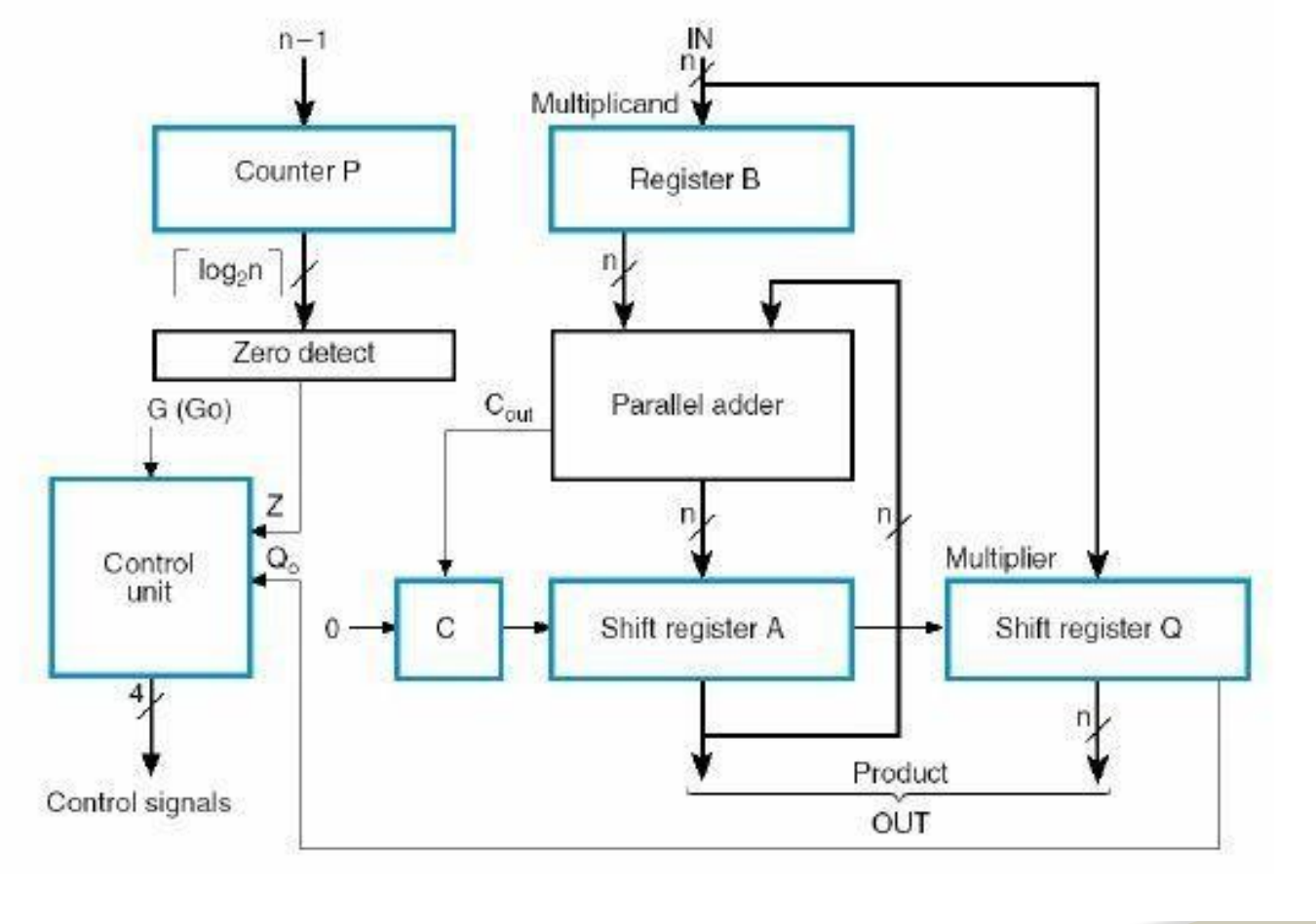
  110÷10 = 11(in decimal, 6 ÷2 = 3)

Multiplication

- Test Q_0
  - If 1, add B
- Recall that MUL1 done all at same time
  - What happens to C?
- Test counter zero



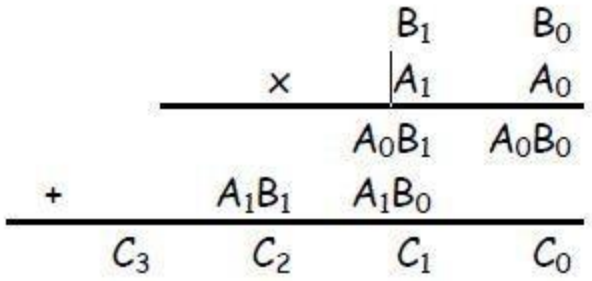
Recall: Multiplier

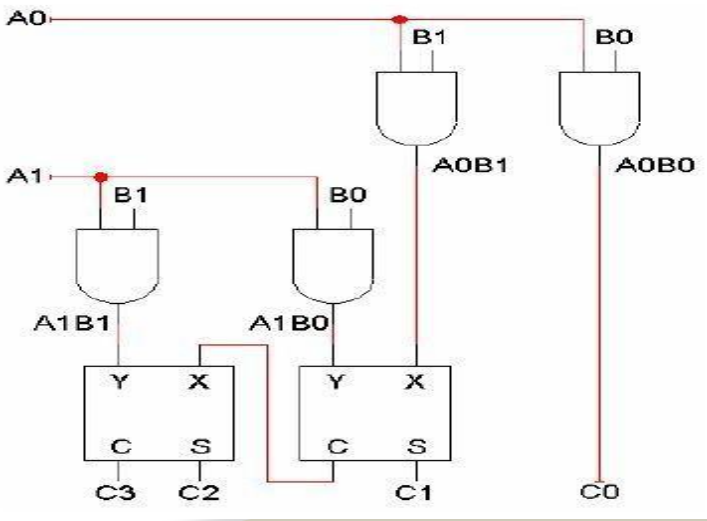


Pipeline Controller

The AND gates produce the partial products.

- For a 2-bit by 2-bit multiplier, we can just use two half adders to sum the partial products
- In general, though, we'll need full adders.
- Here C3-C0are the product, not carries!





Pipeline Controller

- Notice that this 4-bit multiplier produces an 8-bit result.

  We could just keep all 8 bits.

  Or, if we needed a 4-bit result, we could ignore C4-C7, and consider it an overflow condition if the result is longer than 4bits.

- Multipliers are very complex circuits.

  In general, when multiplying an m-bit number by an n-bit number:

  There are n partial products, one for each bit of the multiplier.

  This requires n-1 adders, each of which can add m bits (the size of the multiplicand).

  The circuit for 32-bit or 64-bit multiplication would be huge!

Pipeline Controller

- In decimal, an easy way to multiply by 10 is to shift all the digits to the left, and tack a 0 to the right end.

  128x 10 = 1280

- We can do the same thing in binary. Shifting left is equivalent to multiplying by 2:

  11x 10 = 110(in decimal, 3 x 2 = 6)

- Shifting left twice is equivalent to multiplying by 4:

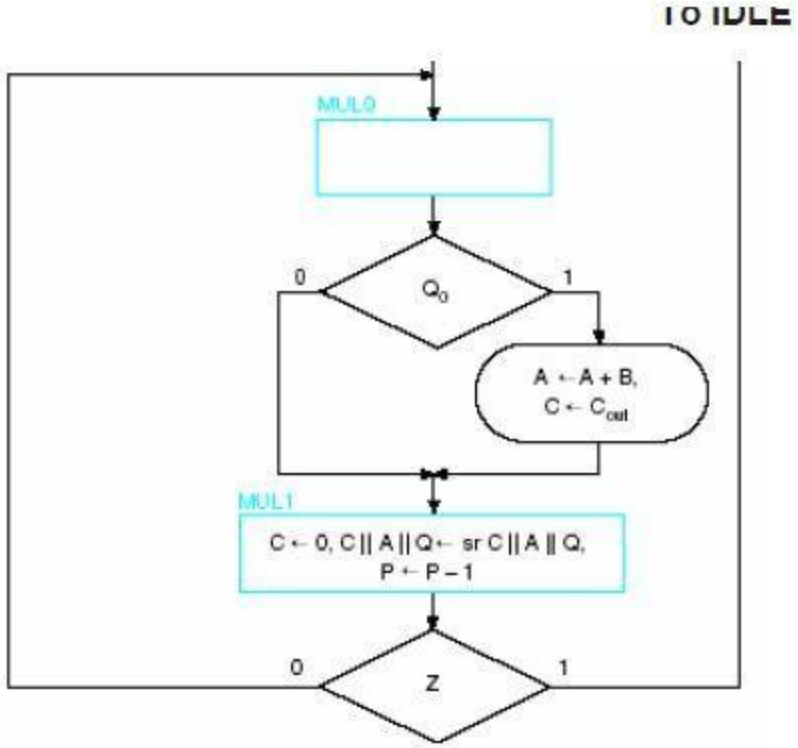
  11x 100 = 1100(in decimal, 3 x 4 = 12)

- As an aside, shifting to the right is equivalent to dividing by 2.

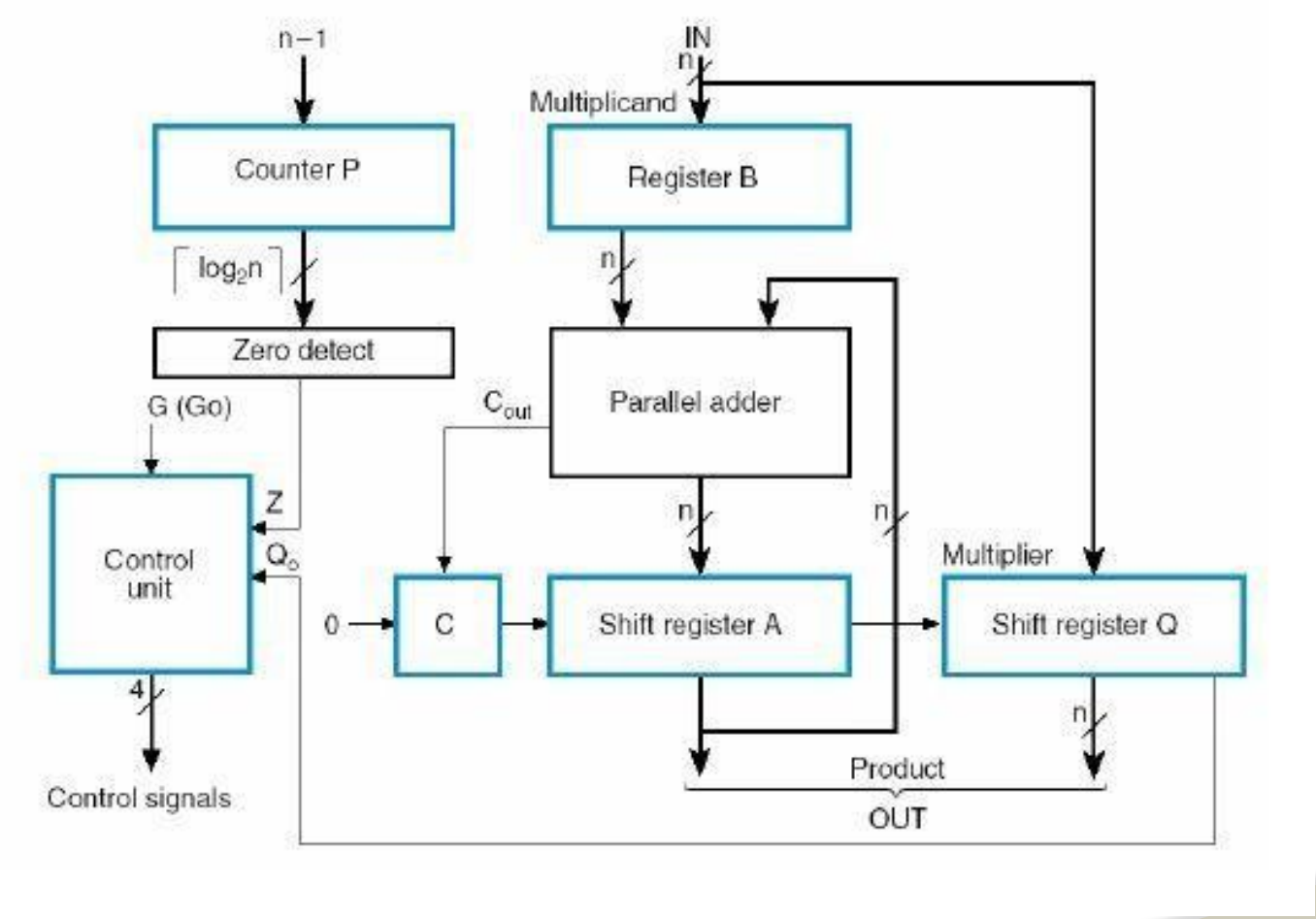
  110÷10 = 11(in decimal, 6 ÷2 = 3)

Multiplication

- Test Q_0
  - If 1, add B
- Recall that MUL1 done all at same time
  - What happens to C?
- Test counter zero



Recall: Multiplier



Introduction

- In conventional micro programmed computers each instruction fetched from main memory is interpreted by micro program stored in a single control memory CM.
- The micro instructions **do not directly issue the signals** that control the

hardware.

- They are used to access a second control memory termed a nano control memory nCM, that **directly controls the hardware.**

2 LEVELS OF CONTROL MEMORY

- Micro controlled memory- higher level
- Nano control memory(Nano instructions)-lower level

Nano Instructions

- Thus a microinstruction is in primary control-store memory, it then has the control signals generated for each microinstruction using a secondary control store memory The output word from the secondary memory is called Nano instruction.
- The µCM stores micro instructions whereas nCM stores nano Instructions.
- The decoder uses Nano instructions from nCMto generate control signals.
- Thus Nano programming gives an alternative strategy to generate control signals. The process of generation of control signals using nano instructions is shown in Figure

Micro Controller

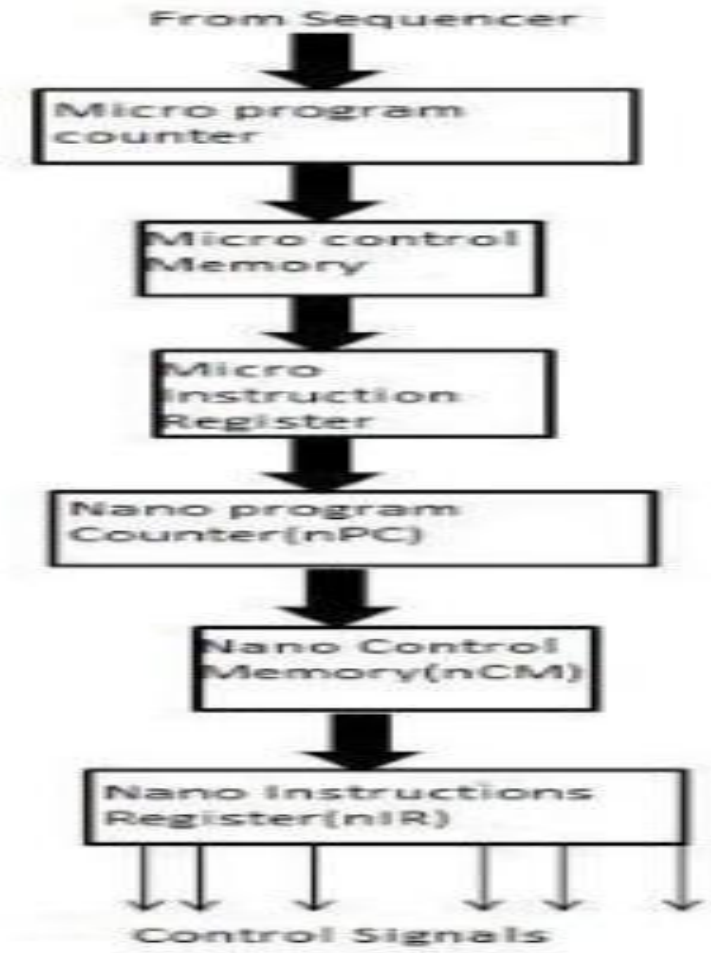


Figure - Nano Programming.

Micro Controller

- Nano instruction addresses are generated by a nano program counter and nano instructions are placed in a register nIR.
- The next address of nIR is directly obtained.
- The next address is generated by either incrementing the nano program counter or loading it from external source (branch field or address from micro instruction opcode)

Nano Programming

Advantages of Nano programming

Reduces total size of required control memory

In two level control design technique, the total control memory size S2can be calculated as *S*2=*HmxWm*+*HnxWn*

Where *H−mn* represents the number of words in the high level memory *Wm* represents the size of word in the high level memory *Hn* represents the number of words in the low level memory*Wn* represents the size of word in the low level memory

Usually, the micro programs are vertically organized so *Hm* is large and *Wm* is small. In Nano programming, we have a highly parallel horizontal organization, which makes Wn large and Hnis small. This gives the compatiblesize for single level control unit as *S*1=*Hmx* Wn which is larger than *S*2. The reduced size of control memory reduces the total chip area.

Nano Programming

Greater design flexibility

Because of two level memories organization more design flexibility exists between instructions and hardware.

Disadvantage of Nano programming

1. Increased memory access time:

The main disadvantage of the two level memory approaches is the loss of speed due to the extra memory access required for Nano control memory.

UNIT IV
MEMORY ORGANIZATION

CLO's	Course Learning outcomes
CLO1	Understand the concept of memory hierarchy and different typed of memory chips.
CLO2	Describe the concepts of magnetic surface recording, optical memories
CLO3	Understand the cache and virtual memory concept in memory organization.
CLO4	Describe the hardware organization of associate memory and understand the read and write operations.

RAM ACCESS MEMORIES

Introduction to Memory:

•A memory unit is the collection of storage units or devices together.

The memory unit stores the binary information in the form of bits.

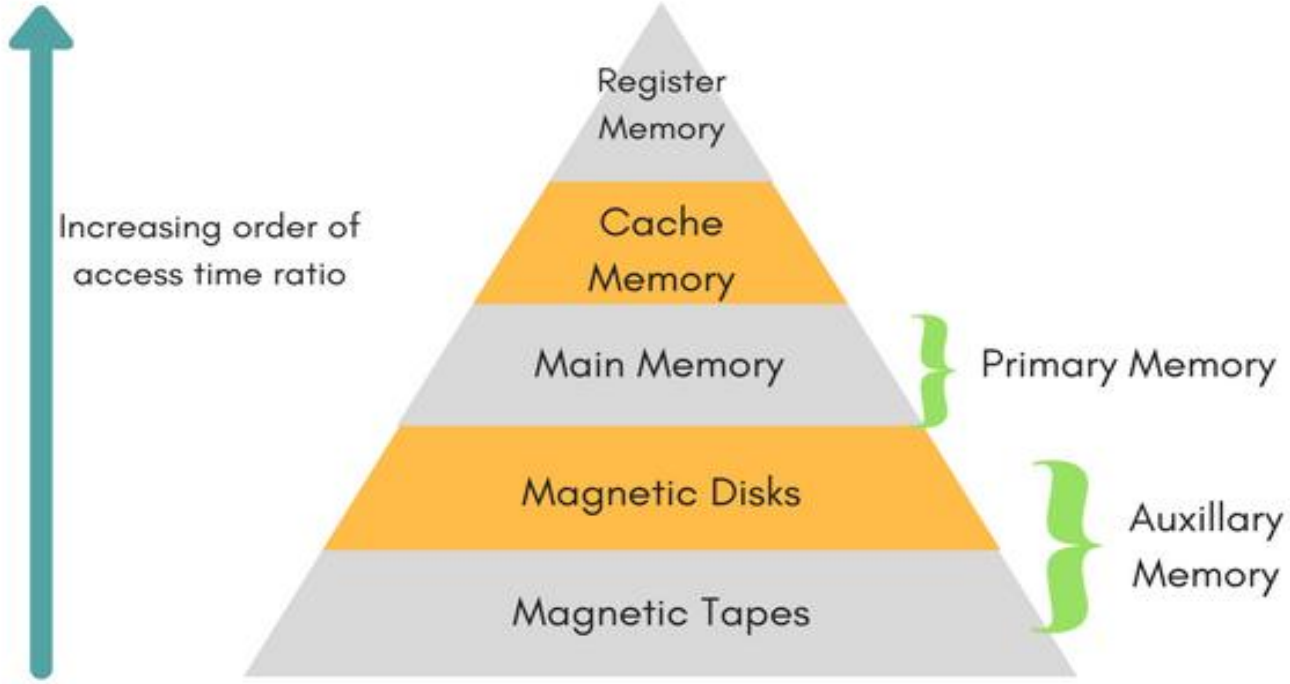
•Generally, memory/storage is classified into 2 categories:

1.Volatile Memory: This loses its data, when power is switched off.

2.Non-Volatile Memory: This is a permanent storage and does not lose any data when power is switched off.

RAM ACCESS MEMORIES

Memory Hierarchy:-



RAM ACCESS MEMORIES

- The total memory capacity of a computer can be visualized by hierarchy of components.
- The memory hierarchy system consists of all storage devices contained in a computer system from the slow Auxiliary Memory to fast Main Memory and to smaller Cache memory.

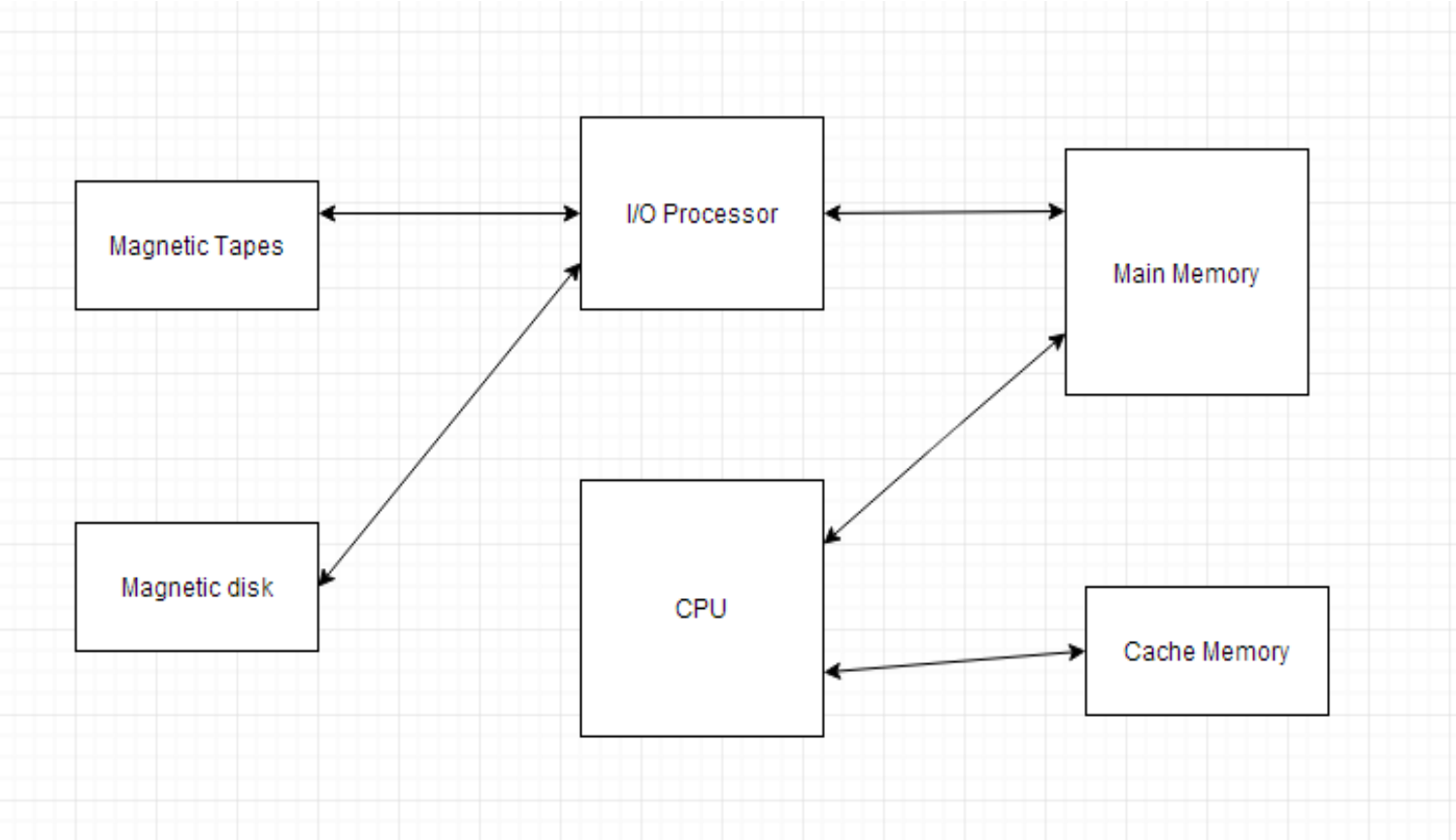
Auxiliary Memory:

Auxiliary memory access time is generally 1000 times that of the main memory, hence it is at the bottom of the hierarchy.

Main Memory:

The main memory occupies the central position because it is equipped to communicate directly with the CPU and with auxiliary memory devices through Input/output processor (I/O).

RAM ACCESS MEMORIES



RAM ACCESS MEMORIES

- Each memory is a collection of numerous memory locations. To access data from any memory, first it must be located and then the data is read from the memory location. Following are the methods to access information from memory locations:

1. **Random Access:** Main memories are random access memories, in which each memory location has a unique address. Using this unique address any memory location can be reached in the same amount of time in any order.
2. **Sequential Access:** This methods allows memory access in a sequence or in order.
3. **Direct Access:** In this mode, information is stored in tracks, with each track having a separate read/write head.

RAM ACCESS MEMORIES

Main Memory:

- The memory unit that communicates directly within the CPU,Auxillary memory and Cache memory, is called main memory.
- It is the central storage unit of the computer system.
- It is a large and fast memory used to store data during computer operations.
- Main memory is made up of RAM and ROM, with RAM integrated circuit chips holing the major share.

MEMORY ACCESS METHODS

RAM: Random Access Memory

a)DRAM: Dynamic RAM, is made of capacitors and transistors, and must be refreshed every 10~100 ms. It is slower and cheaper than SRAM.

b)SRAM: Static RAM, has a six transistor circuit in each cell and retains data, until powered off.

c)NVRAM: Non-Volatile RAM, retains its data, even when turned off.

Example: Flash memory.

MEMORY ACCESS METHODS

ROM: Read Only Memory, is non-volatile and is more like a permanent storage for information. It also stores the bootstrap loader program, to load and start the operating system when computer is turned on.

Types of ROM

a) PROM(Programmable ROM)

b) EPROM(Erasable PROM) and

c) EEPROM(Electrically Erasable PROM)

MEMORY ACCESS METHODS

Auxiliary Memory

- Devices that provide backup storage are called auxiliary memory.
- For example: Magnetic disks and tapes are commonly used auxiliarydevices.
- Other devices used as auxiliary memory are magnetic drums,magnetic bubble memory and optical disks.
- It is not directly accessible to the CPU, and is accessed using the Input/Output channels.

MEMORY ACCESS METHODS

Cache Memory

- The data or contents of the main memory that are used again and again by CPU, are stored in the cache memory so that we can easily access that data in shorter time.
- Whenever the CPU needs to access memory, it first checks the cache memory.
- If the data is not found in cache memory then the CPU moves onto
the main memory.
- It also transfers block of recent data into the cache and keeps on deleting the old data in cache to accomodate the new one.

MEMORY ACCESS METHODS

Hit Ratio

- The performance of cache memory is measured in terms of a quantity called hit ratio.
- When the CPU refers to memory and finds the word in cache it is said to produce a hit.
- If the word is not found in cache, it is in main memory then it counts as a miss.
- The ratio of the number of hits to the total CPU references to memory is called hit ratio.

$$\text{Hit Ratio} = \text{Hit}/(\text{Hit} + \text{Miss})$$

MEMORY ACCESS METHODS

- In random-access memory(RAM) the memory cells can be accessed for information transfer from any desired random location.
- That is, the process of locating a word in memory is the same and requires an equal amount of time no matter where the cells are located physically in memory.
- Communication between a memory and its environment is achieved through data input and output lines, address selection lines, and control lines that specify the direction of transfer.

MEMORY ACCESS METHODS

- The **n** data input lines provide the information to be stored in memory, and the **n** data output lines supply the information coming out of particular word chosen among the 2^k available inside the memory.
- The two control inputs specify the direction of transfer desired.

Write and Read Operations:-

- The two operations that a random access memory can perform are the **write** and **read** operations.
- The write signal specifies a transfer-in operation and the read signal specifies a transfer-out operation.

MEMORY ACCESS METHODS

- On accepting one of these control signals.
- The internal circuits inside the memory provide the desired function.

  The steps that must be taken for the purpose of transferring a new word to be stored into memory are as follows:

1. Apply the binary address of the desired word into the address lines.
2. Apply the data bits that must be stored in memory into the data input lines.
3. Activate the write input.

MEMORY ACCESS METHODS

- The memory unit will then take the bits presently available in the input data lines and store them in the specified by the address lines.

  The steps that must be taken for the purpose of transferring a stored word out of memory are as follows:

1. Apply the **binary address** of the desired word into the address lines.
2. Activate the **read** input.The memory unit will then take the bits from the word that has been selected by the address and apply them into the output data lines. The content of the selected word does not change after reading.

SERIAL ACCESS MEMORIES

Serial Access Memories:-

- Sequential access is a process used for retrieving data from a storage device.
- It is also known as serial access. In sequential access, the storage device moves through all information up to the point it is attempting to read or write.
- An example of sequential access drive is a tape drive where the drive moves the tape forward or backward until the destination is reached. Sequential access memory can also be called "storage system."
- The data is stored and read in a sequential fixed order. Sequential access is the type of memory mostly used for permanent storage, whereas, random access memory is used for temporary storage.

SERIAL ACCESS MEMORIES

Serial Access Devices:-

- Old recording media such as CDs, DVDs, and magnetic tapes are examples of sequential access memory drives.
- Hard drive is also an example of sequential access memory. Examples of random access memory include memory chips and flash memory (such as memory sticks or memory cards).

SERIAL ACCESS MEMORIES

Difference between Sequential Access and Random Access:-

- Comparing sequential versus random disk operations helps to assess systems efficiency.
- Accessing data sequentially is faster than random operations, because it involves more search functions.
- The search operation is performed by the right disk cylinder. It occurs when the disk head positions itself to access the data requested for.
- More ever, random access delivers a lower rate of output.
- If the disk access is random, it is advisable to pay attention and monitor for the emergence of any bottleneck.

SERIAL ACCESS MEMORIES

- For workloads of either random or sequential input/output, it is advisable to use drives with faster rotational speeds.
- For workloads that are predominantly random input/output, it is advisable to use a drive with faster search time.

SERIAL ACCESS MEMORIES

Disadvantages of Sequential Access:-

- The number of records that are affected when updating a file refers to its hit rate.
- Let us consider a file with 5000 records; if there is a delete or an update operation affecting only 50 records, then the hit rate is very low.
- If there are 4500 records that are affected by update or delete operations, then the hit rate is high.
- Sequential access is found to be slow when the hit rate is low.
- It is due to the fact that sequential access has to search all the records in a particular order.
- Moreover, sequential files are executed in a batched transaction to overcome the problem of low hit rate.

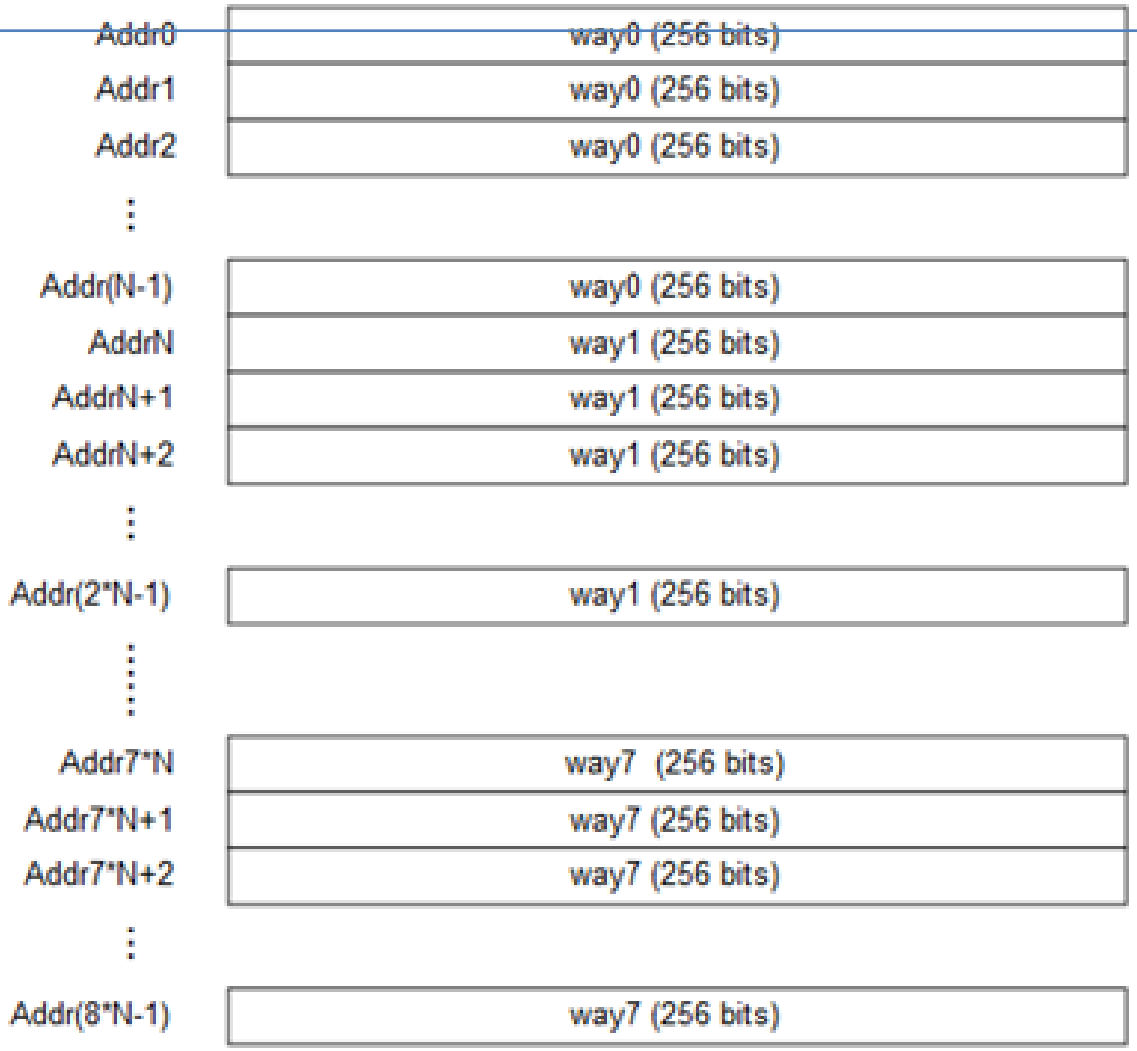
RAM INTERFACES

Data RAM:-

The data RAM shown below is organized as 8 ways 256-bit wide contiguous memories. It supports the following accesses:

- 8 word data reads
- n * 8 bits data writes with byte enables controls
- 8 word data writes for linefills.

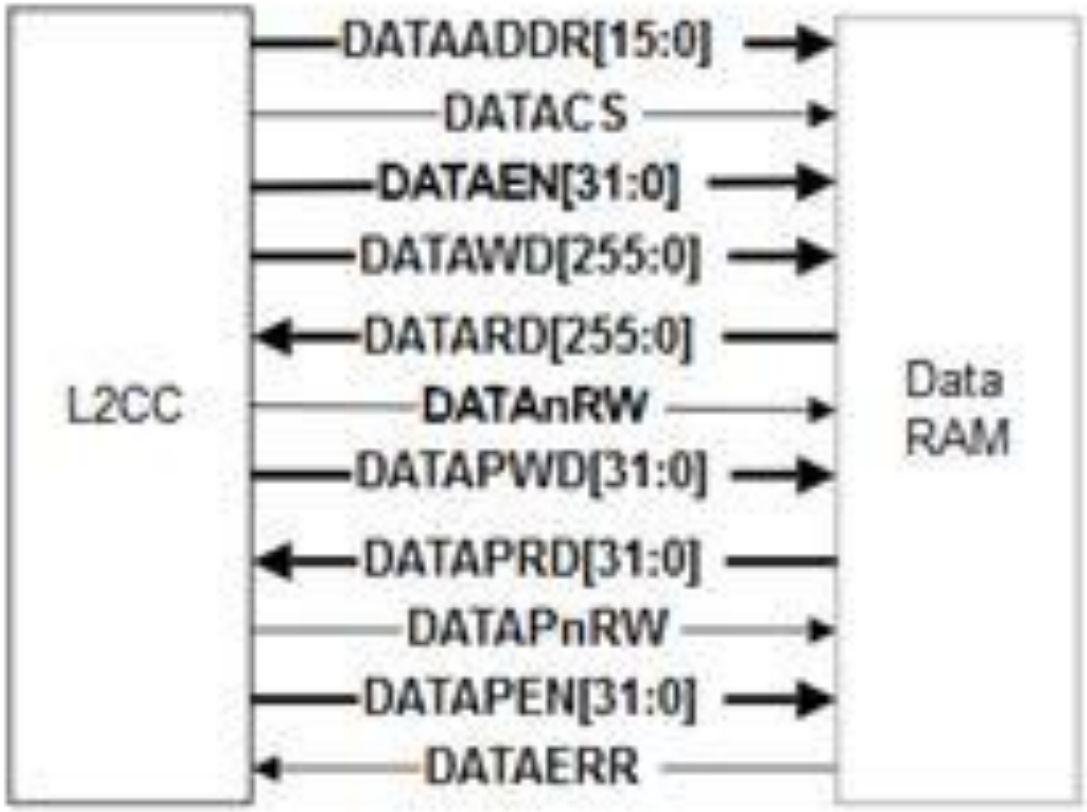
RAM INTERFACES



L2 Cache Size	N =
128KB	512
256KB	1,024
512KB	2,048
1MB	4,096
2MB	8,192

RAM INTERFACES

Data RAM organization:-



RAM INTERFACES

Dirty RAM:-

- The dirty RAM shown below is organized as a 16-bit wide memory, 2 bits per 8-word cache line.
- The dirty RAM address is the same as the tag RAM address bus.
- It supports the following accesses:
    1. 16 bit dirty reads for write-back eviction on a linefill.
    2. 16 bit dirty reads for cache maintenance operations.
    3. 1 or 2 bit dirty writes for writes and allocations.

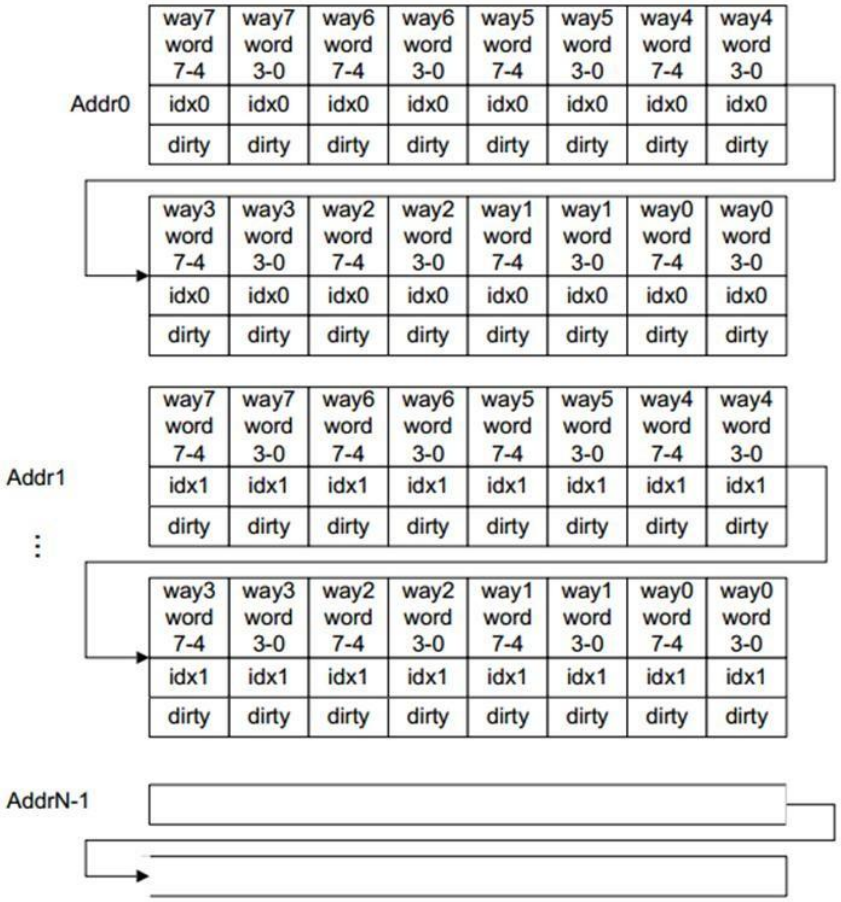
RAM INTERFACES

Dirty RAM organization:-

L2 Cache Size	N =
128KB	512
256KB	1,024
512KB	2,048
1MB	4,096
2MB	8,192

RAM INTERFACES

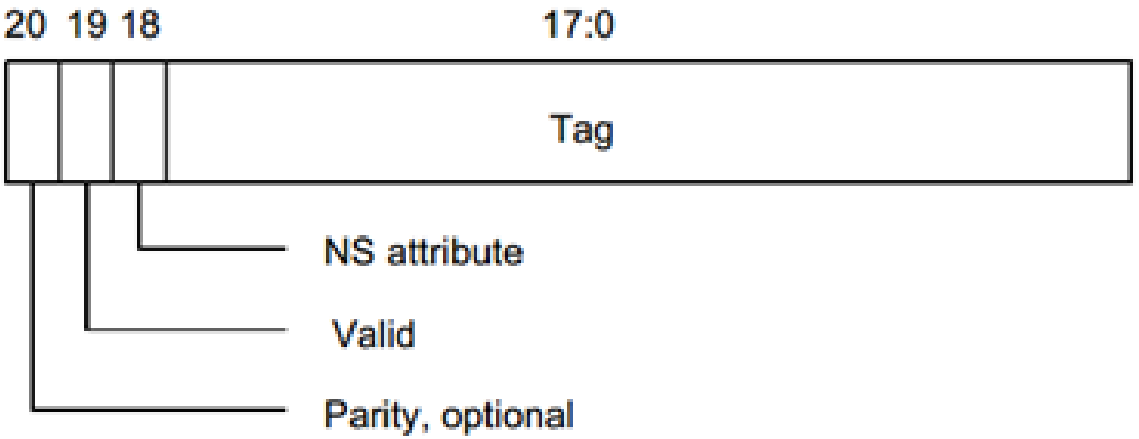
Dirty RAM organization:-



RAM INTERFACES

Cache lookup:-

The tag RAM format:



MAGNETIC SURFACE RECORDING

- A disk is a circular platter constructed of nonmagnetic material,called the substrate, coated with a magnetizable material.
- Traditionally, the substrate has been an aluminum or aluminum alloy material. More recently, glass substrates have been introduced.
- The glass substrate has a number of benefits, including the following:

1. Improvement in the uniformity of the magnetic film surface to increase disk reliability;
2. A significant reduction in overall surface defects to help reduce readwrite errors;
3. Ability to support lower fly heights (described subsequently);
4. Better stiffness to reduce disk dynamics; and
5. Greater ability to withstand shock and damage.

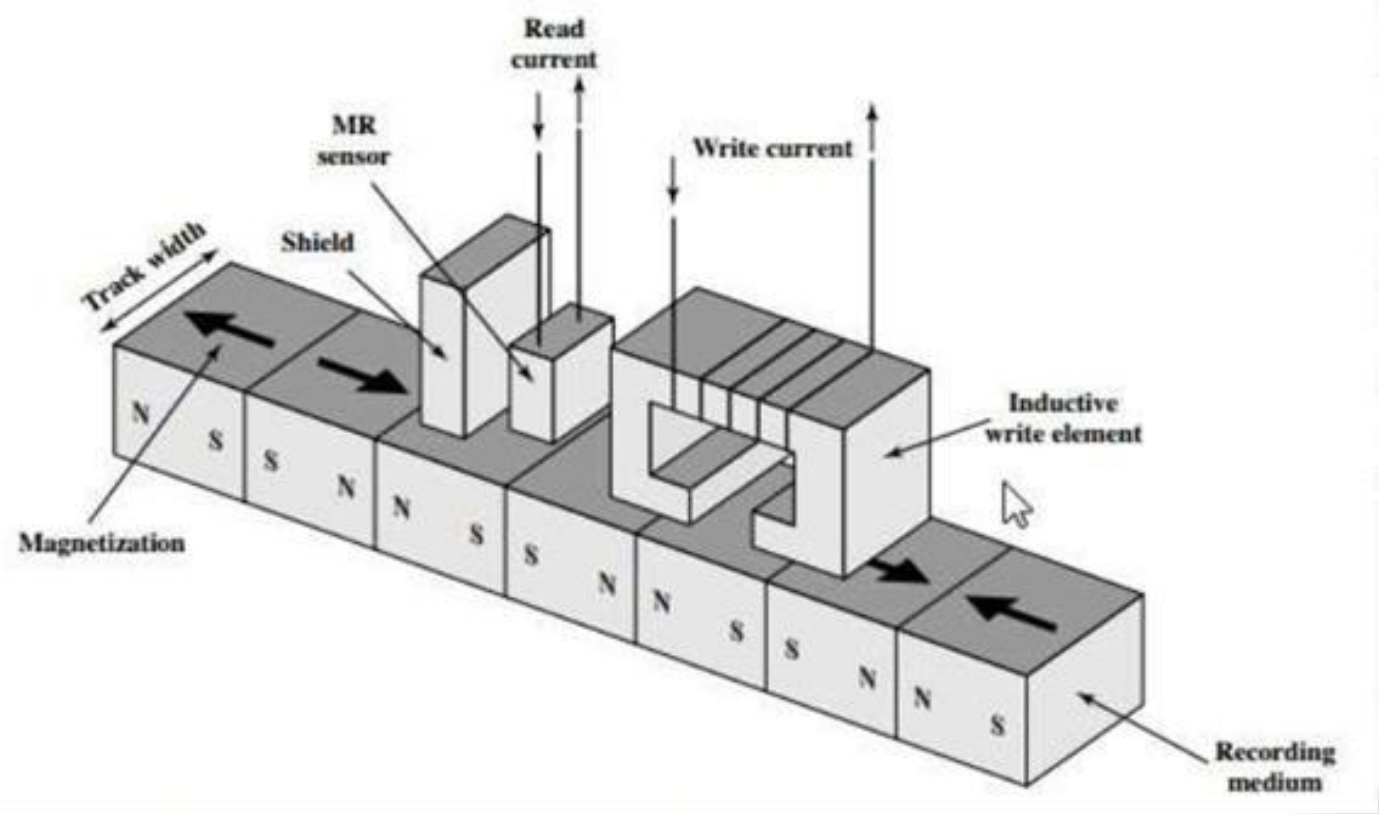
MAGNETIC SURFACE RECORDING

Magnetic Read and Write Memory:-

- Magnetic disks remain the most important component of external memory.
- Both removable and fixed, or hard, disks are used in systems ranging from personal computers to mainframes and supercomputers.
- Data are recorded on and later retrieved from the disk via a conducting coil named the head. In many systems, there are two heads, a read head and a write head.
- During a read or write operation, the head is stationary while the platter rotates beneath it.

MAGNETIC SURFACE RECORDING

Inductive write/Magetoresistive Read Head



MAGNETIC SURFACE RECORDING

- The write mechanism exploits the fact that electricity flowing through a coil produces a magnetic field.
- Electric pulses are sent to the write head, and the resulting magnetic patterns are recorded on the surface below, with different patterns for positive and negative currents.
- The traditional read mechanism exploits the fact that a magnetic field moving relative to a coil produces an electrical current in the coil.
- When the surface of the disk passes under the head, it generates a current of the same polarity as the one already recorded.

MAGNETIC SURFACE RECORDING

- The structure of the head for reading is in this case essentially the same as for writing and therefore the same head can be used for both. Such single heads are used in floppy disk systems and in older rigid disk systems.
- The read head consists of a partially shielded magneto resistive (MR) sensor.
- The MR material has an electrical resistance that depends on the direction of the magnetization of the medium moving under it.

OPTICAL MEMORIES

Optical Memories:

- Optical memories are used for large, storage of data.
- These devices provide the option of variety of data storage.
- These can save up to 20 GB of information.
- The data or information is read or written using a laser beam.
- Due to its low cost and high data storage capacity these memories are being freely used.
- Apart from low cost these memories have long life.But the problem is that of low access time.

OPTICAL MEMORIES

Advantages of CD ROM:

- Storage capacity is high.
- Data storage cost per bit is reasonable.
- Easy to carry.
- Can store variety of data.

Disadvantages of CD ROM:-

- CD ROMs are read only.
- Access time is more than hard disk.

OPTICAL MEMORIES

WORM:

- WORM or Write Once Read Many or CD-R or CD-Record able are a kind of optical device which provides the user the liberty to write once on the CD R.
- The user can write on the disk using the CD R disk drive unit. But this data or information cannot be overwritten or changed.
- CD R does not allow re-writing though reading can be done many times.

OPTICAL MEMORIES

Advantages of WORM:

- Storage capacity is high.
- Can be recorded once.
- Reliable.
- Runs longer.
- Access time is good.

Disadvantages or limitations of WORM:-

- Can be written only once.

OPTICAL MEMORIES

Erasable Optical Disk:

- Erasable Optical Disks are also called CD RW or CD rewritable.
- It gives the user the liberty of erasing data already written by burning the microscopic point on the disk surface.
- The disk can be reused.

Advantages of CD RW:

- Storage capacity is very high.
- Reliability is high.
- Runs longer.
- Easy to rewrite.

MULTILEVEL MEMORY

Multilevel Memories:-

Memory Hierarchy

Memory System Organization

•No matter how big the main memory, how we can organize effectively the memory system in order to store more information than it can hold.

•The traditional solution to storing a great deal of data is a memory hierarchy.

Major design objective of any memory system:-

•To provide adequate storage capacity.

•An acceptable level of performance.

•At a reasonable cost.

MULTILEVEL MEMORY

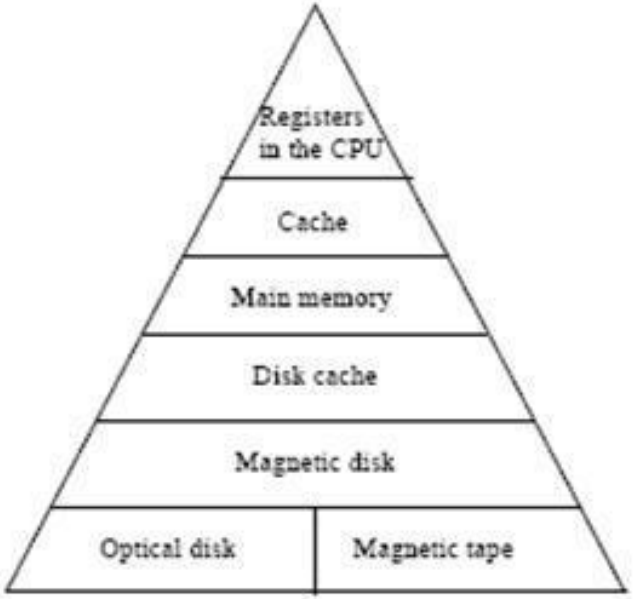
Four interrelated ways to meet this goal

1. Use a hierarchy of storage devices.
2. Develop automatic space allocation methods for efficient use of the memory.
3. Through the use of virtual memory techniques, free the user from memory management tasks.
4. Design the memory and its related interconnection structure so that the processes.

MULTILEVEL MEMORY

Multilevel Memories Organization:-

•Three key characteristics increase for a memory hierarchy. They are the access time, the storage capacity and the cost. The memory hierarchy is illustrated in figure below.



MULTILEVEL MEMORY

- We can see the memory hierarchy with six levels. At the top there are CPU registers, which can be accessed at full CPU speed.
- Next commes the cache memory, which is currently on order of 32 KByte to a few Mbyte.
- The main memory is next, with size currently ranging from 16 MB for entry-level systems to tens of Gigabytes.
- After that come magnetic disks, the current work horse for permanent storage.

MULTILEVEL MEMORY

Finally we have magnetic tape and optical disks for archival storage. Basis of the memory hierarchy

1.Registers internal to the CPU for temporary data storage (small in
number but very fast).

2.External storage for data and programs (relatively large and fast).

3.External permanent storage (much larger and much slower).

MULTILEVEL MEMORY

Characteristics of the memory hierarchy:-

- Consists of distinct “levels” of memory components.
- Each level characterized by its size, access time, and cost per bit.
- Each increasing level in the hierarchy consists of modules of larger capacity, slower access time, and lower cost/bit.

Memory Performance:-

- Goal of the memory hierarchy. Try to match the processor speed with the rate of information transfer from the lowest element in the hierarchy.
- The memory hierarchy speed up the memory performance.
- The memory hierarchy works because of locality of reference.

MULTILEVEL MEMORY

– Memory references made by the processor, for both instructions and data, tend to cluster together

+ Instruction loops, subroutines

+ Data arrays, tables

•Keep these clusters in high speed memory to reduce the average delay in accessing data

•Over time, the clusters being referenced will change -- memory management must deal with this

•Performance of a two level memory

CACHE AND VIRTUAL MEMORY

Cache memory:

- A cache memory is a fast random access memory where the computer hardware stores copies of information currently used by programs (data and instructions), loaded from the main memory.
- The cache has a significantly shorter access time than the main memory due to the applied faster but more expensive implementation technology.
- The cache has a limited volume that also results from the properties of the applied technology.
- If information fetched to the cache memory is used again, the access time to it will be much shorter than in the case if this information were stored in the main memory and the program will execute faster.

CACHE MEMORY

- Time efficiency of using cache memories results from the locality of access to data that is observed during program execution.

We observe here time and space locality:

1. **Time locality** consists in a tendency to use many times the same instructions and data in programs during neighbouring time intervals,
2. **Space locality** is a tendency to store instructions and data used in a program in short distances of time under neighbouring addresses in the main memory.

CACHE MEMORY

- Due to these localities, the information loaded to the cache memory is used several times and the execution time of programs is much reduced.
- Cache can be implemented as a multi-level memory.
- Contemporary computers usually have two levels of caches.
- In older computer models, a cache memory was installed outside a processor (in separate integrated circuits than the processor itself).
- The access to it was organized over the processor external system bus. In today's computers, the first level of the cache memory is installed in the same integrated circuit as the processor.

CACHE MEMORY

- It significantly speeds up processor's co-operation with the cache. Some microprocessors have the second level of cache memory placed also in the processor's integrated circuit.
- The volume of the first level cache memory is from several thousands to several tens of thousands of bytes.
- The second level cache memory has volume of several hundred thousand bytes.
- A cache memory is maintained by a special processor subsystem called cache controller.
- If there is a cache memory in a computer system, then at each access to a main memory address in order to fetch data or instructions, processor hardware sends the address first to the cache memory.

VIRTUAL MEMORY

- In early computers, freedom of programming was seriously restricted by a limited volume of main memory comparing program sizes.
- Small main memory volume was making large programs execution very troublesome and did not enable flexible maintenance of memory space in the case of many co-existing programs.
- It was very uncomfortable, since programmers were forced to spend much time on designing a correct scheme for data and code distribution among the main memory and auxiliary store.

VIRTUAL MEMORY

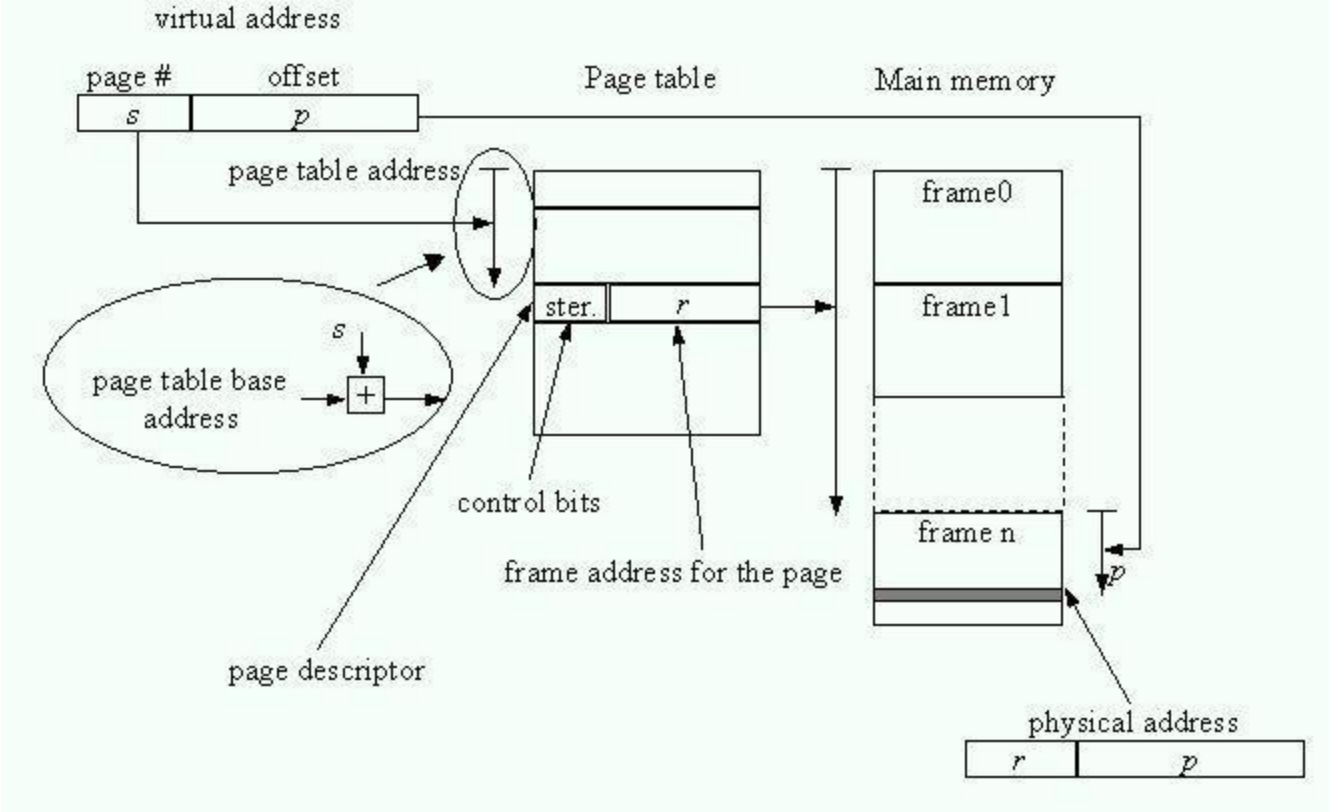
- The solution to this problem was supplied by introduction of the virtual memory concept.
- This concept was introduced at the beginning of years 1970 under the name of **one-level storage** in the British computer called Atlas.
- Only much later, together with application of this idea in computers of the IBM Series 370, the term **virtual memory** was introduced.
- Virtual memory provides a computer programmer with an addressing space many times larger than the physically available addressing space of the main memory.

VIRTUAL MEMORY

- Data and instructions are placed in this space with the use of **virtual addresses**, which can be treated as artificial in some way. In the reality, data and instructions are stored both in the main memory and in the auxiliary memory (usually disk memory).
- It is done under supervision of the virtual memory control system that governs real current placement of data determined by virtual addresses.
- This system automatically (i.e. without any programmer's actions) fetches to the main memory data and instructions requested by currently executed programs.
- The general organization scheme of the virtual memory is shown in the figure below.

VIRTUAL MEMORY

Virtual address translation scheme with two-level page tables:-



VIRTUAL MEMORY

General scheme of the virtual memory:-

- Virtual memory address space is divided into fragments that have pre-determined sizes and identifiers that are consecutive numbers of these fragments in the set of fragments of the virtual memory.
- The sequences of virtual memory locations that correspond to these fragments are called **pages** or **segments**, depending on the type of the virtual memory applied.
- A **virtual memory address** is composed of the number of the respective fragment of the virtual memory address space and the word or byte number in the given fragment.

VIRTUAL MEMORY

We distinguish the following solutions for contemporary virtual memory systems:

1. paged (virtual) memory
2. segmented (virtual) memory
3. segmented (virtual) memory with paging.

VIRTUAL MEMORY

- When accessing data stored under a virtual address, the virtual address has to be converted into a physical memory address by the use of **address translation**.
- Before translation, the virtual memory system checks if the segment or the page, which contains the requested word or byte, resides in the main memory.
- It is done by tests of page or segments descriptors in respective tables residing in the main memory.
- If the test result is negative, a physical address sub-space in the main memory is assigned to the requested page or segment and next it is loaded into this address sub-space from the auxiliary store.

VIRTUAL MEMORY

- Next, the virtual memory system up-dates the page or segment descriptions in the descriptor tables and opens access to the requested word or byte for the processor instruction, which has emitted the virtual address.
- The virtual memory control system is implemented today as partially hardware and software system.
- Accessing descriptor tables and virtual to physical address translation is done by computer hardware.
- Fetching missing pages or segments and up-dating their descriptors is done by the operating system, which, however, is strongly supported by special memory management hardware.

VIRTUAL MEMORY

- This hardware usually constitutes a special functional unit for virtual memory management and special functional blocks designed to perform calculations concerned with virtual address translation.

MEMORY ALLOCATION

Memory allocation:-

- Memory is the processes by which information is encoded, stored and retrieved.
- Encoding allow information that is from the outside world to reach our senses in the forms of chemical and physical stimuli.
- Memory allocation is a process by which computer programs and services are assigned with physical or virtual memory space.
- Memory allocation is the process of reserving a partial or complete portion of computer memory for the execution of programs and processes.
- Memory allocation is achieved through a process known as memory management.

MEMORY ALLOCATION

- Memory allocation is primarily a computer hardware operation but is managed through operating system and software applications.
- Memory allocation process is quite similar in physical and virtual memory management.
- Programs and services are assigned with a specific memory as per their requirements when they are executed.
- Once the program has finished its operation or is idle, the memory is released and allocated to another program or merged within the primary memory.

MEMORY ALLOCATION

Memory allocation has two core types;

1.Static Memory Allocation: The program is allocated memory at compile time.

2.Dynamic Memory Allocation: The programs are allocated with memory at run time.

Static memory allocation:

In static memory allocation, size of the memory may be required for the calculation that must be define before loading and executing the program.

MEMORY ALLOCATION

Dynamic memory allocation:-

There are two methods which are used for dynamic memory allocation:

- Non-Preemptive Allocation
- Preemptive Allocation

Non Preemptive allocation:-

- Consider M1 as a main memory and M2 as secondary memory and
  a block K of n words is to be transferred from M2 to M1.
- For such memory allocation it is necessary to find or create an available reason of n or more words to accommodate K.
- This process is known as nonpreemptive allocation.

MEMORY ALLOCATION

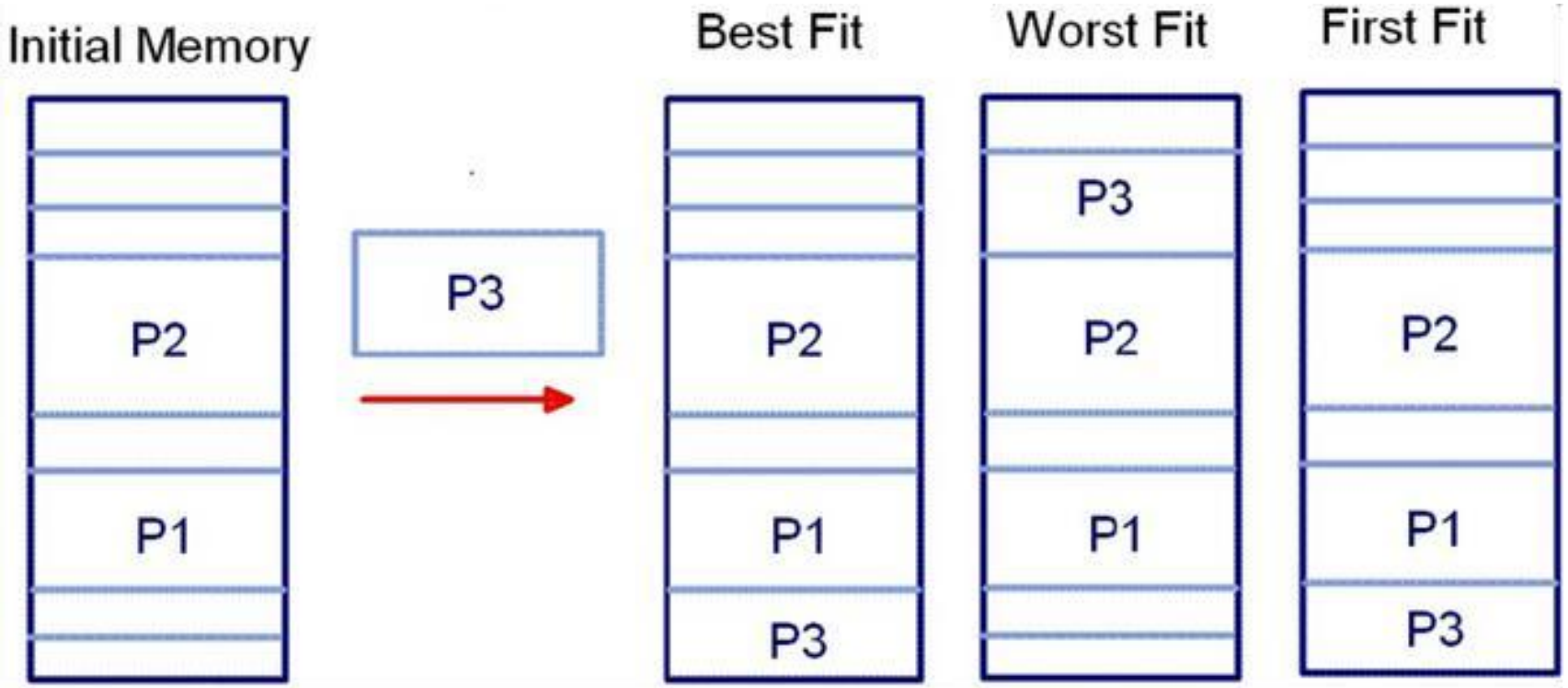
First Fit

•In this algorithm, searching is started either at the beginning of the memory or where the previous first search ended.

Best fit

•In this algorithm, all free memory blocks are searched and smallest free memory block which is large enough to accommodate desired block K is used to allocate K.

MEMORY ALLOCATION

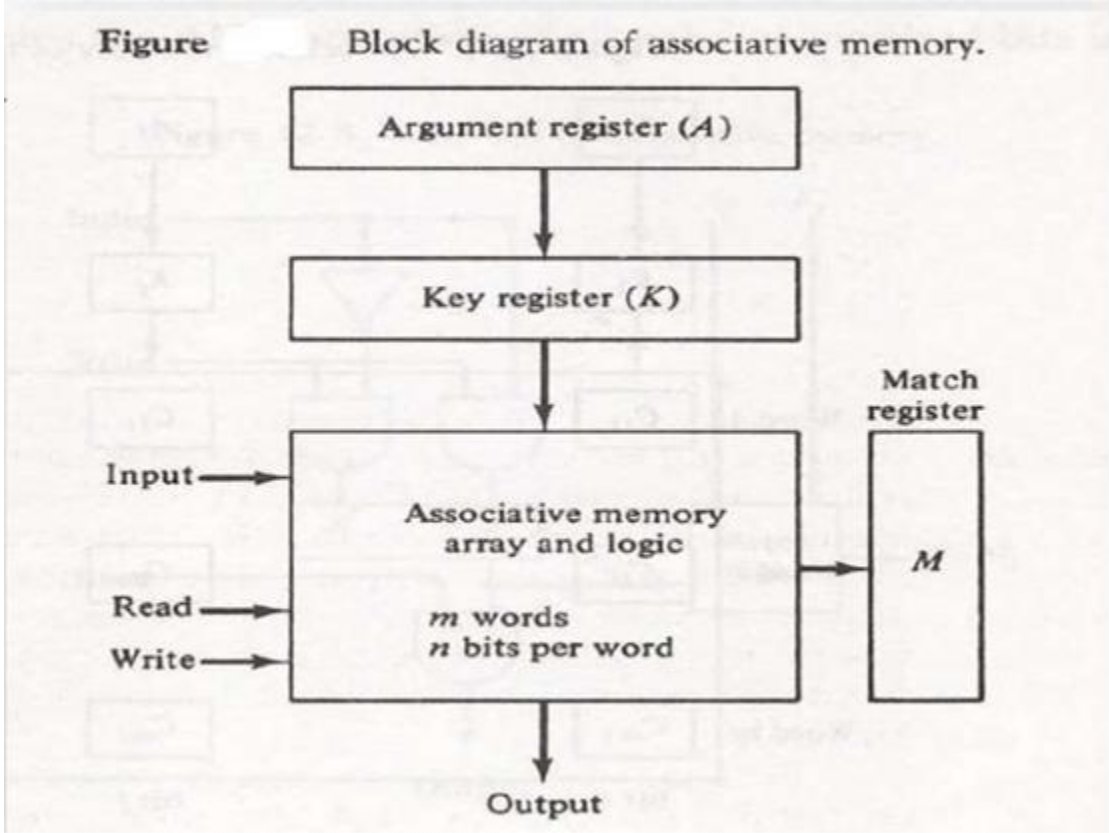


MEMORY ALLOCATION

Preemptive allocation:-

- Non preemptive allocation can't make efficient use of memory in all situation.
- Due scattered memory blocks larger free memory blocks may not be available.
- Much more efficient us of the available memory space is possible if the occupied space can be re allocated to make room for incoming blocks by a method called as Compaction.

HARDWARE ORGANIZATION OF ASSOCIATIVE MEMORY



HARDWARE ORGANISATION

Associative Memory is organized in such a way.

1.Argument register(A): It contains the word to be searched. It has n bits(one for each bit of the word).

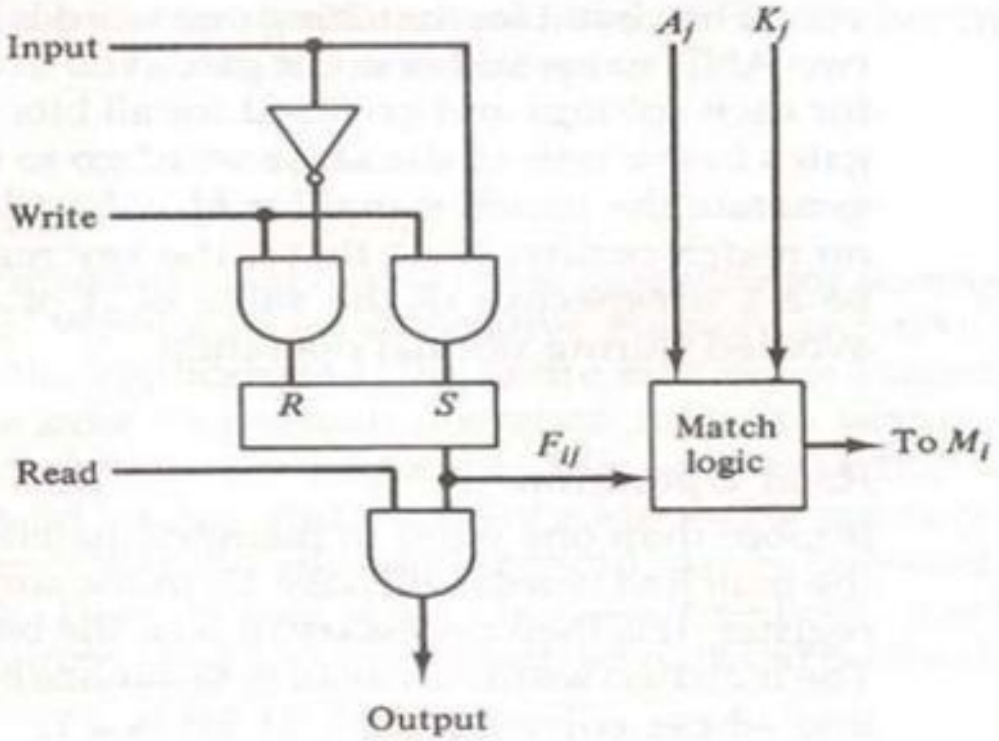
2.Key Register(K): This specifies which part of the argument word needs to be compared with words in memory. If all bits in register are 1, The entire word should be compared. Otherwise, only the bits having k bit set to 1 will be compared.

3.Associative memory array: It contains the words which are to be compared with the argument word.

4.Match Register(M): It has m bits, one bit corresponding to each word in the memory array. After the matching process, the bits corresponding to matching words in match register are set to 1.

MATCH LOGIC

Figure One cell of associative memory.



MATCH LOGIC

- Let us include key register. If Kj=0 then there is no need to compare Aj and Fij.

1. Only when Kj=1, comparison is needed.
2. This achieved by ORing each term with Kj.

$$M_i = (x_1 + K_1')(x_2 + K_2')(x_3 + K_3') \cdots (x_n + K_n')$$

READ OPERATION

When a word is to be read from an associative memory, the contents of the word, or a prt of the word is specified.

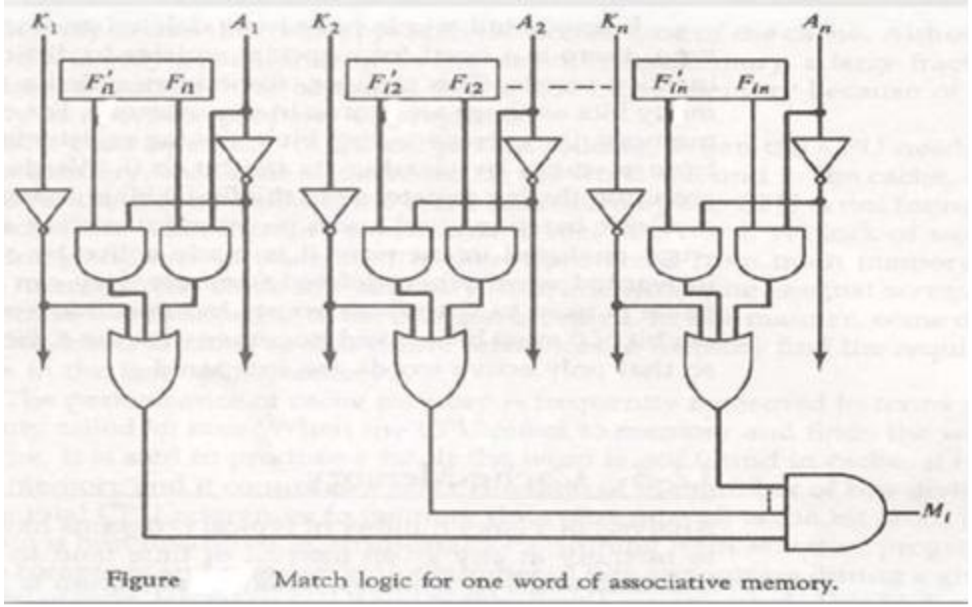


Figure Match logic for one word of associative memory.

WRITE OPERATION

1. If the entire memory is loaded with new information at once prior to search operation then writing can be done by addressing each location in sequence.
2. Tag register contain as many bits as there are words in memory. It contain 1 for active word and 0 for inactive word.
3. If the word is to be inserted, tag register is scanned until 0 is found and word is written at that position and bit is change to 1.

ADVANTAGES

This is suitable for parallel searches. It is also used where search time needs to be short.

1.Associative memory is often used to speed up databases, in neural networks and in the page tables used by the virtual memory of modern computers.

2.CAM-design challenge is to reduce power consumption associated with the large amount of parallel active circuitry, without sacrificing speed or memory density.

DISADVANTAGES

1. An associative memory is more expensive than a random access memory because each cell must have an extra storage capability as well as logic circuits for matching its content with an external argument.
2. Usually associative memories are used in applications where the search time is very critical and must be very short.

UNIT V
SYSTEM ORGANIZATION

UNIT-V

CLO's	Course Learning outcomes
CLO1	Understand the various bus control interfaces and system control interfaces.
CLO2	Describe the various interrupts (Vectored Interrupts, PCI interrupts, Pipeline interrupts).
CLO3	Understand the functionally of RISC and CISC processors.
CLO4	Describe the concepts of superscalar and vector processor.

IO ORGANZATION

Input/Output Processor:-

•An input-output processor (IOP) is a processor with direct memory access capability.

•In this, the computer system is divided into a memory unit and number of processors. Each IOP controls and manage the input-output tasks.

•The IOP is similar to CPU except that it handles only the details of I/O processing.

•The IOP can fetch and execute its own instructions. These IOP instructions are designed to manage I/O transfers only.

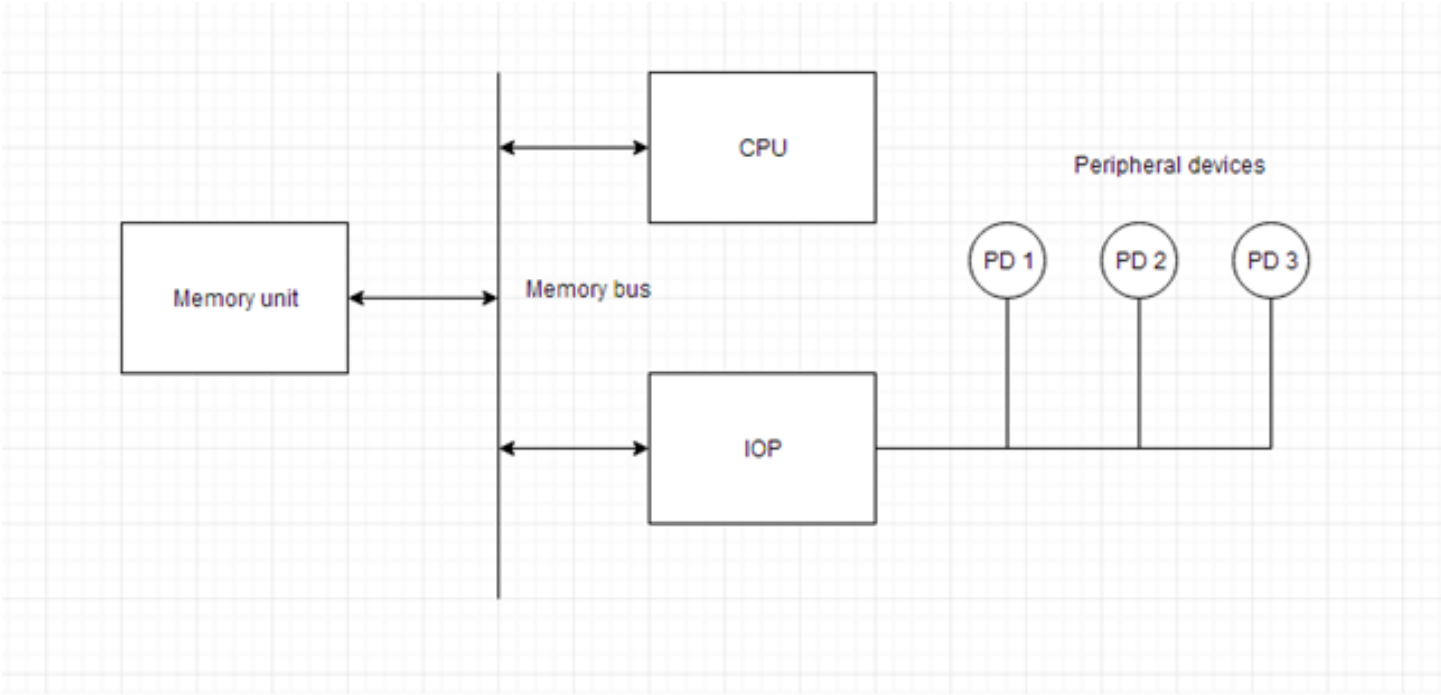
IO ORGANZATION

Block Diagram of IOP

- Below is a block diagram of a computer along with various I/O Processors.
- The memory unit occupies the central position and can communicate with each processor.
- The CPU processes the data required for solving the computational tasks.
- The IOP provides a path for transfer of data between peripherals and memory.
- The CPU assigns the task of initiating the I/O program.
- The IOP operates independent from CPU and transfer data between peripherals and memory.

IO ORGANZATION

Block Diagram of IOP



IO ORGANZATION

- The communication between the IOP and the devices is similar to the program control method of transfer.
- And the communication with the memory is similar to the direct memory access method.
- In large scale computers, each processor is independent of other processors and any processor can initiate the operation.
- The CPU can act as master and the IOP act as slave processor.
- The CPU assigns the task of initiating operations but it is the IOP, who executes the instructions, and not the CPU.
- CPU instructions provide operations to start an I/O transfer. The IOP asks for CPU through interrupt.

IO ORGANZATION

- Instructions that are read from memory by an IOP are also called *commands* to distinguish them from instructions that are read by CPU
- Commands are prepared by programmers and are stored in memory. Command words make the program for IOP.
- CPU informs the IOP where to find the data.

MULTIPROCESSOR

Multiprocessors:

- A multiprocessor system is an interconnection of two or more CPU's with memory and input- output equipment.
- Multiprocessors system are classified as multiple instruction stream, multiple data stream systems(MIMD).
- There exists a distinction between multiprocessor and multicomputers that though both support concurrent operations.
- In multicomputers several autonomous computers are connected through a network and they may or may not communicate but in a multiprocessor system there is a single OS Control that provides interaction between processors and all the components of the system to cooperate in the solution of the problem.

MULTIPROCESSOR

- VLSI circuit technology has reduced the cost of the computers to such a low Level that the concept of applying multiple processors to meet system performance requirements has become an attractive design possibility.

MULTIPROCESSOR

Benefits of Multiprocessing:

1.Multiprocessing increases the reliability of the system so that a failure or error in one part has limited effect on the rest of the system. If a fault causes one processor to fail, a second processor can be assigned to perform the functions of the disabled one.

2.Improved System performance. System derives high performance from the fact that computations can proceed in parallel in one of the two ways:

a) Multiple independent jobs can be made to operate in parallel.

b) A single job can be partitioned into multiple parallel tasks. This can be achieved in two ways:

MULTIPROCESSOR

- The user explicitly declares that the tasks of the program be executed in parallel.
- The compiler provided with multiprocessor s/w that can automatically detect parallelism in program. Actually it checks for Data dependency.

FAULT TOLERANCE

- Such systems automatically detect a failure of the computer processor unit, I/O subsystem, memory cards, motherboard, power supply or network components.
- The failure point is identified, and a backup component or procedure immediately takes its place with no loss of service.
- To ensure fault tolerance, enterprises need to purchase an inventory of formatted computer equipment and a secondary uninterruptible power supply device.
- The goal is to prevent the crash of key systems and networks, focusing on issues related to uptime and downtime.
- Fault tolerance can be provided with software embedded in hardware, or by some combination of the two.

FAULT TOLERANCE

- In a software implementation, the operating system (OS) provides an interface that allows a programmer to checkpoint critical data at predetermined points within a transaction.
- In a hardware implementation (for example, with Stratus and its Virtual Operating System), the programmer does not need to be aware of the fault-tolerant capabilities of the machine.
- At a hardware level, fault tolerance is achieved by duplexing each hardware component.
- Disks are mirrored. Multiple processors are lockstepped together and their outputs are compared for correctness.
- When an anomaly occurs, the faulty component is determined and taken out of service, but the machine continues to function as usual.

FAULT TOLERANCE

Fault tolerance vs. high availability:-

- Fault tolerance is closely associated with maintaining business continuity via highly available computer systems and networks.
- Fault-tolerant environments are defined as those that restore service instantaneously following a service outage, whereas a high- availability environment strives for five nines of operational service.
- In a high-availability cluster, sets of independent servers are coupled loosely together to guarantee system-wide sharing of critical data and resources.
- The clusters monitor each other's health and provide fault recovery to ensure applications remain available.

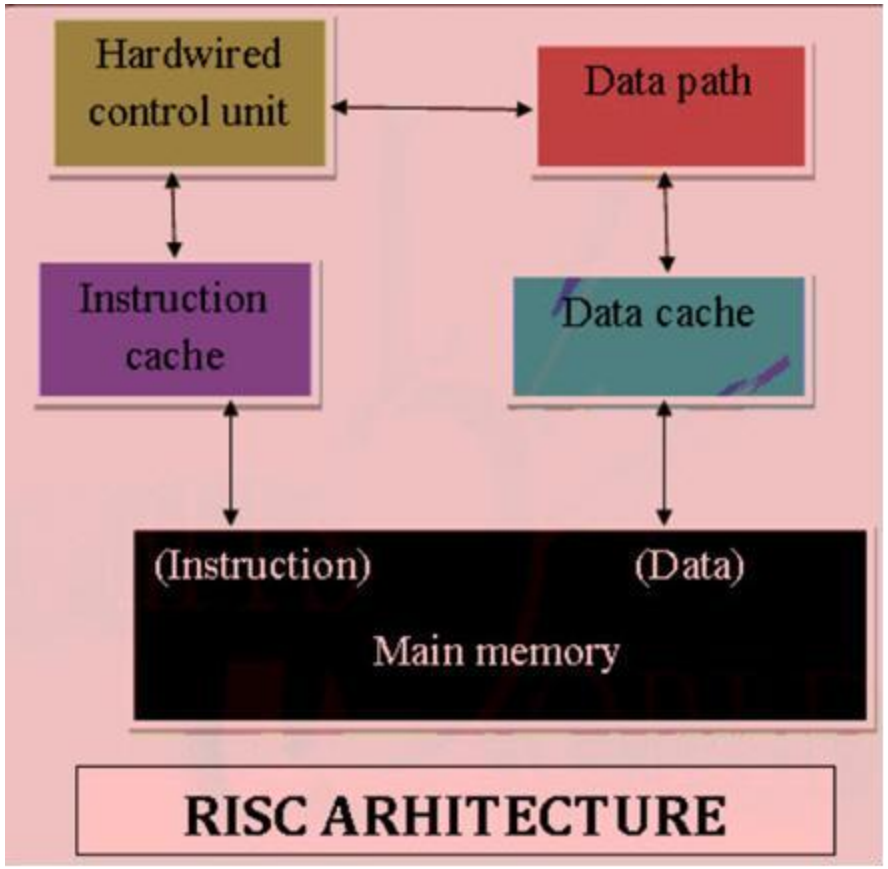
FAULT TOLERANCE

- Conversely, a fault-tolerant cluster consists of multiple physical systems that share a single copy of a computer's OS.
- Software commands issued by one system are also executed on the other system. The tradeoff between fault tolerance and high availability is cost.
- Systems with integrated fault tolerance incur a higher cost due to the inclusion of additional hardware.

RISC PROCESSORS

- RISC (Reduced Instruction Set Computer) is used in portable devices due to its power efficiency.
- For Example, Apple iPod and Nintendo DS. RISC is a type of microprocessor architecture that uses highly-optimized set of instructions.
- RISC does the opposite, reducing the cycles per instruction at the cost of the number of instructions per program Pipelining is one of the unique feature of RISC.
- It is performed by overlapping the execution of several instructions in a pipeline fashion.
- It has a high performance advantage over CISC.

RISC PROCESSORS



RISC processors take simple instructions and are executed within a clock cycle

RISC PROCESSORS

RISC ARCHITECTURE CHARACTERISTICS:-

1.Simple Instructions are used in RISC architecture.

2.RISC helps and supports few simple data types and synthesize complex data types.

3.RISC utilizes simple addressing modes and fixed length instructions for pipelining.

4.RISC permits any register to use in any context.

5.One Cycle Execution Time.

6.The amount of work that a computer can perform is reduced by separating “LOAD” and “STORE” instructions.

7.RISC contains Large Number of Registers in order to prevent various number of interactions with memory.

RISC PROCESSORS

7. RISC contains Large Number of Registers in order to prevent various number of interactions with memory.
8. In RISC, Pipelining is easy as the execution of all instructions will be done in a uniform interval of time i.e. one click.
9. In RISC, more RAM is required to store assembly level instructions.
10. Reduced instructions need a less number of transistors in RISC.
11. RISC uses Harvard memory model means it is Harvard Architecture.
12. A compiler is used to perform the conversion operation means to convert a high-level language statement into the code of its form.

RISC PROCESSORS

Comparison between CISC & RISC

CISC	RISC
It is prominent on Hardware	It is prominent on the Software
It has high cycles per second	It has low cycles per second
It has transistors used for storing Instructions which are complex	More transistors are used for storing memory
LOAD and STORE memory-to-memory is induced in instructions	LOAD and STORE register-register are independent
It has multi-clock	It has a single - clock

RISC PROCESSORS

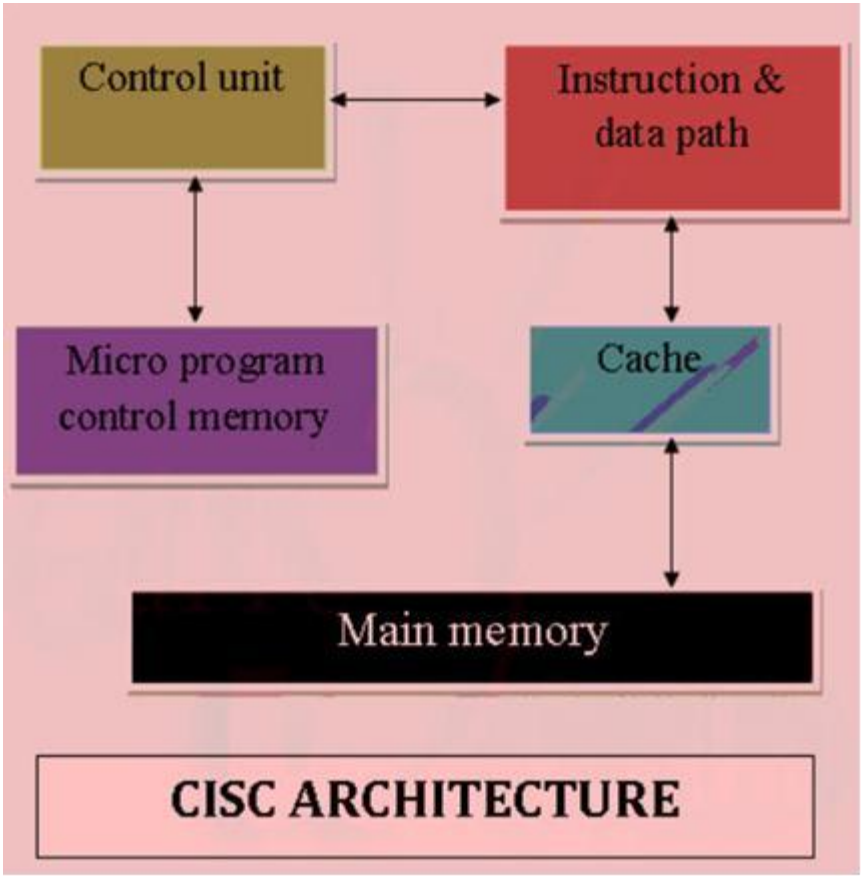
MUL instruction is divided into three instructions

1. “LOAD” – moves data from the memory bank to a register.
2. “PROD” – finds product of two operands located within the registers.
3. “STORE” – moves data from a register to the memory banks.

CISC PROCESSORS

- The CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction.
- Computers based on the CISC architecture are designed to decrease the memory cost.
- Because, the large programs need more storage, thus increasing the memory cost and large memory becomes more expensive.
- To solve these problems, the number of instructions per program can be reduced by embedding the number of operations in a single instruction, thereby making the instructions more complex.

CISC PROCESSORS



CISC processors take simple instructions and are executed within a clock cycle

CISC PROCESSORS

- MUL loads two values from the memory into separate registers in CISC.
- CISC uses minimum possible instructions by implementing hardware and executes operations.
- Instruction Set Architecture is a medium to permit communication between the programmer and the hardware. Data execution part, copying of data, deleting or editing is the user commands used in the microprocessor and with this microprocessor the Instruction set architecture is operated.

CISC PROCESSORS

The main keywords used in the above Instruction Set Architecture are as below

Instruction Set:

- Group of instructions given to execute the program and they direct the computer by manipulating the data.
- Instructions are in the form – Opcode (operational code) and Operand.
- Where, opcode is the instruction applied to load and store data, etc.
- The operand is a memory register where instruction applied.

CISC PROCESSORS

Addressing Modes:

- Addressing modes are the manner in the data is accessed.
- Depending upon the type of instruction applied, addressing modes are of various types such as direct mode where straight data is accessed or indirect mode where the location of the data is accessed.
- Processors having identical ISA may be very different in organization.
- Processors with identical ISA and nearly identical organization is still not nearly identical.

CISC PROCESSORS

CPU performance is given by the fundamental law

$$CPU\ Time = \frac{Seconds}{Program} = \frac{Instructions}{Program} X \frac{Cycles}{Instructions} X \frac{Seconds}{Cycle}$$

Thus, CPU performance is dependent upon Instruction Count, CPI (Cycles per instruction) and Clock cycle time. And all three are affected by the instruction set architecture.

Instruction Count of the CPU

	Instruction Count	CPI	Clock
Program	X		
Compiler	X	X	
Instruction Set Architecture	X	X	X
Microarchitecture		X	X
Physical Design			X

CISC PROCESSORS

- This underlines the importance of the instruction set architecture.

There are two prevalent instruction set architectures

Examples of CISC PROCESSORS

- **IBM 370/168 –** It was introduced in the year 1970. CISC design is a

32 bit processor and four 64-bit floating point registers.

- **VAX 11/780 –** CISC design is a 32-bit processor and it supports many numbers of addressing modes and machine instructions which is from Digital Equipment Corporation.
- **Intel 80486 –** It was launched in the year 1989 and it is a CISC processor, which has instructions varying lengths from 1 to 11 and it will have 235 instructions.

CISC PROCESSORS

CHARACTERISTICS OF CISC ARCHITECTURE:-

- Instruction-decoding logic will be Complex.
- One instruction is required to support multiple addressing modes.
- Less chip space is enough for general purpose registers for the instructions that are 0operated directly on memory.
- Various CISC designs are set up two special registers for the stack pointer, handling interrupts, etc.
- MUL is referred to as a "complex instruction" and requires the programmer for storing functions.

SUPERSCALAR AND VECTOR PR0CESSOR

Superscalar and vector processor:

•A **Scalar processor** is a normal processor, which works on simple instruction at a time, which operates on single data items.

•But in today's world, this technique will prove to be highly inefficient, as the overall processing of instructions will be very slow.

Vector(Array) Processing

•There is a class of computational problems that are beyond the capabilities of a conventional computer.

•These problems require vast number of computations on multiple data items, that will take a conventional computer(with scalar processor) days or even weeks to complete.

SUPERSCALAR AND VECTOR PR0CESSOR

- Such complex instructions, which operates on multiple data at the same time, requires a better way of instruction execution, which was achieved by Vector processors.
- Scalar CPUs can manipulate one or two data items at a time, which is not very efficient.
- Also, simple instructions like **ADD A to B, and store into C** are not practically efficient.
- Addresses are used to point to the memory location where the data to be operated will be found, which leads to added overhead of data lookup.
- So until the data is found, the CPU would be sitting ideal, which is a big performance issue.

SUPERSCALAR AND VECTOR PR0CESSOR

- Hence, the concept of **Instruction Pipeline** comes into picture, in which the instruction passes through several sub-units in turn.
- These sub-units perform various independent functions, **for example**: the **first** one decodes the instruction, the **second** sub-unit fetches the data and the **third** sub-unit performs the math itself.
- Therefore, while the data is fetched for one instruction, CPU does not sit idle, it rather works on decoding the next instruction set, ending up working like an assembly line.
- Vector processor, not only use Instruction pipeline, but it also pipelines the data, working on multiple data at the same time.

SUPERSCALAR AND VECTOR PR0CESSOR

- A normal scalar processor instruction would be ADD A, B, which leads to addition of two operands, but what if we can instruct the processor to ADD a group of numbers(from 0 to n memory location) to another group of numbers(lets say, n to k memory location).
- This can be achieved by vector processors.
- In vector processor a single instruction, can ask for multiple data operations, which saves time, as instruction is decoded once, and then it keeps on operating on different data items.

SUPERSCALAR AND VECTOR PR0CESSOR

Applications of Vector Processors:-

•Computer with vector processing capabilities are in demand in specialized applications. The following are some areas where vector processing is used:

•Petroleum exploration.

•Medical diagnosis.

•Data analysis.

•Weather forecasting.

•Aerodynamics and space flight simulations.

•Image processing.

•Artificial intelligence.

SUPERSCALAR AND VECTOR PR0CESSOR

Superscalar Processors:-

- It was first invented in 1987. It is a machine which is designed to improve the performance of the scalar processor.
- In most applications, most of the operations are on scalar quantities.
- Superscalar approach produces the high performance general purpose processors.
- The main principle of superscalar approach is that it executes instructions independently in different pipelines.
- As we already know, that Instruction pipelining leads to parallel processing thereby speeding up the processing of instructions.

SUPERSCALAR AND VECTOR PR0CESSOR

- In Superscalar processor, multiple such pipelines are introduced for different operations, which further improves parallel processing.
- There are multiple functional units each of which is implemented as a pipeline.
- It increases the throughput because the CPU can execute multiple instructions per clock cycle.
- Thus,superscalar processors are much faster than scalar processors.

SUPERSCALAR AND VECTOR PR0CESSOR

- A **scalar processor** works on one or two data items, while the **vector processor** works with multiple data items.
- A **superscalar processor** is a combination of both. Each instruction processes one data item, but there are multiple execution units within each CPU thus multiple instructions can be processing separate data items concurrently.
- While a superscalar CPU is also pipelined, there are two different performance enhancement techniques.
- It is possible to have a non-pipelined superscalar CPU or pipelined non-superscalar CPU.

SUPERSCALAR AND VECTOR PR0CESSOR

The superscalar technique is associated with some characteristics, these are:

1.Instructions are issued from a sequential instruction stream.

2.CPU must dynamically check for data dependencies.

3.Should accept multiple instructions per clock cycle.

VECTOR PROCESSOR

Vector(Array) Processor and its Types:-

- Array processors are also known as multiprocessorsor vector
  processors.
- They perform computations on large arrays of data. Thus, they are
  used to improve the performance of the computer.

Types of Array Processors:-

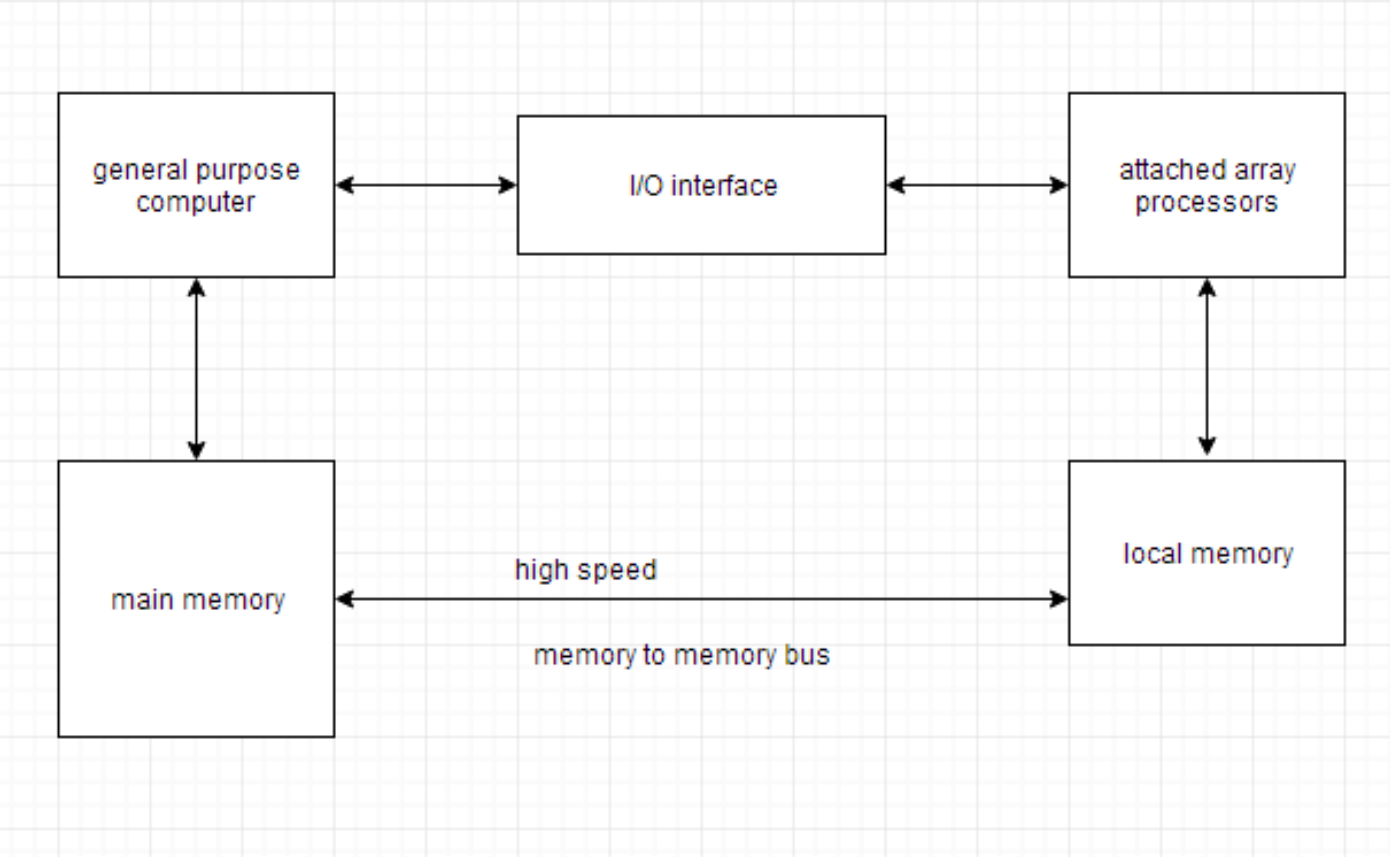
- There are basically two types of array processors:
- Attached Array Processors
- SIMD Array Processors

VECTOR PROCESSOR

Attached Array Processors:-

- An attached array processor is a processor which is attached to a general purpose computer and its purpose is to enhance and improve the performance of that computer in numerical computational tasks.
- It achieves high performance by means of parallel processing with multiple functional units.

VECTOR PROCESSOR



VECTOR PROCESSOR

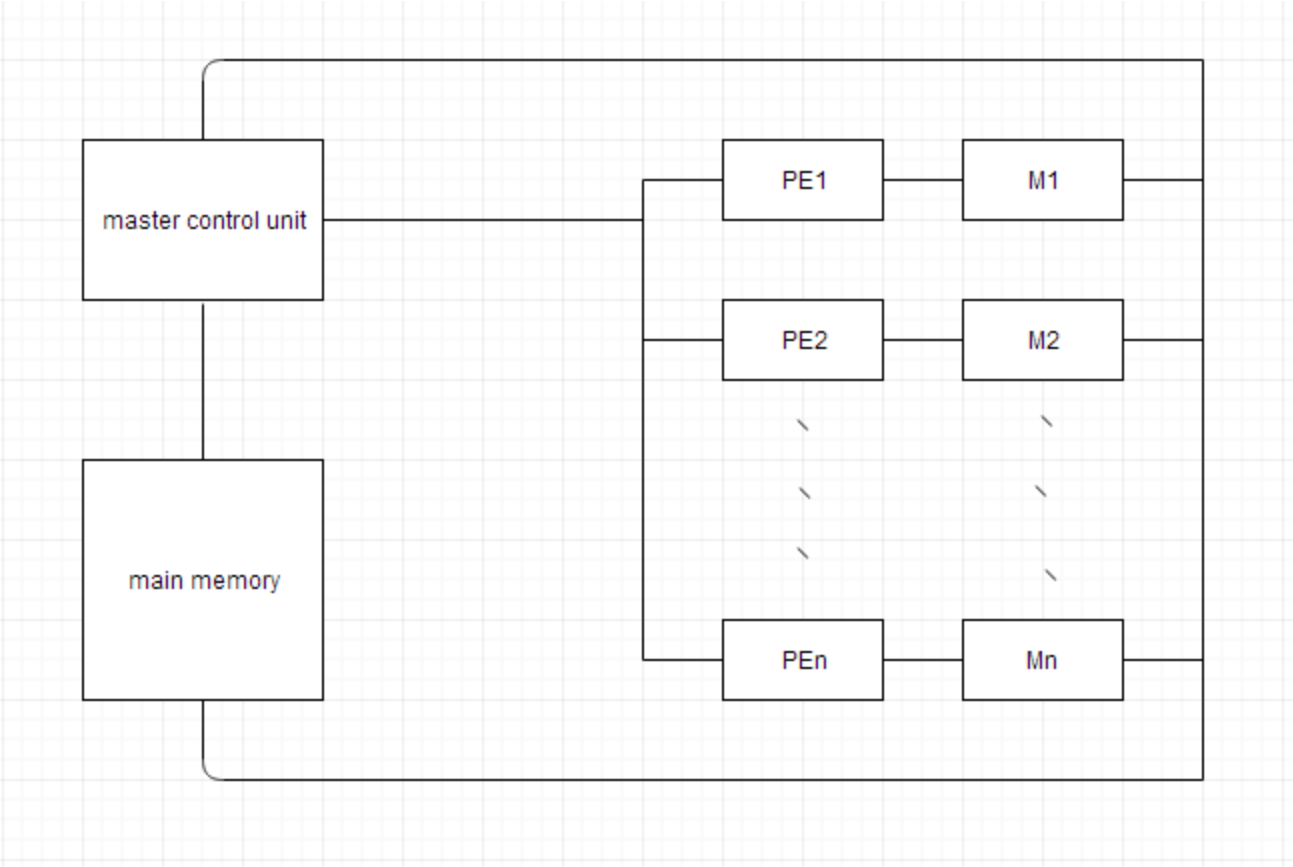
SIMD Array Processors:-

- SIMD is the organization of a single computer containing multiple processors operating in parallel.
- The processing units are made to operate under the control of a common control unit, thus providing a single instruction stream and multiple data streams.
- A general block diagram of an array processor is shown below. It contains a set of identical processing elements (PE's), each of which is having a local memory M.
- Each processor element includes an ALU and registers.
- The master control unit controls all the operations of the processor elements.
- It also decodes the instructions and determines how the instruction is to be executed.

VECTOR PROCESSOR

- The main memory is used for storing the program. The control unit is responsible for fetching the instructions.
- Vector instructions are send to all PE's simultaneously and results are returned to the memory.
- The best known SIMD array processor is the ILLIAC IV computer developed by the Burroughs corps.
- SIMD processors are highly specialized computers.
- They are only suitable for numerical problems that can be expressed in vector or matrix form and they are not suitable for other types of computations.

VECTOR PROCESSOR



VECTOR PROCESSOR

Why use the Array Processor:-

- Array processors increases the overall instruction processing speed.
- As most of the Array processors operates asynchronously from the
  host CPU, hence it improves the overall capacity of the system.
- Array Processors has its own local memory, hence providing extra
  memory for systems with low memory.

IO System

Accessing I/O Devices I/O interface

Input/output mechanism

–Memory-mapped I/O

–Programmed I/O

–Interrupts

–Direct Memory Access

Buses

–Synchronous Bus

–Asynchronous Bus

•IO in Computer Organization and Operating System:

1.Programmed I/O

2.Interrupts

3.DMA (Direct memory Access

IO in Computer Organization and Operating System:

Device	Behavior	Partner	Data rate (Mbit/sec)
Keyboard	input	human	0.0001
Mouse	input	human	0.0038
Voice input	input	human	0.2640
Sound input	input	machine	3.0000
Scanner	input	human	3.2000
Voice output	output	human	0.2640
Sound output	output	human	8.0000
Laser printer	output	human	3.2000
Graphics display	output	human	800.0000–8000.0000
Modem	input or output	machine	0.0160–0.0640
Network/LAN	input or output	machine	100.0000–1000.0000
Network/wireless LAN	input or output	machine	11.0000–54.0000
Optical disk	storage	machine	80.0000
Magnetic tape	storage	machine	32.0000
Magnetic disk	storage	machine	240.0000–2560.0000

IO System

• An IO interface consists of the circuitry required to connect output device to a computer system.

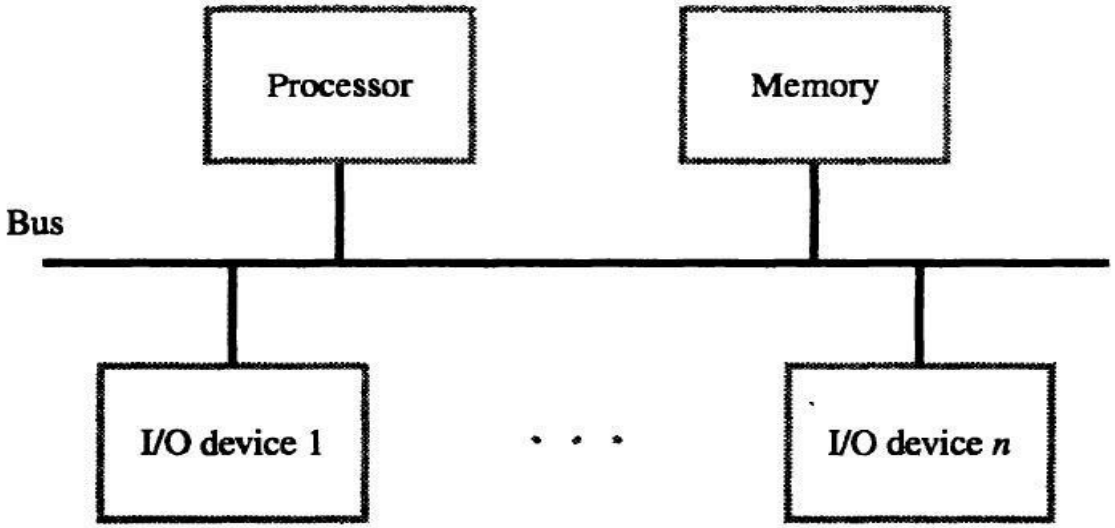


Figure – Bus Connection IO devices to Processor and memory.

IO System

- A bus is a shared communication link, which uses one set of wires to connect multiple subsystems.
- The two major advantages of the bus organization are versatility and low cost.
- Accessing I/O Devices:
- Most modern computers use single bus arrangement for connecting I/O devices to CPU & Memory.
- The bus enables all the devices connected to it to exchange information.
- Bus consists of 3 set of lines: *Address, Data, and Control.*
- Processor places a particular address (unique for an I/O Dev.) on address lines.
- Device which recognizes this address responds to the
- Commands issued on the Control lines.
- Processor requests for either Read / Write.
- The data will be placed on Data lines.

IO System

- Hardware to connect I/O devices to bus:
- Interface Circuit
- Address Decoder
- Control Circuits
- Data registers
- Status registers
- The Registers in I/O Interface – buffer and control.
- Flags in Status Registers, like SIN, SOUT
- Data Registers, like Data-IN, Data-OUT

IO System

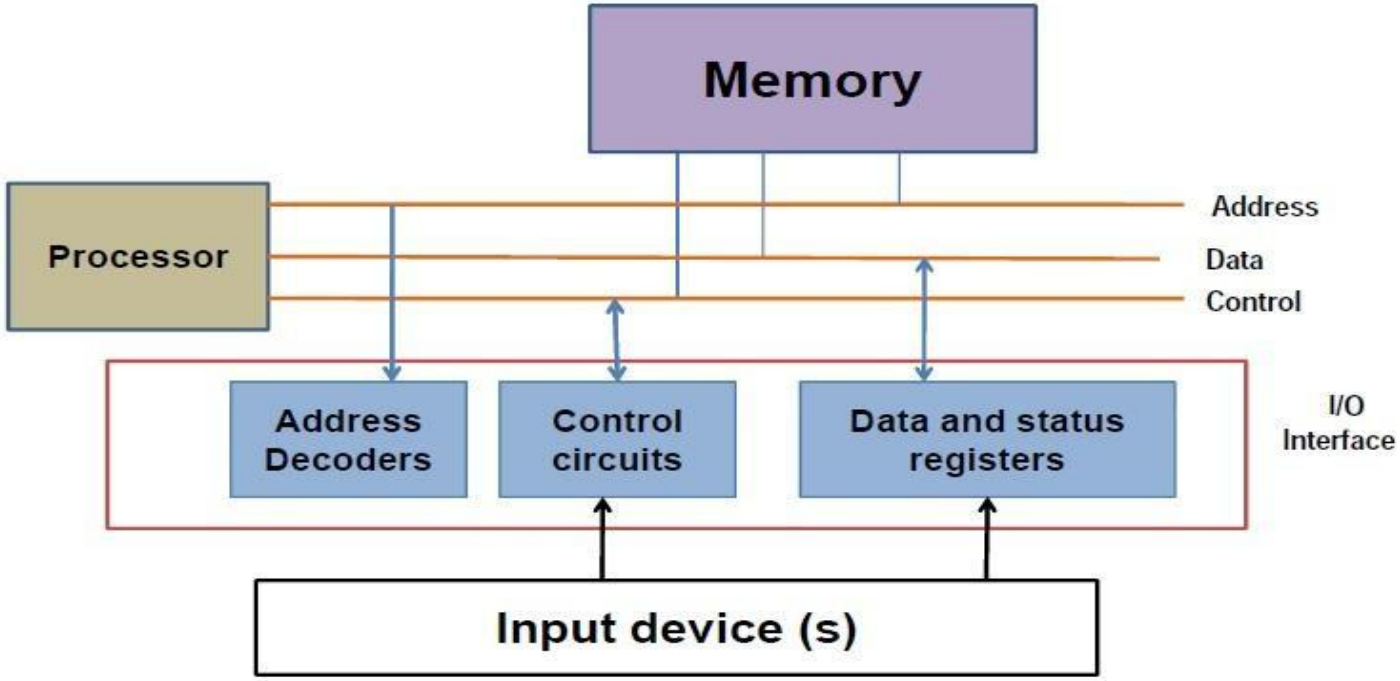


Figure – IO interface for an input device:

IO System

Input Output mechanism:

- Memory mapped I/O
- Programmed I/O
- Interrupts
- DMA (Direct memory Access)

IO System

- I/O devices and the memory share the same address space, the arrangement is called *Memory-mapped I/O.*
- In Memory-mapped I/O portions of address space are assigned to I/O devices and reads and writes to those addresses are interpreted as commands to the I/O device.
- —DATAIN‖ is the address of the input buffer associated with the keyboard.
- - Move DATAIN, R0
- Reads the data from DATAIN and stores them into processor register R0;
- - Move R0, DATAOUT
- Sends the contents of register R0 to location DATAOUT.
- Option of special I/O address space or incorporate as a part of memory address space (address bus is same always).

IO System

Accessing I/O Devices I/O interface

Input/output mechanism

–Memory-mapped I/O

–Programmed I/O

–Interrupts

–Direct Memory Access

Buses

–Synchronous Bus

–Asynchronous Bus

•IO in Computer Organization and Operating System:

1.Programmed I/O

2.Interrupts

3.DMA (Direct memory Access

IO System

Device	Behavior	Partner	Data rate (Mbit/sec)
Keyboard	input	human	0.0001
Mouse	input	human	0.0038
Voice input	input	human	0.2640
Sound input	input	machine	3.0000
Scanner	input	human	3.2000
Voice output	output	human	0.2640
Sound output	output	human	8.0000
Laser printer	output	human	3.2000
Graphics display	output	human	800.0000–8000.0000
Modem	input or output	machine	0.0160–0.0640
Network/LAN	input or output	machine	100.0000–1000.0000
Network/wireless LAN	input or output	machine	11.0000–54.0000
Optical disk	storage	machine	80.0000
Magnetic tape	storage	machine	32.0000
Magnetic disk	storage	machine	240.0000–2560.0000

IO System

• An IO interface consists of the circuitry required to connect output device to a computer system.

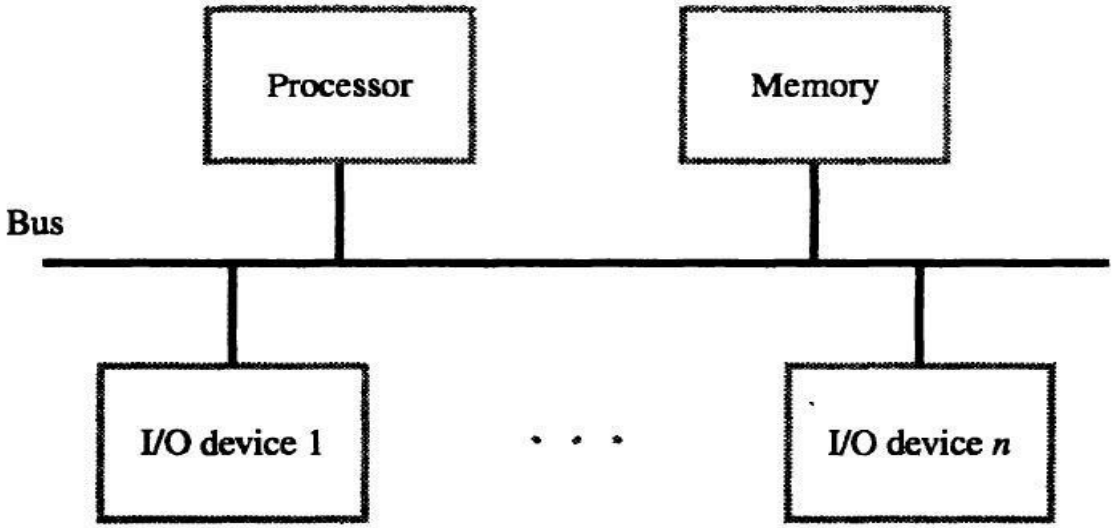


Figure – Bus Connection IO devices to Processor and memory.

IO System

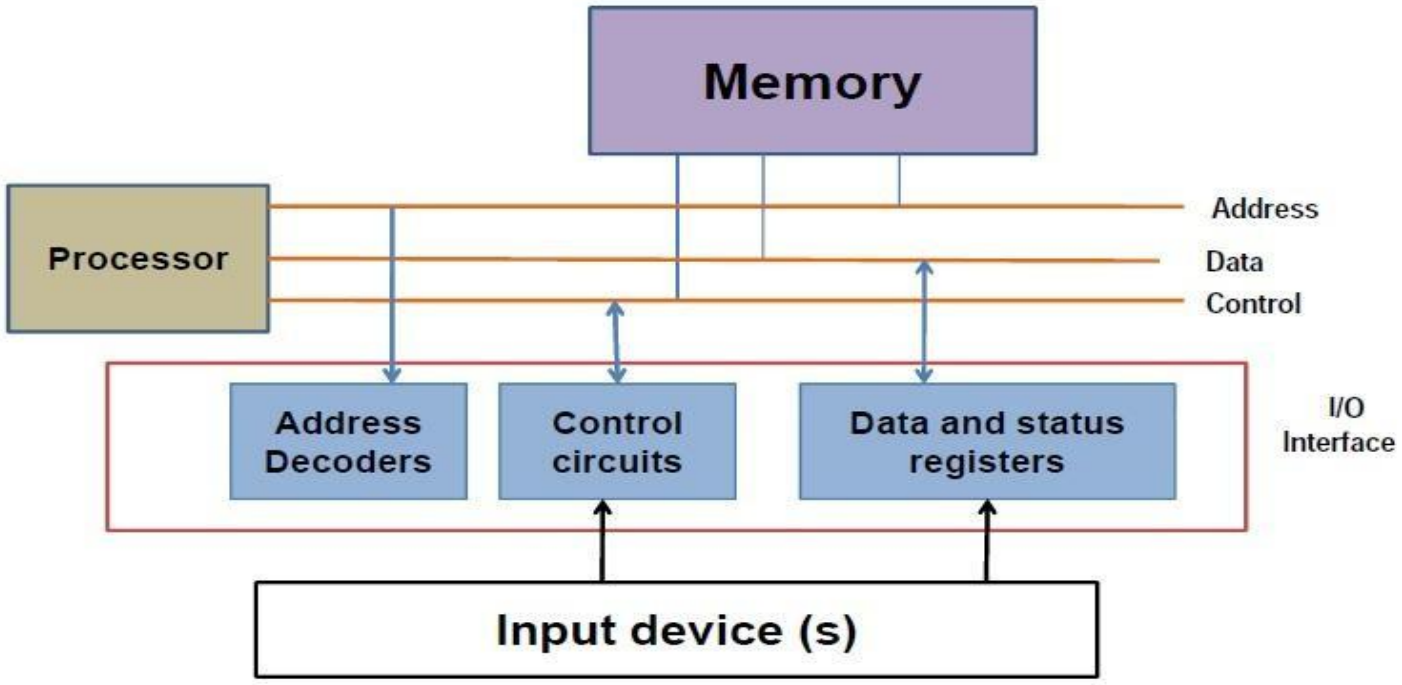


Figure – IO interface for an input device:

Interface Circuits

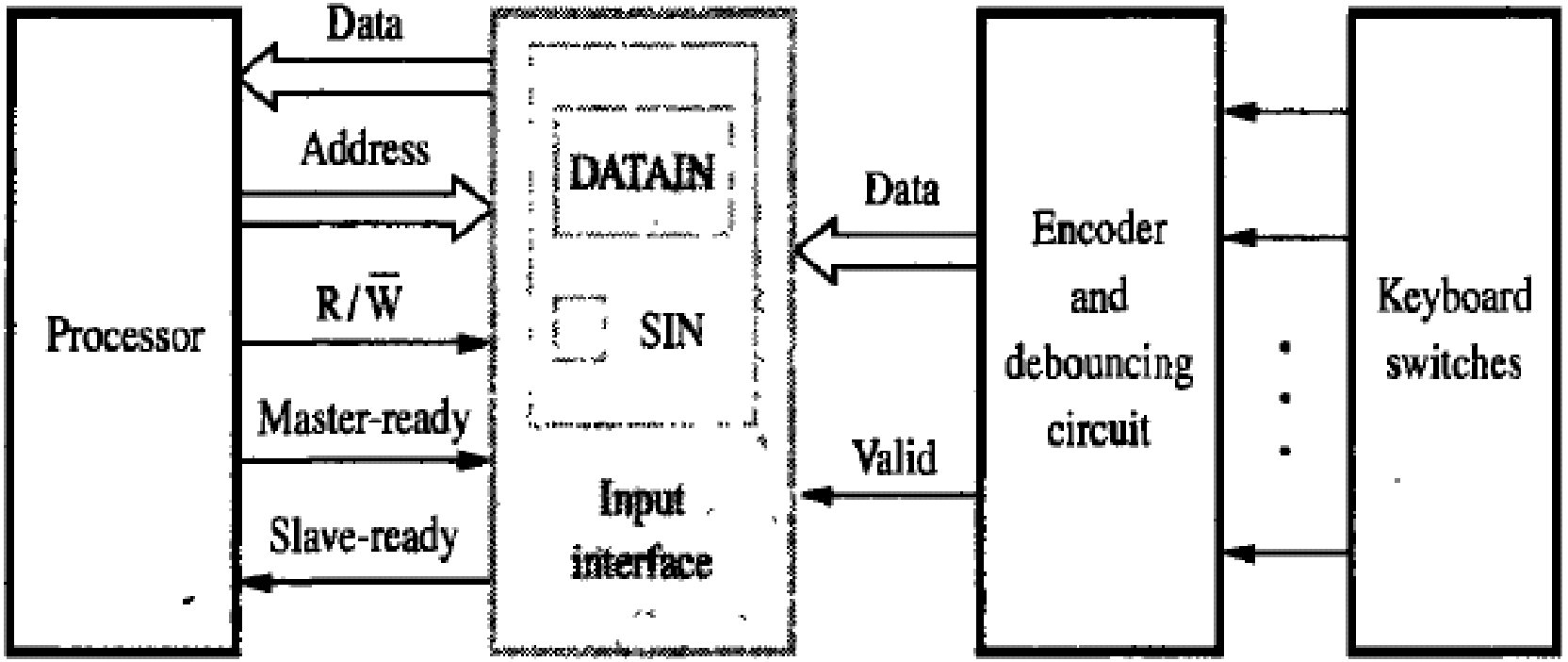


Figure 4.28 Keyboard to processor connection.

Serialport

1. It transmits and receives data one bit at a time.
2. For long distance, it is convenient and cost effective.
3. It is used to transmit/ receive data serially. i.e one at a time.
4. A key feature of an interface circuit in serial port is that it is capable of communicating in a bit-serial on the device side and in a bit-parallel on the bus side.

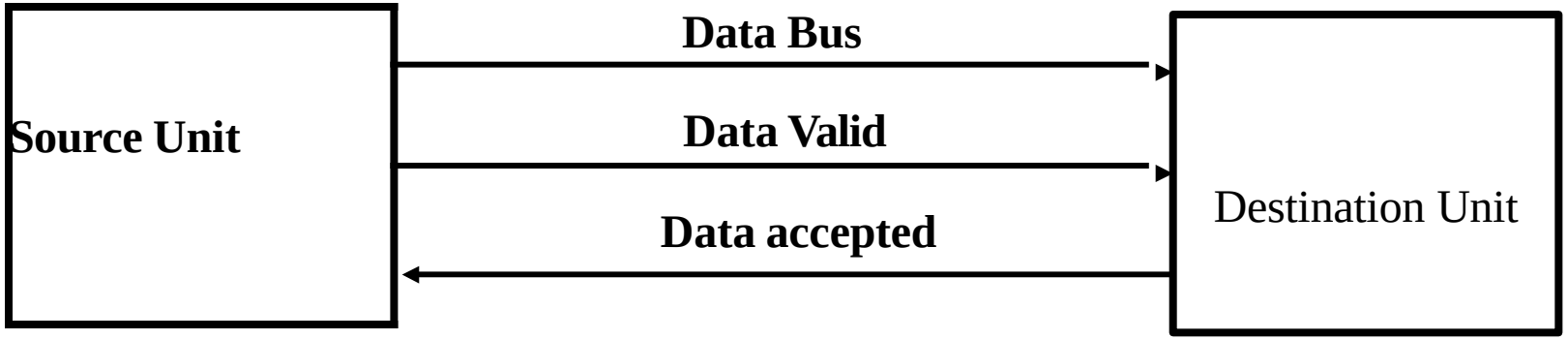
Interface Circuits

- An I/O interface consists of the circuitry required to connect an input/ output device to a computer system.
- On one side of the interface have bus signals for address, data and control.
- On the other hand data path with its associated controls to transfer data between the interface and the input/ output device.
- This side is called a port. It can be classified as
  - **Serial port**
  - **Parallel port**

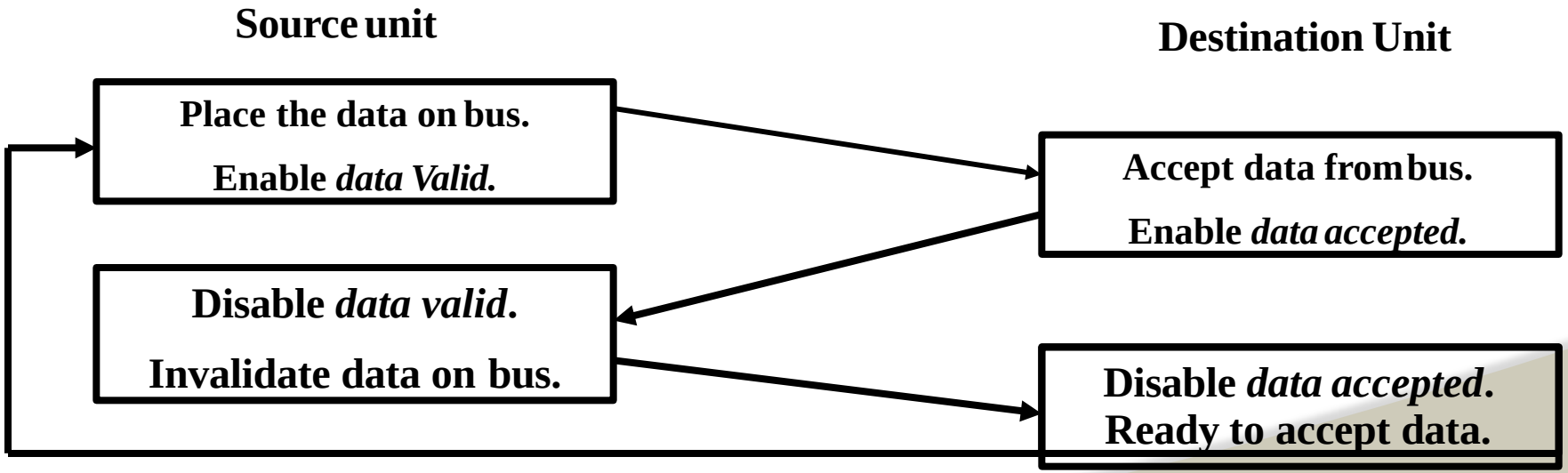
Source Initiated Transfer using Handshaking

The sequence of events shows four possible states that the system can be at any given time. The source unit initiates the transfer by placing the data on the bus and enabling its ***data valid*** signal. The ***data accepted*** signal is activated by the destination unit after it accepts the data from the bus. The source unit then disables its ***data accepted*** signal and the system goes into its initial state.

Source Initiated Transfer using Handshaking



(a) Block Diagram



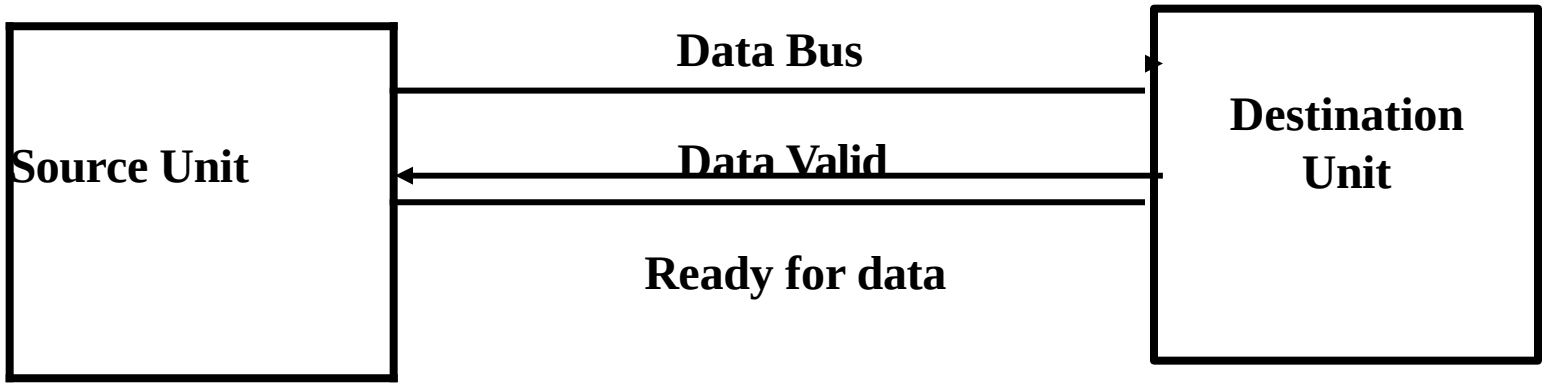
(b) Sequence of events

Destination Initiated Transfer Using Handshaking

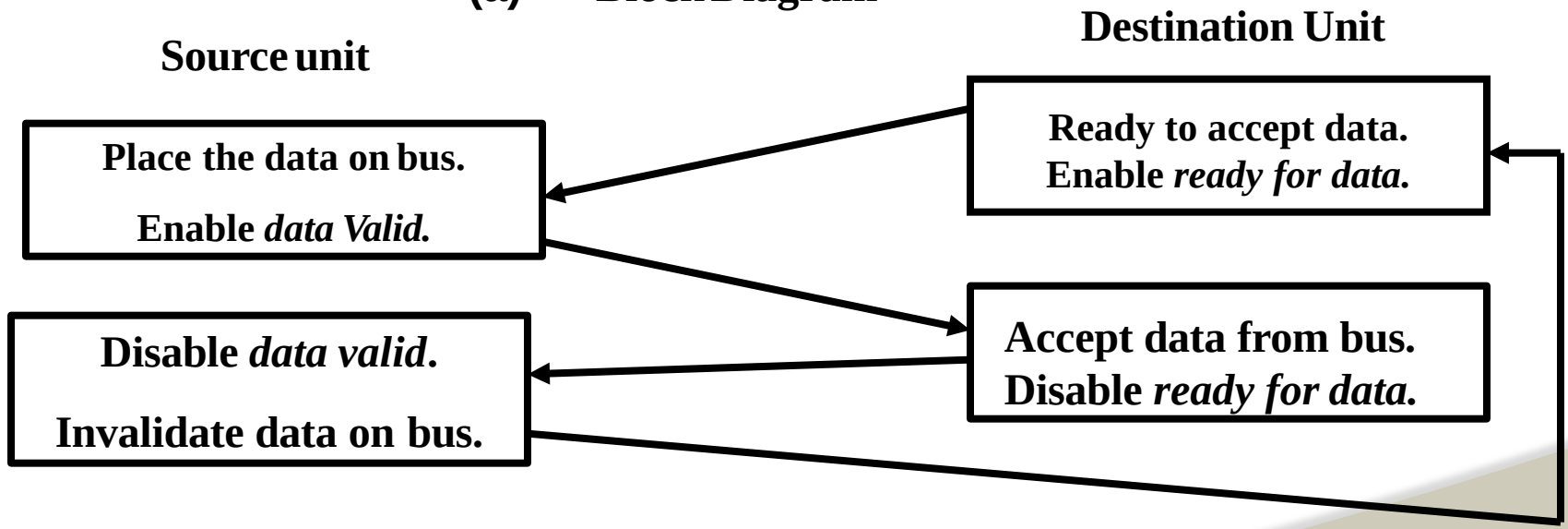
The name of the signal generated by the destination unit has been changed to ***ready for data*** to reflects its new meaning. The source unit in this case does not place data on the bus until after it receives the ***ready for data*** signal from the destination unit. From there on, the handshaking procedure follows the same pattern as in the source initiated case.

The only **difference** betweenthe Source Initiated and the Destination Initiated transfer is in their choice of Initial sate.

Destination Initiated Transfer Using Handshaking



(a) Block Diagram



(b) Sequence of events

Destination-Initiated transfer using Handshaking

Advantage of the Handshaking method

The Handshaking scheme provides degree of flexibility and reliability because the successful completion of data transfer relies on active participation by both units.

If any of one unit is faulty, the data transfer will not be completed. Such an error can be detected by means of a ***Timeout mechanism*** which provides an alarm if the data is not completed within time.

Advantage of the Handshaking method

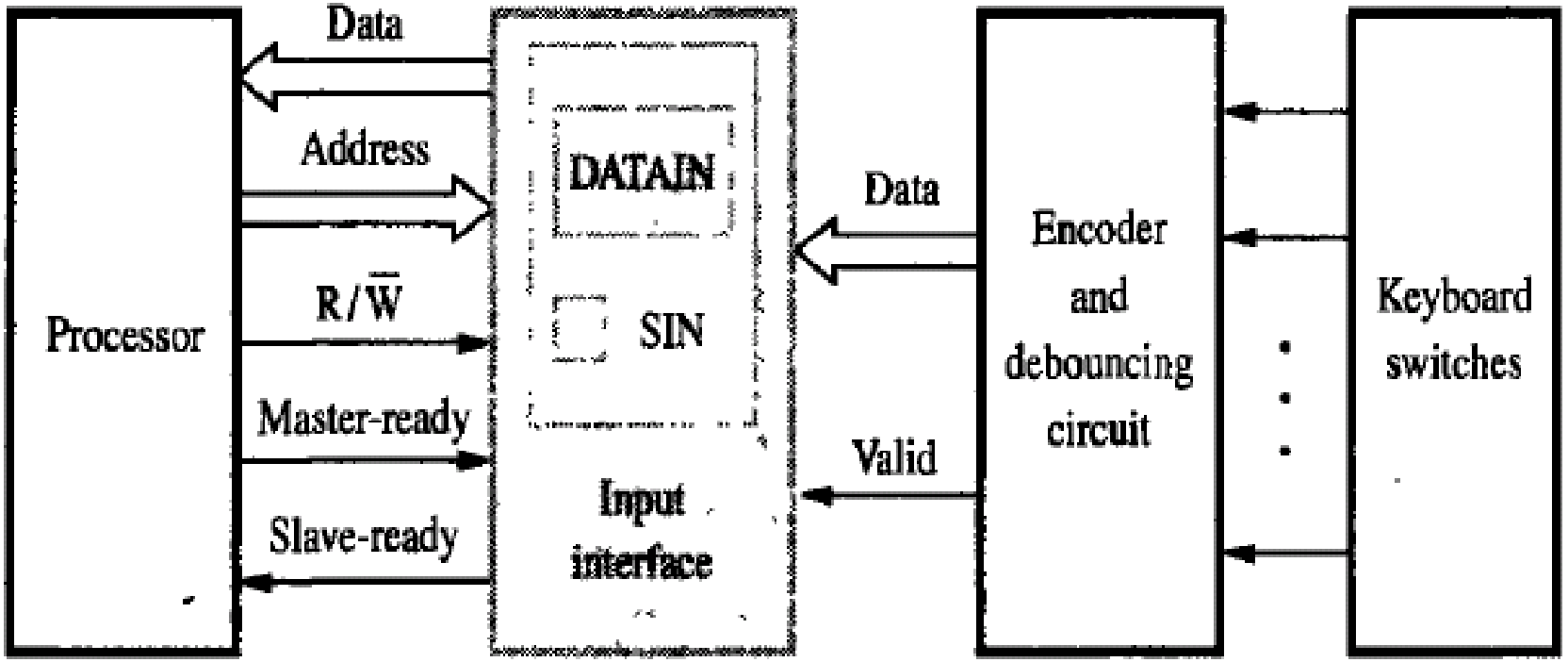


Figure 4.28 Keyboard to processor connection.

Parallel port

- It transfers data simultaneously to (or) from the device.
- It uses multiple pin connector.
- Circuit is simple.
- Parallel port is used to send (or) receive data having group of bits(8 bits or 16 bits) simultaneously.
- Parallel ports are classified as input port and output port.

  Input port – used to receive the data
- Output port- used to send the data.

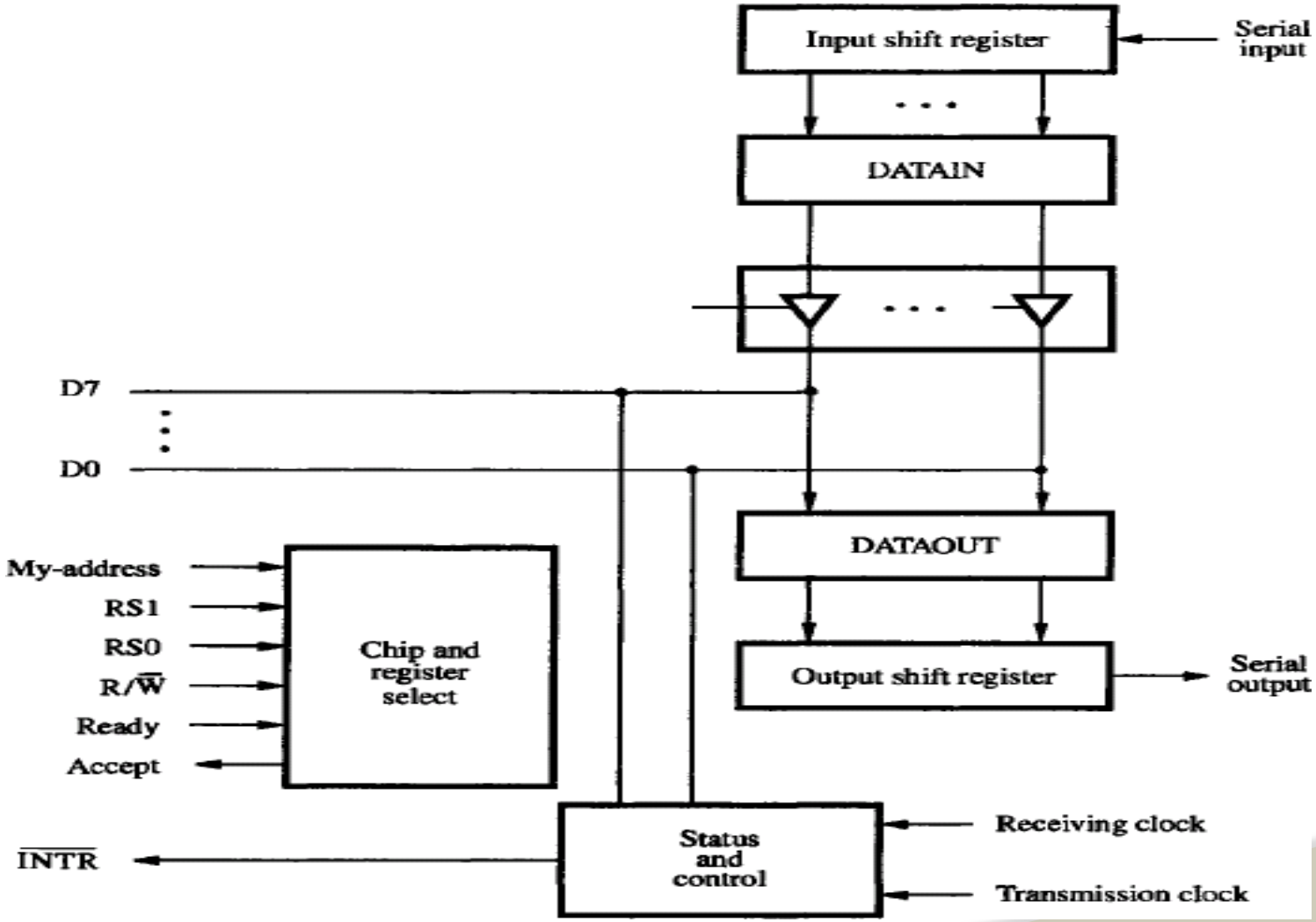
Serialport

1. It transmits and receives data one bit at a time.
2. For long distance, it is convenient and cost effective.
3. It is used to transmit/ receive data serially. i.e one at a time.
4. A key feature of an interface circuit in serial port is that it is capable of communicating in a bit-serial on the device side and in a bit-parallel on the bus side.

I/O Interface

1. It provides a storage buffer for atleast one word of data.
2. Contains status flags that can be accessed by the processor to determine whether the buffer is full (or) empty.
3. Address decoding circuitry to determine when it is being addressed by the processor.
4. Performs any format conversion that may be necessary to transfer data between the bus and the input/ output device such as parallel – serial conversion in the case of a serial port.

Output Port


Input shift register
Serial input
DATAIN
D7
D0
DATAOUT
My-address
RS1
RS0
R/W
Ready
Accept
Chip and register select
Output shift register
Serial output
Status and control
Receiving clock
Transmission clock
INTR