www.jntuworldupdates.org

UNIT - I

BASIC STRUCTURE OF COMPUTERS

Computer Types :-

Digital Computer :-

It is a fast electronic calculating machine which accepts digitized i/p information, processes to a list of internally stored instructions and produces the resulting o/p information.

The list of instructions is called a computer program and the internal storage is called computer memory. The computers may widely vary in size, cost, power

The most common computer is 'personal computer' which can be used in homes, schools, business, office. It is the most common form of desktop computers

Desktop computers have processing & storage units. Visual display, audio o/p units and a keyboard that can be located easily on a home.

In this, the storage media includes hard disks, CD-ROM's and diskettes.

Portable "notebook computers" is another version of pc, here all these components are packed into single unit, the size of a thin brief case.

"Work stations" with high-resolution graphics, i/o capability. It has the capability, performs more computational power than personal computers. This can be used in engineering applications, especially for interactive design work.

www.specworld.in

www.smartzmail.com
www.smartzworld.com

www.smartzmail.com
www.jntuworldupdates.org

A large & very powerful computer systems exist, that are called 'enterprise systems' and servers at the low end of the range and super computers at the high end.

Enterprise systems or mainframes are used for business data processing, that require much more computing power and storage.

Servers contain sizable data base, storage units and capable of handling large volumes of requests to access the data. Servers are widely used in education, business and personal user communities.

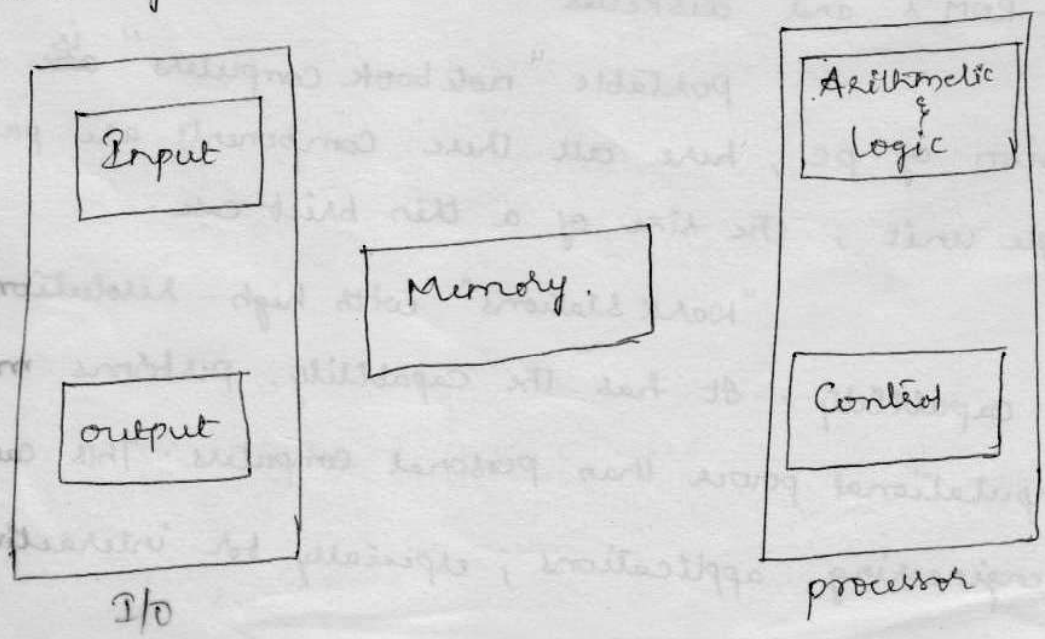
Super computers are used for the large-scale numerical calculations required in applications such as weather forecasting, air craft design and simulation.

Functional units :-

A computer consists of five functional independent main parts

1) I/P 2) Memory 3) ALU 4) O/P 5) Control units.

The below fig. shows



www.smartzmail.com
www.jntuworldupdates.org

The input unit accepts coded information from human operators, from electro mechanical devices such as keyboards or from other computers. The information received is either stored in memory or immediately used by the arithmetic & logic circuitary to perform desired operations. The processing steps are determined by a program stored in the memory. Finally, the results are sent back to the outside world through the o/p unit.

The arithmetic & logic circuits are placed in the processor & the input & output equipment is often collectively referred to as the input - output (i/o) unit.

A list of instructions that performs a task is called a 'program'. The program is stored in the memory. The processor fetches the instructions from the memory & perform the desired operations. The computer is completely controlled by the stored program, except for possible external interruption by an operator or by I/o devices connected to the machine.

Data are numbers & encoded characters that are used as operands by the instructions. The term data is often used to any digital information. The task of compiling a HLL source program into a list of machine instructions constituting a machine language program, called the object program.

The source program is the i/p data to the compiler program which translates the source program into a machine language program.

www.specworld.in
www.smartzmail.com

www.smartzworld.com

www.smartzmail.com
www.jntuworldupdates.org

Information handled by a computer must be encoded in a suitable format. The present h/w implements the digital circuits that have only two stable states i.e, ON & OFF. Each number, character or instruction is encoded as a string of binary digits called 'bits', each having one of two possible values 0 or 1.

Alphanumeric characters are also expressed interms of binary codes. Several binary codes have been developed. Two of most widely used schemes are ASCII (American Standard code for Information Interchange), each character is represent as a 7-bit code and EBCDIC (Extended Binary-Coded Decimal Interchange code), each character is represent as a 8-bit code.

Input unit :-

Computers accept coded information through i/p units; which read the data. The most well-known i/p device is the keyboard. when ever a key is pressed, the corresponding letter or digit is automatically translated into its corresponding binary code & that code is transferred to either memory or the processor.

Memory unit :-

The function of the memory unit is to store programs & data. There are two classes of storage, called primary & secondary storage.

primary storage is a fast memory that operates at electronic speeds. programs must be stored in the memory while they are being executed. The memory contains a large number of semi conductor storage cells, each capable of

www.specworld.in
www.smartzmail.com
www.smartzworld.com

www.smartzmail.com
www.jntuworldupdates.org

primary storage is a fast memory that operates at electronic speeds. programs must be stored in the memory while they are being executed. The memory contains a large no. of semiconductor storage cells, each capable of storing one bit of information. These cells are rarely read or written as individual cells but using a words. The memory is organized so that the contents of one word, containing 'n' bits, can be stored or retrieved in one basic operation.

To provide easy access to any word in the memory, a distinct address is associated with each word location. A given word is accessed by specifying its address & issuing a control command that starts the storage or retrieval process.

The no. of bits in each word is referred to as the 'Word length'. Typical word length range from 16 to 64 bits. The capacity of the memory is depends upon the size of the computer

programs must reside in the memory during execution. Instructions & data can be written into the memory or read from the memory.

Memory in which any location can be reached in a short & fixed amount of time after specifying its address is called Random-access memory (RAM). The time required to access one word is called the memory access time.

The memory of a computer is normally implemented as a memory hierarchy of three or four levels of semiconductor RAM units with different speeds, sizes. The small and fastest RAM units is cache memory.

The largest & slowest unit is referred to as the main memory.

With all these, the secondary storage is used

www.specworld.in
www.smartzmail.com
www.smartzworld.com

www.smartzmail.com
www.jntuworldupdates.org

When large amounts of data & many programs have to be stored. The typical devices are magnetic disks, magnetic tapes & optical disks.

Arithmetic & logic unit :-

The basic operations are performed in it.

for eg, two numbers located in the memory to be added. They are brought into the processor & the actual addition is carried by the ALU. The sum may be stored in the memory or retained in the processor for immediate use.

When operands are brought into the processor, they are stored in high-speed storage elements called 'registers'.

Access times to registers are some what faster than access times to the fastest cache unit in the memory hierarchy.

Output unit :-

Its function is to send processor results to the outside world (or user). The most familiar device is printer.

Some units, such as graphic displays, provide an both an o/p function & an i/p function.

Control unit :-

The memory, arithmetic & logic, i/p & o/p units store and process information & perform i/o operations. The control unit is effectively the nerve center that sends control signals to other units & senses their states.

The actual transfers are generated in the control circuits by 'timing signals'. Timing signals are the signals that determine when a given action is to takes place.

The operation of a computer can be summarized as follows.

www.specworld.in
www.smartzmail.com

www.smartzworld.com

www.jntuworldupdates.org

Basic operational Concepts:

for eg, consider the following instruction,

Add LOCA, R0

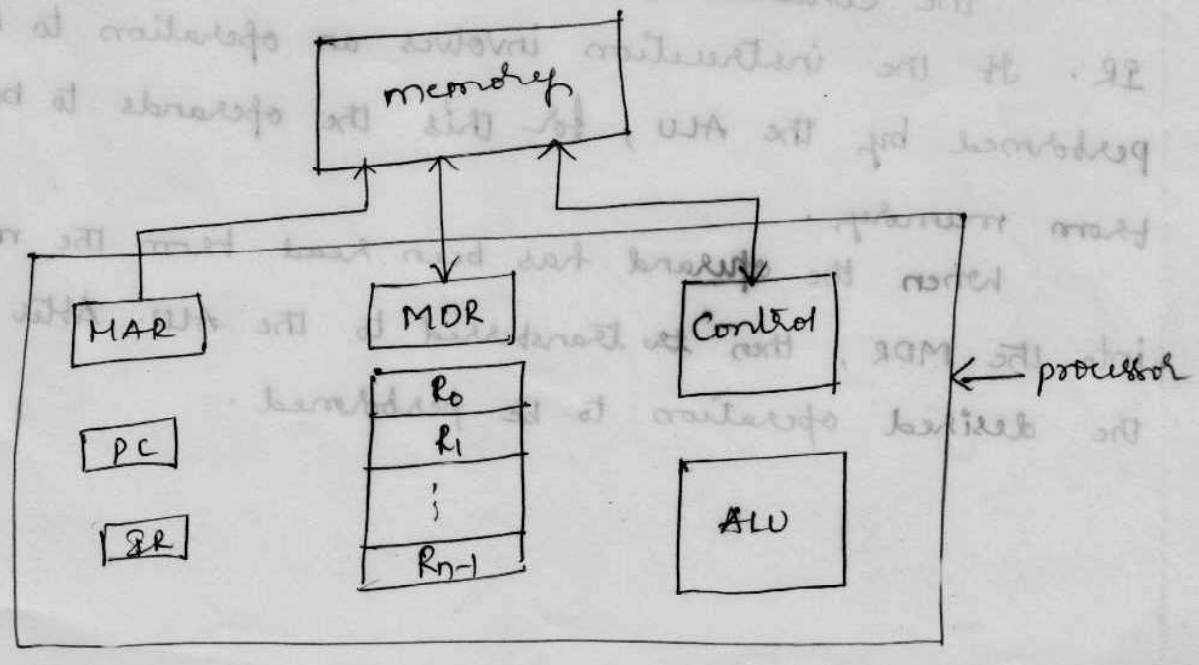
In this, adds the operand at memory 'LOCA' to the operand in a register R0, which is present in processor and here the sum is stored in the R0, which is overwritten.

The above instruction performs the following steps

1) The instruction is fetched from the memory into processor
2) The operand at LOCA is fetched and added to the contents of R0.
3) The resulting sum is stored in Register R0.

Transfers between the memory and the processor are started by sending the address of the memory location to be accessed to the memory unit and issuing the appropriate control signals.

The below fig. shows. how the transfers can be done between the memory and the processor.



www.jntuworldupdates.org

(IR) Instruction Register :-

It holds the instruction that is currently being executed.

Program Counter :-

It contains the address of the instruction that is to be fetched.

Memory address Register (MAR) :-

It holds the address which is to be accessed.

Memory Data Register (MDR) :-

It contains the data to be written into or read out of the addressed location.

Execution of the program starts when the PC is set to point to the first instruction of the program. The contents of the PC are transferred to the MAR and a 'Read' control signal is sent to the memory. From that, the word is read out of the memory and loaded into the MDR.

The contents of the MDR are transferred to the IR. If the instruction involves an operation to be performed by the ALU, for this the operands to be fetched from memory.

When the operand has been read from the memory into the MDR, then it transferred to the ALU. After that, the desired operation to be performed.

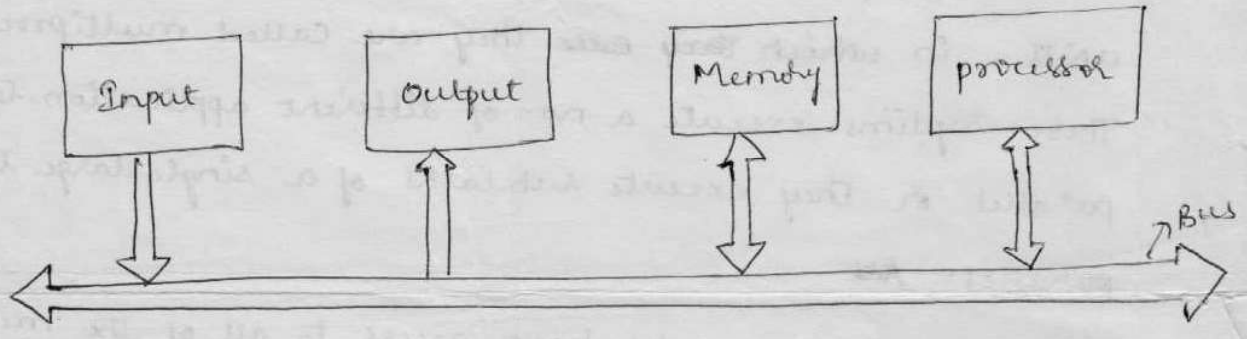
www.smartzmail.com
www.jntuworldupdates.org

Bus Structures :-

When a word of data is transferred between units, all its bits are transferred in parallel i.e., the bits are transferred simultaneously over many wires or lines, one bit per line. A group of lines that serves as a connecting path for several devices is called a "bus".

The lines that carry the data, address & control purposes.

The simplest way to interconnect functional units is to use a single bus. This can be shown in below fig.



All units are connected to the bus. Because the bus can be used for only one transfer at a time, only two units can actively use the bus at any given time. Bus control lines are used to arbitrate multiple requests for use of the bus.

The main thing of single-bus structure is its low cost & its flexibility for attaching peripheral devices.

Systems that contain multiple buses achieve more concurrency in operations by allowing two or more transfers to be carried out at the same time. This leads to better performance but at an increased cost.

Memory & processor units operate at electronic speeds, making them the fastest parts of a computer. Because all these devices must communicate with each other over a bus, an efficient transfer mechanism that is not constrained by the slow devices & that can be used to smooth out the differences in

www.specworld.in
www.smartzworld.com

www.jntuworldupdates.org

- The computer accepts information in the form of programs & data through an i/p unit & stores it in the memory.
- Information stored in the memory is fetched, under program control, into an arithmetic & logic unit, where its processed.
- processed information leaves the computer through an o/p unit
- All activities inside the machine are directed by the control unit.

Multi processors & multi computers :-

Large computer systems may contain a no. of processor units, in which they are called multiprocessor systems. These systems execute a no. of different application tasks in parallel or they execute subtasks of a single large task in parallel.

All processors have access to all of the memory in such systems, & the term shared-memory multiprocessors systems. The high performance of these systems comes with much increased complexity & cost. In addition to multiple processors and memory units, cost is increased because of the need for more complex interconnection n/w's.

In contrast to this, it is also possible to use an interconnected group of complete computers to achieve high total computational power. The computers normally have access only to their own memory units. When the tasks they are executing need to communicate data, so that they by exchanging messages over a communication n/w.

www.jntuworldupdates.org

timing among processors, memories & external devices is necessary.

A common approach is to include buffer registers with the devices to hold the information during transfers.

Eg: Consider the transfer of an encoded character from a processor to a character printer. The processor sends the character over the bus to the printer buffer. Since the buffer is an electronic register, this transfer requires relatively little time. Once the buffer is loaded, the printer can start printing. The bus & the processor are no longer needed & can be released for other activity. The printer continues printing the character in its buffer & is n't available for further transfers until this process is completed. Thus, buffer registers carry timing differences among processors, memories & I/O devices.

Software :-

System software is a collection of programs that are executed as needed to perform functions such as,

- Receiving & interpreting user commands.
- Entering & editing application programs & storing them as files in secondary storage devices.
- Managing the storage & retrieval of files in secondary storage devices.
- Running standard application programs such as word processors, spread sheets or games, with data supplied by the user.
- Controlling I/O units to receive i/p information & produce o/p results.
- Translating programs from source form to object form
- Linking & running user-written application programs.

www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org

Multiple Bus hierarchies:

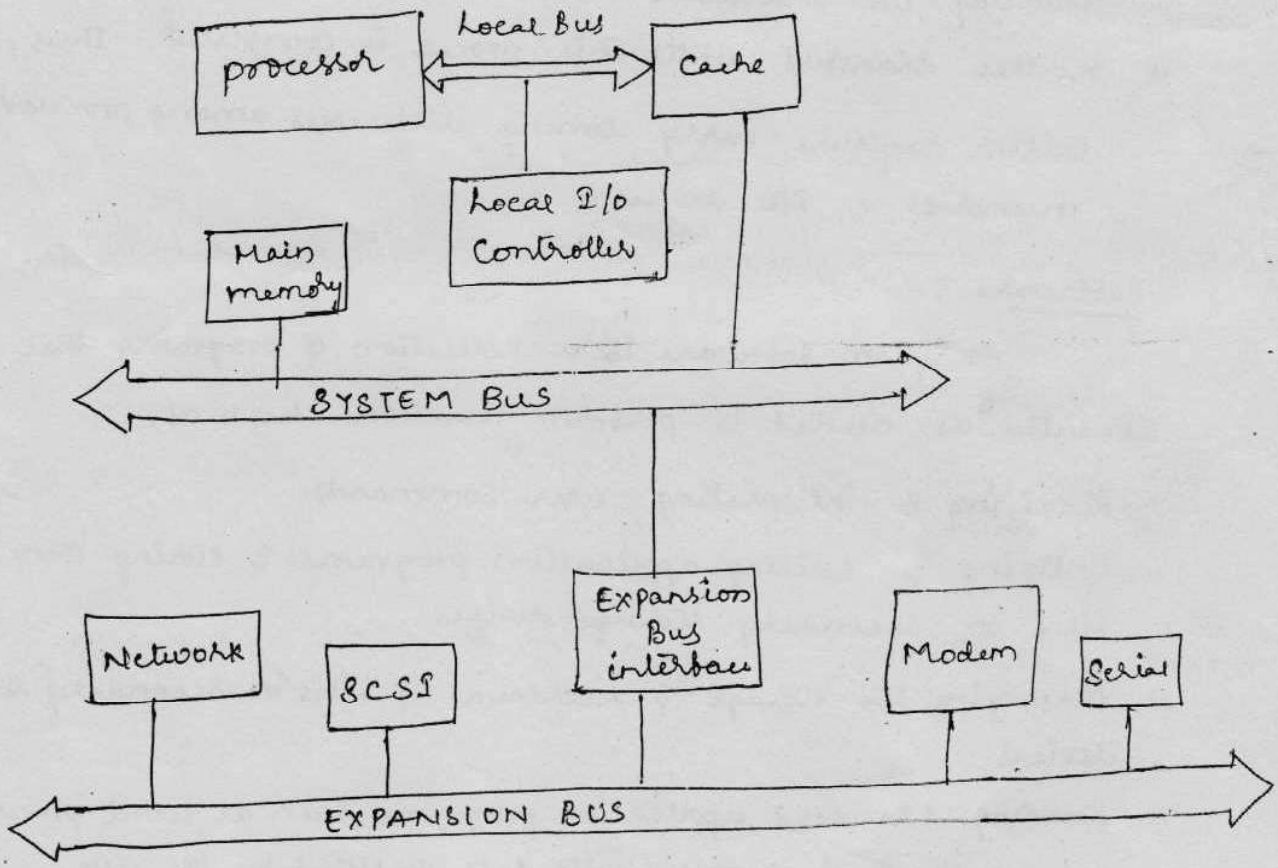
There are two different types

1) Traditional - Bus architecture

2) High - performance architecture.

Traditional - Bus architecture :-

The below fig. shows the architecture of the traditional - Bus.



There is a local bus that connects the processor to a cache memory and that may support one or more local devices. The cache memory controller connects to the system bus also, which is attached to all the main memory modules.

www.jntuworldupdates.org

It is possible to connect I/O controllers directly onto the system bus. The most efficient solution is to make use of one or more expansion buses.

An expansion bus interface buffers data transfers between the system bus and the I/O controllers on the expansion bus. This arrangement allows the system to support a wide variety of I/O devices and at the same time insulate memory-to-processor traffic from I/O traffic.

Let us see, ~~the~~ some of the devices which is attached to the expansion bus.

Network connections include Local Area networks & Wide Area networks.

SCSI (Small Computer System Interface) is itself a type of bus used to support local disk drives & other peripherals.

A serial port could be used to support a printer or scanner.

This architecture is efficient, but begins to break down as higher and higher performance is seen in the I/O devices.

In addition to, growing demands, a common approach taken by industry is to build a high-speed bus, requiring a bridge between the processor's bus & the high-speed bus.

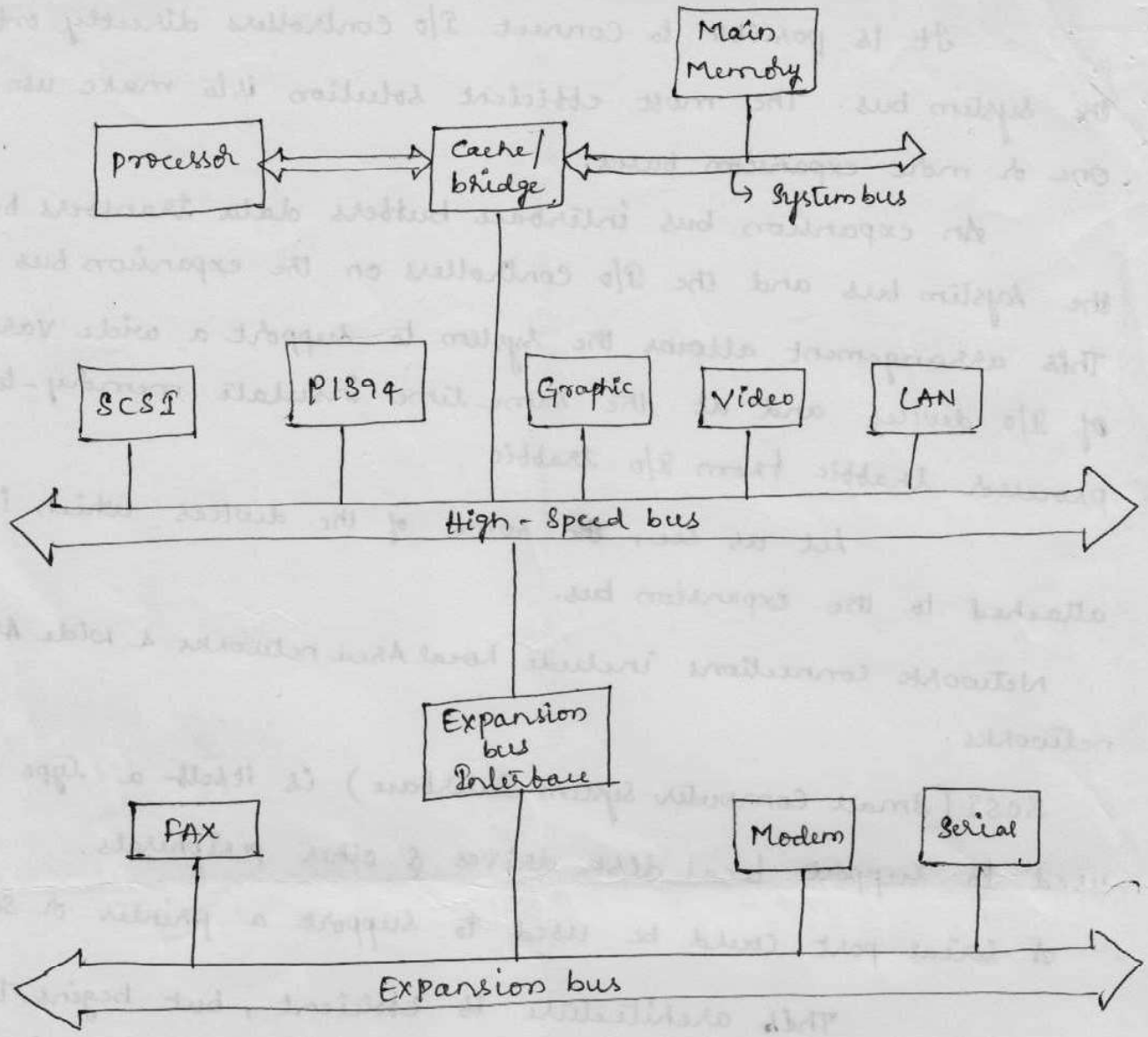
This arrangement is sometimes called mezzanine architecture.

The fig. shows the architecture of high-performance architecture.

www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org



www.jntuworldupdates.org

performance :-

The most important measure of the computer is performance i.e., how quickly it can execute programs.

Let us see, some of the performance measures.

processor clock :-

processor circuits are controlled by a 'timing signal' called a 'clock'. The clock defines regular time intervals called 'clock cycles'.

To execute a machine instruction, the processor divides the action to be performed into a sequence of basic step, such that each step can be completed in one clock cycle.

Basic performance equation :-

$$T = \frac{N \times S}{R}$$

N = No. of actual machine instructions

S = steps to be taken

R = clock rate per second.

pipelining and superscalar operation :-

The improvement of a performance can be achieved by overlapping the execution of successive instructions called 'pipelining'.

When multiple no. of instructions are to be executed, by creating parallel paths, i.e., the several instructions can be executed in every clock cycle. This mode of operation is called 'superscalar' operation.

www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org

Clock Rate :-

There are two possibilities for increasing the clock rate.

1) Improving the Integrated - circuit technology to work in a fast way.

2) Reducing the amount of processing done in one basic step.

Instruction set : CISC & RISC

CISC - Complex instruction set Computer
- which involves a complex instructions i.e, it ~~large~~ involves a large steps

RISC - Reduced instruction set Computer
- which involves a simple instructions i.e, it requires a minimum no. of steps.

Compiler :-

A Compiler translates a high-level language program into a sequence of machine instructions. So, ~~to~~ we need to have a suitable machine instruction set, then the Compiler will provide good performance.

performance Measurement :-

The idea of measuring Computer performance using benchmark programs. A non-profit organization called System performance Evaluation Corporation (SPEC) selects & publishes representative application programs for different application domains.

www.jntuworldupdates.org

Data Representation :- UNIT-I

Data Types :-

The data types found in the ~~categories~~ registers of digital computers may be classified as

(i) numbers used in arithmetic computations

(ii) letters of the alphabet used in data processing

(iii) other discrete symbol used for specific purposes.

We are representing in binary form because registers are made up of flip-flops and flip-flops are two-state devices that can store only 0's & 1's.

Number Systems :-

Radix :- A number system of base or radix, r is a system that uses distinct symbols for 'r' digits. It is necessary to multiply each digit by an integer power of 'r' and then form the sum of all weighted digits.

Decimal :- eg: The decimal system number uses the radix 10 system. The 10 symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

The digit 824.5 is represented as

$$8\times10^2 + 2\times10^1 + 4\times10^0 + 5\times10^{-1}$$

binary :- The binary number system uses the radix 2. The two digits will be 0 & 1. The string of digits 101101 is interpreted as

$$1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$$
$$= 45$$

www.jntuworldupdates.org

Octal & hexadecimal:

The octal (radix 8) and hexadecimal (radix 16). The eight symbols are 0, 1, 2, 3, 4, 5, 6, 7. The sixteen symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. The symbols A, B, C, D, E, F correspond to the decimal numbers 10, 11, 12, 13, 14, 15 respectively.

A number in radix 'r' can be converted to the familiar decimal system by forming the sum of the weighted digits.

$(101)_2 = 1\times2^2 + 0\times2^1 + 1\times2^0$
$= 4+1 = 5$

Eg: Octal 736.4 to decimal

$$(736.4)_8 = 7\times8^2 + 3\times8^1 + 6\times8^0 + 4\times8^{-1}$$
$$= 7\times64 + 3\times8 + 6\times1 + \frac{4}{8} = (478.5)_{10}.$$

The decimal number of hexadecimal F3 is

$$(F3)_{16} = F\times16^1 + 3\times16^0 = (15\times16)+3 = (243)_{10}.$$

Conversion:-

The conversion of decimal 41.6875 into binary is done by first separating the number into its integer part 41 and fraction part .6875. The integer part is converted by dividing 41 by r=2 to an integer quotient of 20 and a remainder of 1. The quotient is again divided by 2 to give a new quotient and remainder. This process is repeated until the integer quotient becomes 0. The first remainder will give the low-order bit of the converted binary number.

2 | 41
2 | 20 – 1
2 | 10 – 0
2 | 5 – 0
2 | 2 – 1
1 – 0

101001

www.jntuworldupdates.org

The fraction part is converted by multiplying it by $r=2$ to give an integer and a fraction. The new fraction is again multiplied by 2 to give a new integer and a new fraction. This process is repeated until the fraction part becomes to zero. Finally the two parts are combined to give the total required conversion.

Eg:

41

2 | 41
2 | 20 – 1
2 | 10 – 0
2 | 5 – 0
2 | 2 – 1
1 – 0

$(41)_{10} = (101001)_2$

Fraction = 0.6875

0.6875 ×2
1.3750 ×2
0.7500 ×2
1.5000 ×2
1.0000

$(0.6875)_{10} = (0.1011)_2$

$(41.6875)_{10} = (101001.1011)_2$

Octal and Hexadecimal numbers

The conversion from and to binary, octal & hexadecimal rep. plays an important role in digital computers. Since $2^3 = 8$ & $2^4 = 16$. each octal digit represents to three and each hexadecimal digit represent to 4 binary digits. The conversion from binary to octal is three digits. Conversion from binary to hexadecimal is from four digits.

Eg:

A	F	6	3		Hexadecimal
1010	1111	0110	0011		Binary
1 2	7	5	4	3	Octal

Octal number	Binary-coded	Decimal equivalent
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5		

www.jntuworldupdates.org

6	110	6
7	111	7
10	001 000	8
12	001 001	9
143	001 100 011	99

Hexadecimal number	Binary-coded	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15
14	0001 0100	20
F8	1111 1000	248.

<u>Decimal representation</u>:-

The binary number system is the most natural system for a computer, but people are accustomed to the decimal system. One way to solve this conflict is to convert all i/p decimal numbers into a binary number.

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
10	0001 0000
248	0010 0100 1000.

www.jntuworldupdates.org

Complements:

Complements are used in digital computers for simplifying the subtraction operation & for logical manipulation. There are two types of complements (1) r's complement (2) (r-1)'s complement. The 'r' value is substituted, the two types are referred to as the 2's & 1's complement for binary numbers & 10's & 9's complement for decimal numbers.

(r-1)'s complement

Given a number N in base 'r' having 'n' digits, the (r-1)'s complement of N is defined as $(r^n - 1) - N$.

For eg: $r = 10$, $r - 1 = 9$.

So the 9's complement is $(10^n - 1) - N$. Now, 10^n represents a number that consists of a single 1 followed by n 0's. $10^n - 1$ is a number represented by n 9's.

for eg: with $n = 4$, we have $10^4 = 10000$ and $10^4 - 1 = 9999$.

The 9's complement is then obtained by subtracting each digit from 9.

for eg: 9's complement of 546700, 12389

$999999 - 546700 = 453299$

$99999 - 12389 = 87610$

1's complement

For binary numbers, $r = 2$ & $r - 1 = 1$, so the 1's complement of N is $(2^n - 1) - N$ for eg: $n = 4$, $2^4 = (10000)_2$ & $2^4 - 1 = (1111)_2$

The 1's complement of a binary number is formed by

www.jntuworldupdates.org

changing 1's into 0's and 0's into 1's.

for eg, the 1's complement of 1011001 is 0100110

The (r-1)'s complement of octal or hexadecimal numbers are obtained by subtracting each digit from 7 or F (decimal 15) respectively.

r's complement:-

The r's complement of an n-digit number N in base is defined as $r^n - N$ for $N \neq 0$ and 0 for $N = 0$. Comparing with the (r-1)'s complement, we note that the r's complement is obtained by adding 1 to the (r-1)'s complement. Since

$r^n - N = [(r^n - 1) - N] + 1$.

Thus the 10's complement of the decimal 2389 is 7610 + 1 = 7611. is ~~by~~ obtained by adding 1 to the 9's complement.

$$\begin{array}{r} 9999 \\ 2389 \\ \hline 7610 \end{array}$$

The 2's complement of binary 101100 is 010011 + 1 = 010100.

Subtraction of unsigned numbers:-

- The direct method of subtraction uses the borrowing concept

The subtraction of two n-digit unsigned numbers M-N ($N \neq 0$) in base 'r' can be done as follows.

1. Add the minuend M to the r's complement of the subtrahend N. This performs $M + (r^n - N) = M - N + r^n$.
2. If $M \geq N$, the sum will produce an end carry r^n which is discarded, and what is left is the result M-N.

www.jntuworldupdates.org

3. If $M<N$, the sum doesn't produce an end carry and is equal to $r^n-(N-M)$, which is the r's complement of $(N-M)$.

for eg, $72532 - 13250 = 59282$

The 10's complement of 13250 is

$$99999 - 13250 = 86749 + 1 = 86750.$$

$M = 72532$

10's complement $N = 86750$

Sum $= 159282$

Discard end carry $10^5 = -100000$

Answer $= 59282$

for eg, $M<N$ i.e., $\overset{M}{13250} - \overset{N}{72532}$ produces negative 59282.

$M = 13250$

10's complement $N = 27468$

Sum $= 40718$ There is no end carry.

$$\begin{array}{r} \cancel{1000}\ 99999 \\ 40718 \\ \hline 59281 \\ +1 \\ \hline 59282 \end{array}$$

Answer is negative 59282 = 10's complement of 40718

Binary Numbers :-

$X = 1010100$ & $Y = 1000011$

$X = 1010100$

2's complement of $Y = 0111101$

Sum $= 10000001$ $X-Y$

Discard end carry $2^7 = -10000000$

0010001

www.jntuworldupdates.org

Fixed point representation :-

There are two ways, we are going to represent the position of the binary point in a register.

1) fixed position
2) floating point representation

Fixed point method assumes that the binary point is always fixed in one position.

The two positions most widely used are

(1) a binary point in the extreme left of the register to make the stored number a fraction.

(2) a binary point in the extreme right of the register to make the stored number as an integer.

Integer Representation :-

When an integer binary number is positive, the sign is represented by 0 and the magnitude by a positive binary number. When an integer binary number is negative, the sign is represented by '1' but the rest of the number may be represented in one of three possible ways.

1. Signed magnitude representation
2. Signed-1's Complement "
3. Signed-2's " "

The signed magnitude rep. of a negative number consists of the magnitude and a negative sign. Remaining 2, we are going to perform either 1's or 2's complement

www.jntuworldupdates.org

for eg +14 = 00001110 – 8 bit register

Left most bits are zero in 8 bit reg.

-14 can be represent in 3 ways.

The signed magnitude rep. of -14 is obtained from +14 by complementing only the sign bit.

The signed 1's complement rep. of -14 is obtained by complementing all ~~complementing~~ all the bits of +14, including the sign bit.

The signed 2's complement is obtained by taking the 2's complement of the positive number, including its sign bit.

Arithmetic Addition :-

The addition of two numbers in the signed-magnitude system follows the rules.

- If the signs are the same, we add the two magnitudes and give the sum the common sign.
- If the signs are different, we subtract the smaller magnitude from the larger and give the result the sign of the larger.

+6	00000110		-6	11111010
+13	00001101		+13	00001101
+19	00010011		+7	00000111

11111001
+1
11111010

www.jntuworldupdates.org

			-13+6
+6	00000110	-6	11111010
-13	11110011	-13	11110011
-7	11111001	-19	11101011

In each & every case, we are going to perform only addition.

If there is any negative value, the total result will be in the 2's complement value.

Arithmetic Subtraction :-

Subtraction of two signed binary numbers when negative numbers are in 2's complement form is very simple.

Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including the sign bit). A carry out of the sign bit position is discarded.

A subtraction operation can be changed to an addition operation if the sign of the subtrahend is changed. This can be demonstrated thro' the relationship.

$$(\pm A) - (+B) = (\pm A) + (-B)$$

$$(\pm A) - (-B) = (\pm A) + (+B)$$

But changing a positive number to a negative number is easily done by taking its 2's complement. The reverse is also true.

www.jntuworldupdates.org

for eg,

$(-6)-(-13) = +7$.

11111010 − 11110011 .

The subtraction is changed to addition by taking the 2's complement of the subtrahend (-13) to give (+13).

i.e., 11111010 + 00001101 = 100000111. By removing the end carry, we obtain the correct answer 00000111 (+7)

Overflow:

When two numbers of n digits each are added and the sum occupies n+1 digits, we say that an overflow occured.

An overflow is a problem in digital computers because the width of the registers is finite.

The detection of an overflow after the addition of two binary numbers depends on whether the numbers are considered to be unsigned or signed. When two unsigned numbers are added, an overflow is detected from the end carryout of the MSB. In case of signed numbers, the left most bits always represent the sign and negative numbers are in 2's complement. When two signed numbers are added, the sign bit is treated as part of the number and the end carry doesn't indicate an overflow.

An overflow doesn't occur ~~getter~~ if one number is +ve and another is -ve.

www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org

Eg:

+70	0 1000110	-70	1 0111010
+80	0 1010000	-80	1 0110000
+150	1 0010110	-150	0 1001010

carries 0 1

Since, the answer can't be accomodated with in 8-bits, we say that an overflow occurred.

Overflow detection :-

The overflow can be detected by observing the carry into the sign bit position & the carry out of the sign bit position. If these two carries are not equal, an overflow condition is produced.

If these two carries are applied to an ex-OR gate, an overflow will be detected when the o/p of the gate is equal to '1'.

Decimal Fixed-point representation :-

The representation of decimal numbers in registers is a function of the binary code used to represent a decimal digit. A 4-bit decimal code requires four flip-flops for each decimal digit. The representation of 4385 in BCD requires 16 flipflops i.e., four flipflops for each digit.

0100 0011 1000 0101

By representing numbers in decimal we are wasting a considerable amount of storage space

www.intuworldupdates.org

Floating - point representation :-

The floating - point representation of a number has two parts

(1) ~~The~~ It represents a signed, fixed - point number called mantissa.

(2) The second part designates the position of the decimal (or binary point) and is called the exponent.

The fixed - point mantissa may be a fraction or an integer.

Ex: the decimal number +6132.789

Fraction	Exponent
0.6132789	+04

The value of the exponent indicates that the actual position of the decimal point is four positions to the right of the indicated decimal point in the fraction.

The equivalent is $+0.6132789 \times 10^{+4}$

The floating - point is represented as,

$$m \times r^e$$

Only the mantissa m and the exponent e are physically represented in the register (including signs)

A floating - point binary number is represented in a similar manner except that it uses base 2.

Ex: +1001.11

Fraction	Exponent
01001110	000100

www.jntuworldupdates.org

The fraction has zero in the MSB that it indicates positive. The exponent has the equivalent binary number +4. The floating-point number is,

$$m \times 2^e = +(.1001110)_2 \times 2^{+4}.$$

Normalization :-

A floating point number is said to be normalized if the MSB of the mantissa is nonzero.

For eg, the decimal number 350 is normalized but 0035 is not.

For eg, the binary number 00011010 is not normalized because the MSB is zero.

The floating-point numbers are more complicated than the fixed-point numbers and their execution takes place longer & requires more complex hardware.

Error Detection codes :-

An error detection code is a binary code that detects digital errors during transmission. The detected errors can't be corrected but their presence is indicated.

parity bit :-

The most common error detection code used is the parity bit. A parity bit is an extra bit included with a binary message to make the total number of 1's either odd or even.

www.jntuworldupdates.org

The even-parity scheme has the disadvantage of having a bit combination of all 0's, while in the odd parity there is always one bit i.e, 1. The P(odd) is a complement of P(even).

During transfer of information from one location to another, the parity bit is handled as follows -

~~(The message, including the parity bit, is transmitted to its destination).~~

At the sending end, the message is applied to parity generator, where the required parity bit is generated. The message, including the parity bit, is transmitted to its destination. At the receiving end, all the incoming bits are applied to a parity checker that checks the proper parity adopted (odd or even). An error is detected if the checked parity doesn't confirm to the adopted parity. The parity method detects the presence of one, three or any odd number of errors. An even number of errors is n't detected.

Parity generator & checker n/w's are logic circuits constructed with X-OR functions. An odd function is a logic function whose value is binary 1 iff an odd no. of variables are equal to 1. According to this definition, the P(even) function is the ex-OR of x, y, z because it is equal to 1 when either one or all three of the variables are equal to 1. The P(odd) function is the complement of the P(even) function.

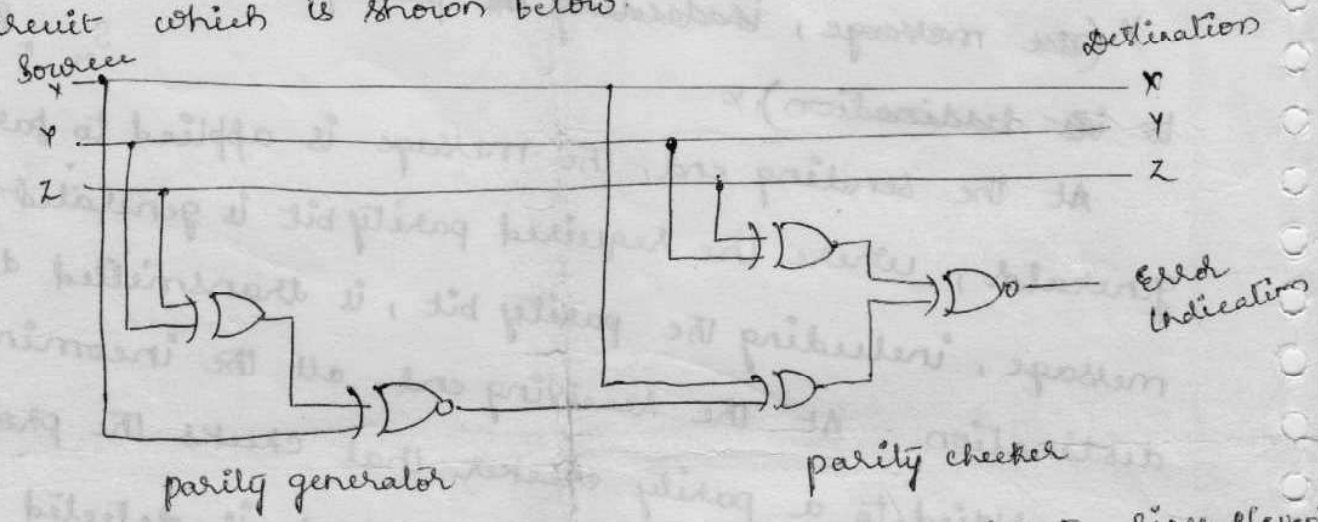
www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org

Message XYZ	P(odd)	P(even)
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1

Eg: a 3-bit message to be transmitted with an odd parity bit. At the sending end, the odd-parity bit is generated by a parity generator circuit which is shown below.



This circuit consists of one X-OR & one X-NOR gate. Since P(even) is the EX-OR of X, Y, Z & P(odd) is the complement of P(even).

The message & the odd-parity bits are transmitted to their destination where they applied to a parity checker. An error has occurred during transmission if the parity of the four bits received is even, since the binary information transmitted was originally odd.

The o/p of the parity checker would be 1 when an error occurs.

Addition and Subtraction with Signed 2's complement Data

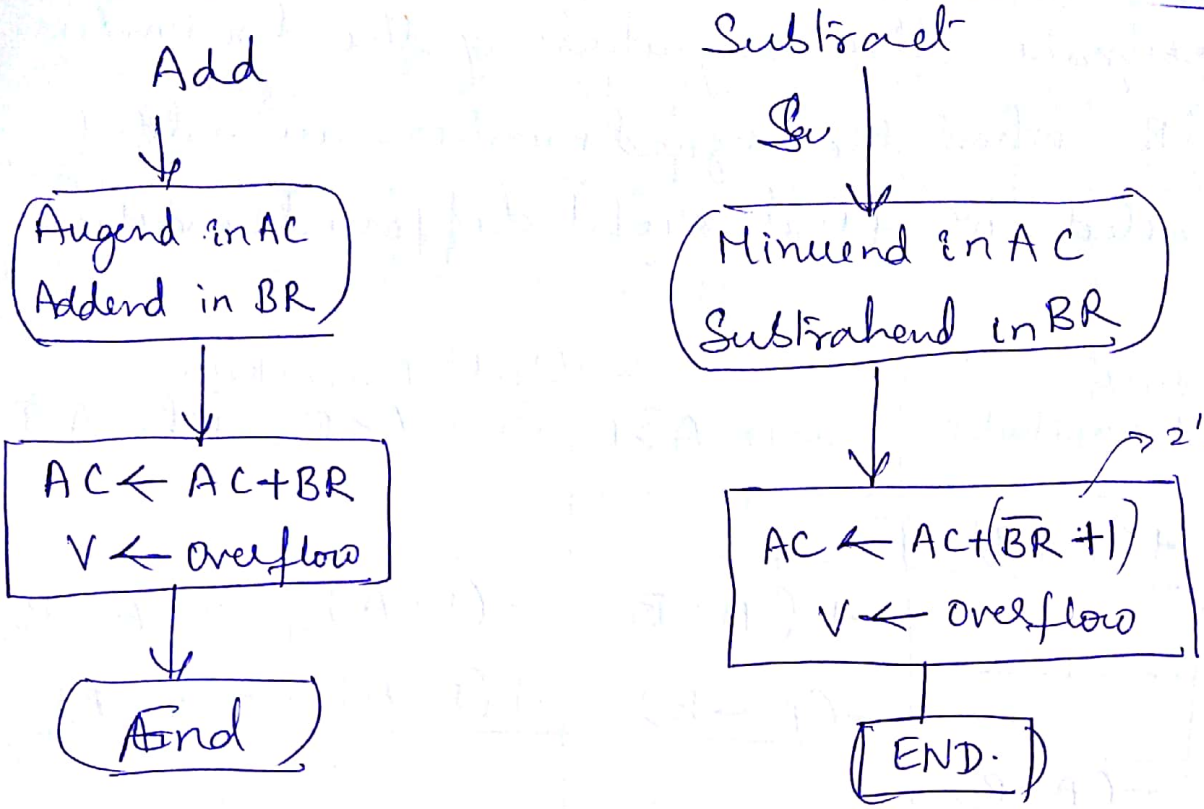


fig: flowchart for add & sub for 2's complement Data

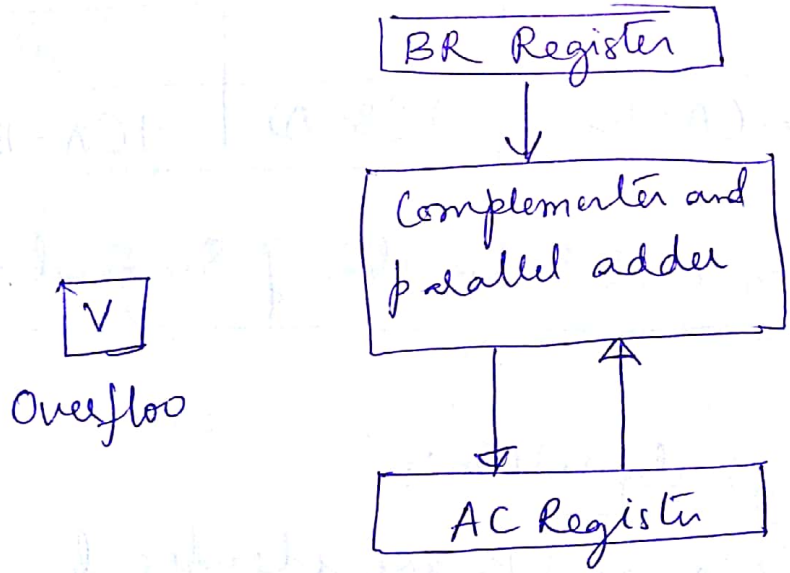


fig: Hardware for signed 2's complement addition and subtraction

Addition and Subtraction with Signed-Magnitude

Data :-

we designate the magnitude of the two numbers by A and B. when the signed numbers are added & subtracted, we find eight different conditions

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)

Algorithm for Addition and Subtraction of Singned-Magnitude data

Addition (Subtraction) algorithm :-

1. when the signs of A and B are identical (different), add the two magnitudes and attach the sign of A to the result.
2. when the signs of A and B are different (identical), compare the magnitudes and subtract the smaller no from the larger.

3. Choose the sign of the result to be the same as A if A>B or the complement of the sign of A if A<B. If the two magnitudes are equal, subtract B from A and make the sign of the result +ve.

Note: The words inside the parenthesis are used for subtraction algorithm.

Hardware Implementation

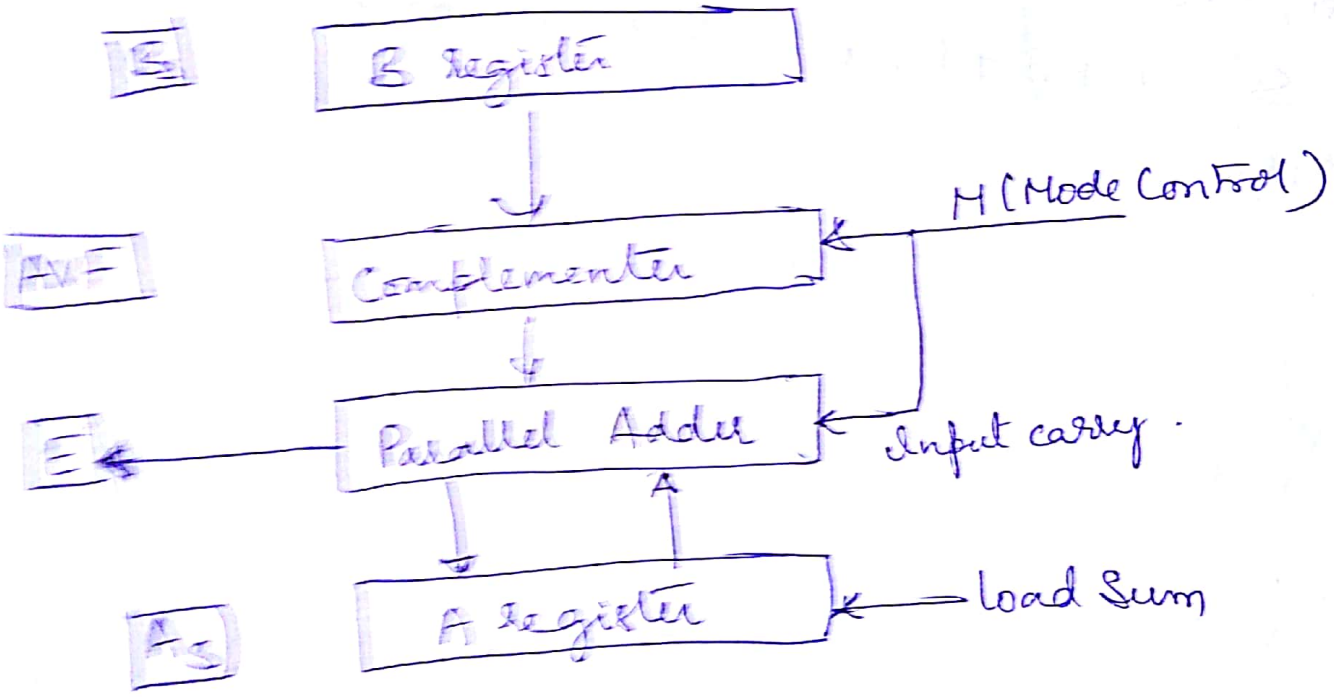


Fig: Hardware for signed-magnitude addition and subtraction:

→ It consists of two registers A and B and sign flip flop A_s and B_s. Subtraction is done by adding 2's complement of B.

The output carry is tranferred to flipflop 'E', where it can be checked to determine the relative magnitudes of the two numbers. The add - overflow flip flop AVF hold the overflow bit when A and B are added.

~~The A register pro~~ Mode control, M.

→ when M = 0, the output of B is transferred to the adder, the input carry is 0, and the o/p of adder is equal to A + B.

→ when M = 1, the 1's complement of B is applied to the adder, the input carry is 1 and output S = A + $\bar{B}$ + 1, which is equal to A − B.

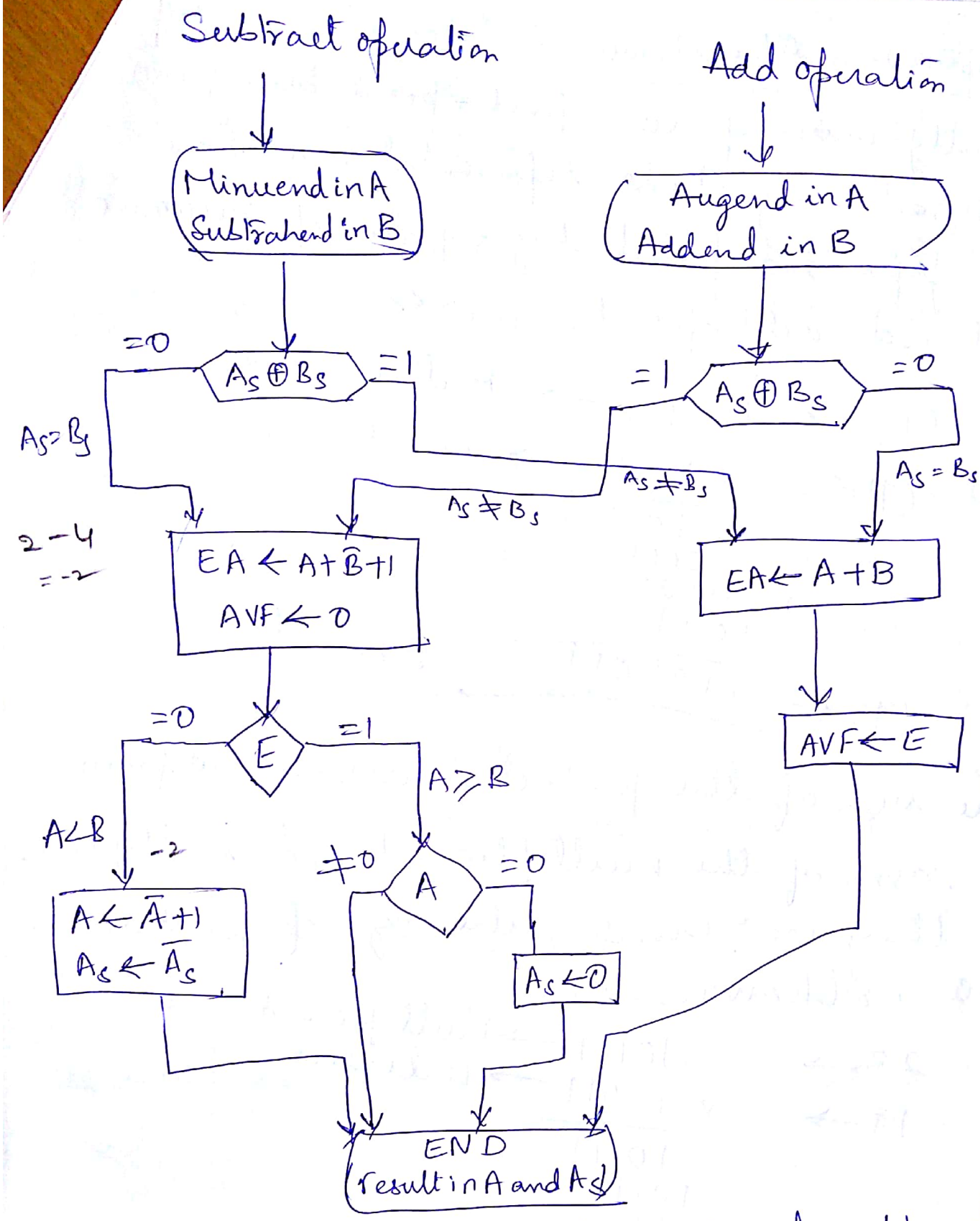


fig: flowchart for add and subtract.

→ Multiplication of two fixed-point binary nos. in signed-magnitude representation is done with paper and pencil by a process of successive shift and add operations. for ex

```
(11) ← 1011  → Multiplicand
(9) ←  × 1001 → multiplier.
       ------
        1011
       0000
      0000
     1011
     --------
(99) ← 1100011  → Product.
```

The sign of the product is determined from the signs of the multiplicand and multiplier. If they are same, the sign of product is +ve, otherwise -ve.

```
23 →      10111 → Multiplicand
19 →    × 10011 → Multiplier.
        -------
         10111
        10111
       00000
      00000
     10111
     ---------
437 → 110110101  → Product
```

when multiplication is implemented in digital computer, it is convenient to change the process slightly.

(1). Instead of providing registers to store and add simultaneously as many binary no as there are bits in the multiplier, it is convenient to provide an adder for the summation of only two binary no and successively accumulate the partial products in a register

(2). Instead of shifting the multiplicand to the left, the partial product is shifted to right. which results in leaving the partial product and multiplicand in the required relative position.

(3). when the corresponding bit of the multiplier is '0', no need to add all zeros to the partial product because it will not alter the value.

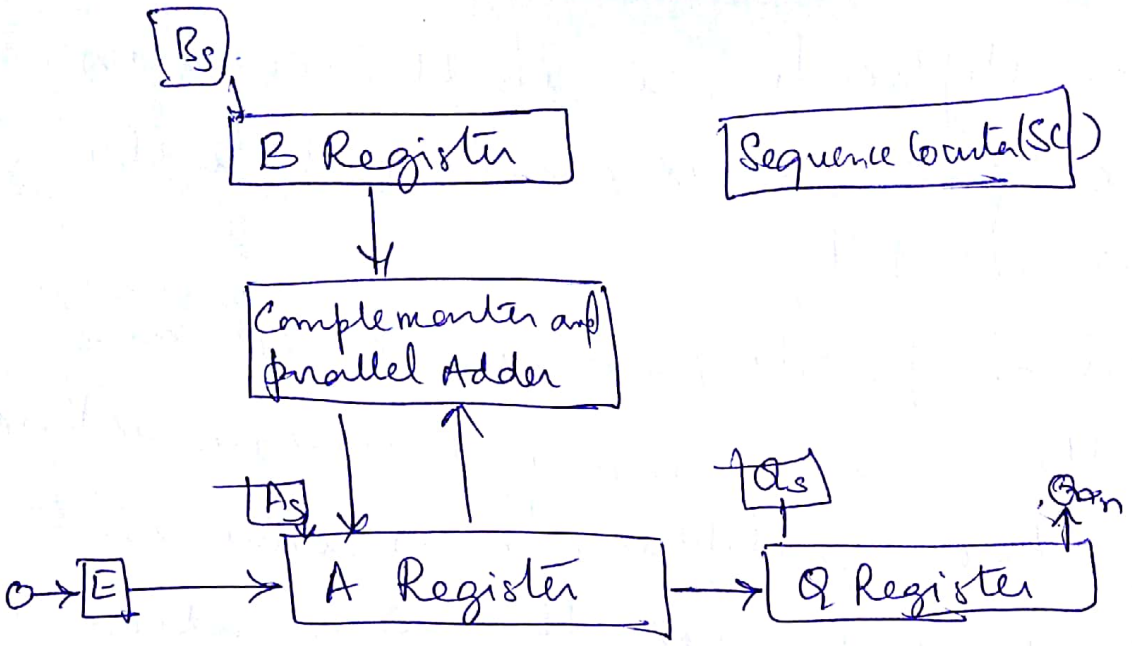


fig: Hardware implementation for multiplication in Signed-magnitude Data.

The multiplicand is in register B and the multiplier in Q. The sequence counter SC is set to a no equal to the no of bits in multiplier. The counter is decremented by 1 after forming each partial product. When the content of the counter reaches to zero, the product is formed and the process stops.

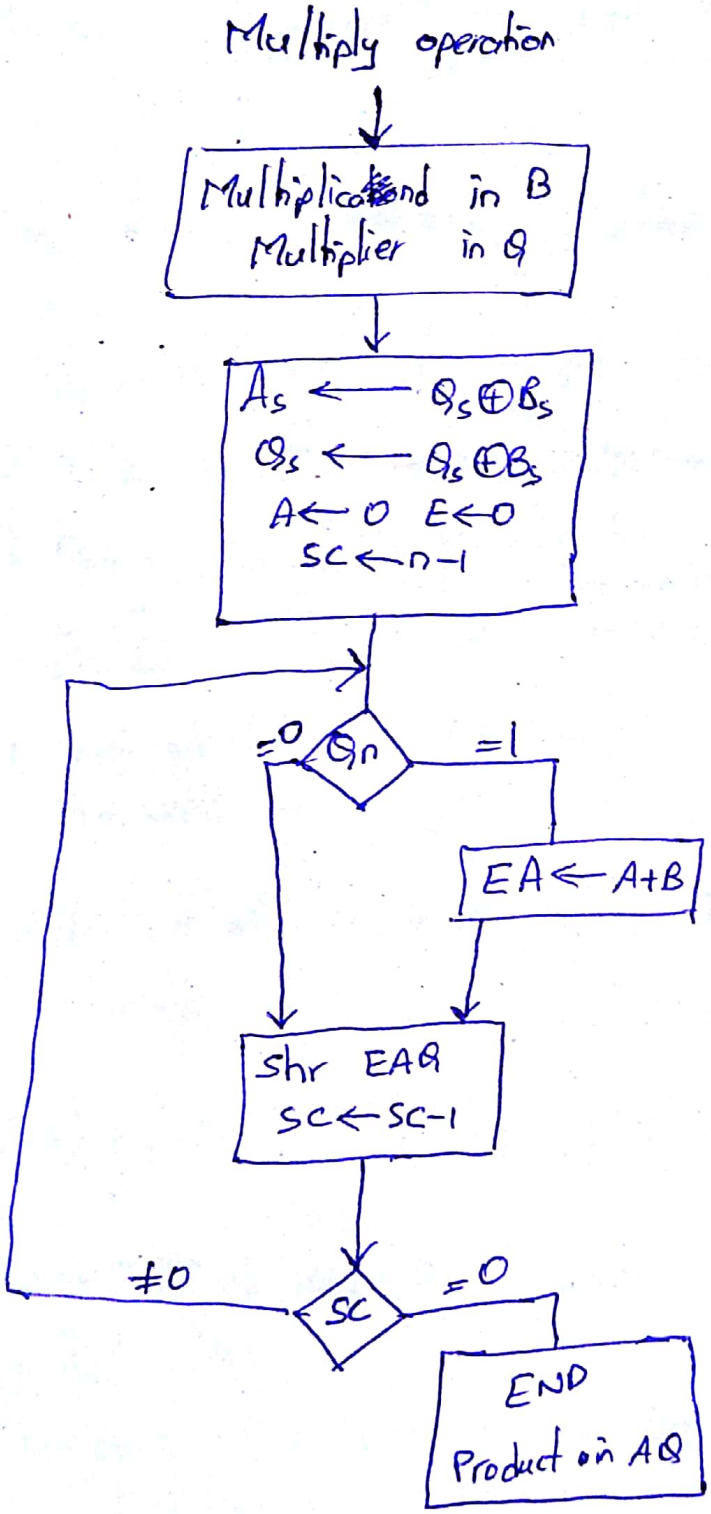
The sum of A and B forms a partial product ~~and~~ which is transferred EA register. Both partial product and multiplier are shifted to right (shr EAQ). ∴ Qn will hold the bit of the multiplier, which must be inspected next.

Q:

Multiplication:-

Signed magnitude data Algorithm:

Booth's Algorithm:-



shr → shift right ~~register~~
SC → sequence counter

Eg:- $B \leftarrow 10111$
$Q \leftarrow 10011$

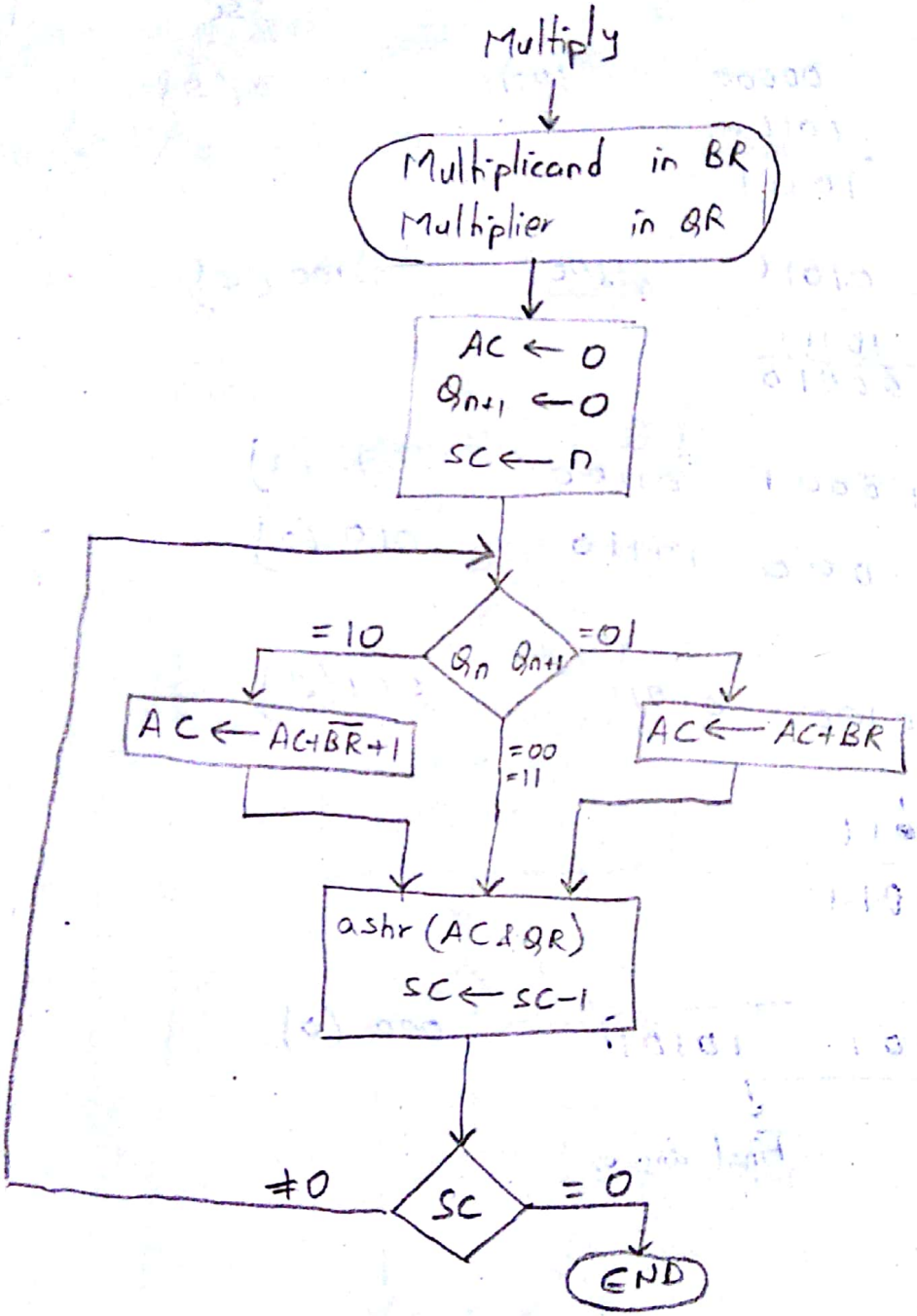
carry will go here

Multiplicand $B = 10111$	E	A	Q	SC
Multiplier in Q	0	00000	10011	101 (5)
$Q_n = 1$, Add B		10111		
Find partial product	0	10111		
shift right EAQ →	0	01011	11001	100 (4)
$Q_n = 1$; Add B →		10111		
	1	00010		
second pp shift right EAQ	0	10001	01100	011 (3)
$Q_n = 0$; shift right EAQ	0	01000	10110	010 (2)
$Q_n = 0$, shift right EAQ	0	00100	01011	001 (1)
$Q_n = 1$; Add $EA \leftarrow A + B$		10011		
	0	11011		
5th pp shift right EAQ	0	01101	10101	000 (0)

Final answer

Booth's Algorithm (Multiplication)

Flow Chart



Note:- AC−BR
= AC+(−BR)
= AC+(BR̄+1)

Eg:- $(-9) \times (-13) = +117$

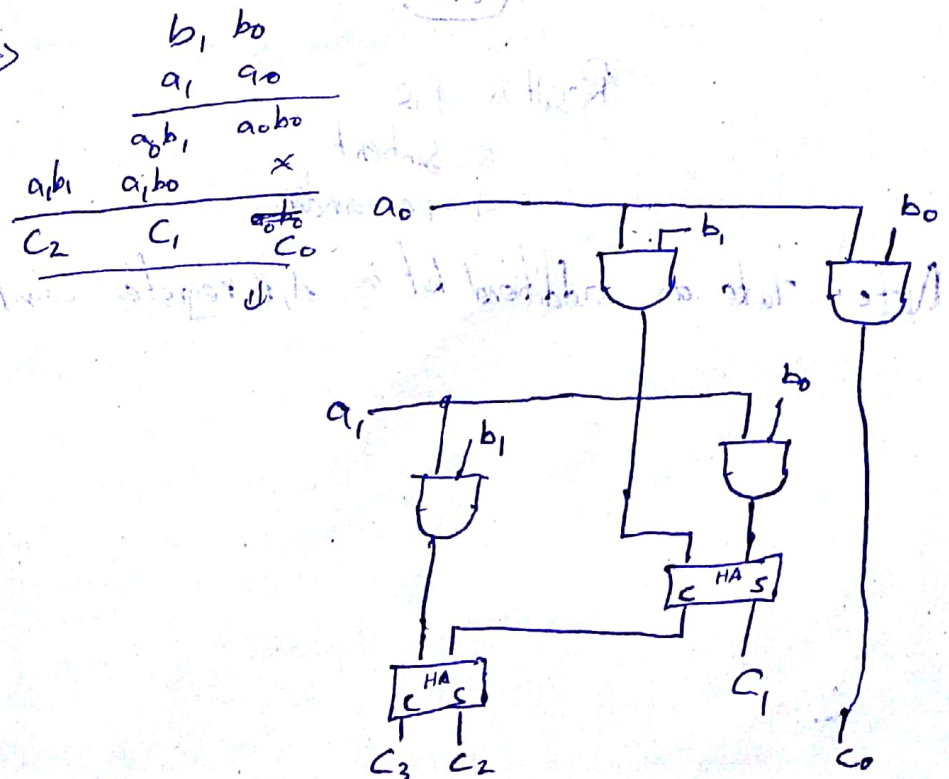
$-9\ (B) = 10111$

$-13\ (Q) = 10011$

$Q_n\ Q_{n+1}$	BR = 10111 $\overline{BR}+1$ = 01001 (BR)	AC	QR	Q_{n+1}	SC
~~Initial~~	Initial values	00000	10011 (Q_n)	0	101 (5)
10	Substract BR from AC	01001			
		01001			
	~~a~~arithmetic shift (SC←SC−1)	00100	11001	1	100 (4)
11	ashr (AC & QR) SC←SC−1	00010	01100	1	011 (3)
01	add BR	10111			
		11001			
	ashr (AC&QR) SC←SC−1	11100	10110	0	010 (2)
00	ashr (AC&QR) SC←SC−1	11110	01011	0	001 (1)
10	Asub BR i.e. Add $\overline{BR}+1$	01001			
		100111 → discard end carry			
	ashr (AC&QR) SC←SC−1	000111	10101	1	000 (0)

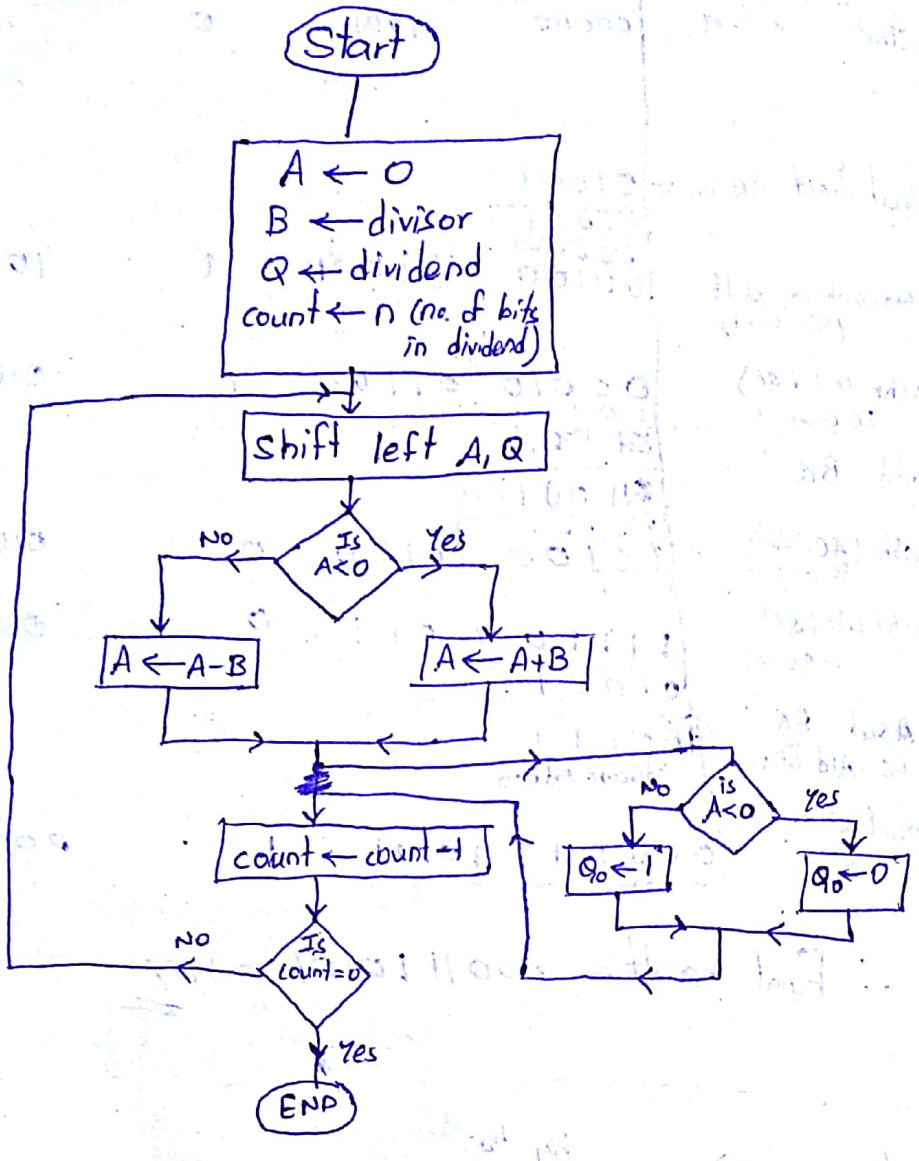
∴ Final result ⇒ 0001110101 = 117

Circuit diagram ⇒



non-Restoring divison :-

Algorithm :-



Result in A Q.

Q = Quotient

A = remainder

Note :- Take an additional bit in A, B register w.r.t Q register

Eg:- 3)10(3
9
1

~~B = 3 = 00011~~

Q = 10 = 1010 (4 bit)

B = 3 = 00011 (5 bits)

-B = 11101

Process	A	Q	Q_0	Count
Initial	00000	1010		4
Shift A, Q	00001	010	□	
A: A>0 ⇒ A ← A-B	11101			
As A<0 ⇒ Q_0 ← 0	11110	010	0	3
Shift left A, Q	11100	100	□	
A<0 ⇒ A ← A+B	00011			
A<0 ⇒ Q_0 ← 0	11111	100	0	2
Shift L A, Q	11111	000	□	
A<0 ⇒ A ← A+B	00011			
A>0 ⇒ Q_0 ← 1	1 00010	0001		1
Shift L, A, Q	00100	001	□	
A>0 ⇒ A ← A-B	11101			
A>0 ⇒ Q_0 ← 1	1 00001	0011		0

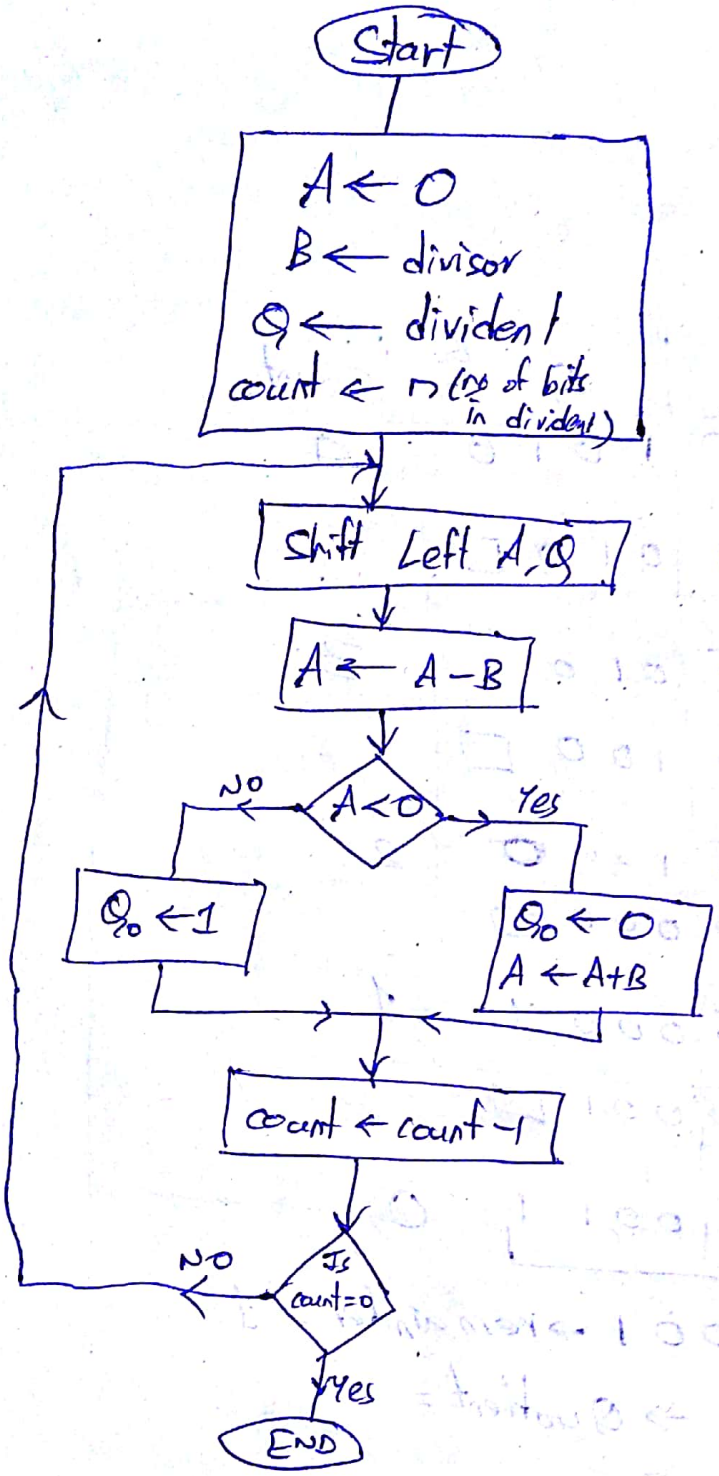
Discard end carry

Discard

∴ A = 00001 → remainder = 1

Q = 0011 → Quotient =

Restoring Algorithm



Eg:- 3)10(3
9
1

$Q = 10 = 1010$

$B = 3 = 00011$

$-B = 11101$

Process	A	Q	count
Initial	00000	1010	4
SL A,Q	00001	010 □	
$A \leftarrow A-B$	11101		
	11110	0100	
$A<0 \Rightarrow Q_0 \leftarrow 0$	00011		
$A \leftarrow A+B$			
disrad ←	1 00001	0100	3
$c \leftarrow c-1$			
SL A,Q	00010	100 □	
$A \leftarrow A-B$	11101		
	11111	1000	
$A<0 \Rightarrow Q_0 \leftarrow 0$	00011		
$A \leftarrow A+B$			
	1 00010	1000	2
SL A,Q	00101	000 □	
$A \leftarrow A-B$	11101		
	1 00010	0001	1
$A>0 \Rightarrow Q_0 \leftarrow 1$			
$c \leftarrow c-1$			
SL A,Q	00100	001 □	
$A \leftarrow A-B$	11101		
	1 00001	0011	0
$A>0$ $Q_0 \leftarrow 1$			

$A = 00001$

$Q = 0011$

UNIT - II

Register Transfer Language.

→ The operations executed on data stored in registers are called micro operations

→ The internal h/w organization of a digital computer is best designed by

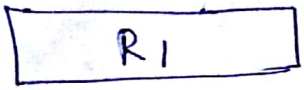
1. The set of registers it contains and their function
2. The sequence of micro operations performed on the binary info stored in the registers
3. The control that initiates the sequence of micro operations

→ The symbolic notation used to describe the micro operation transfers among registers is called a register transfer language

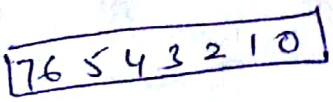
Register Transfer

The contents of one register is transfered to another with a symbolic form by means of replacement operator

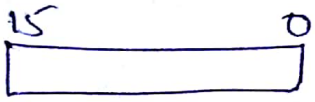
$$R_2 \leftarrow R_1$$



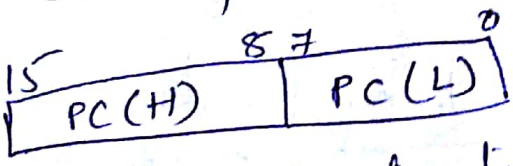
(a) Reigistin R_1



(b) Showing individual bits



(c) Numbering of bits



(d) Divided into two parts.

→ We want the transfer to occur only under a predetermined control condition by means of if - then statement

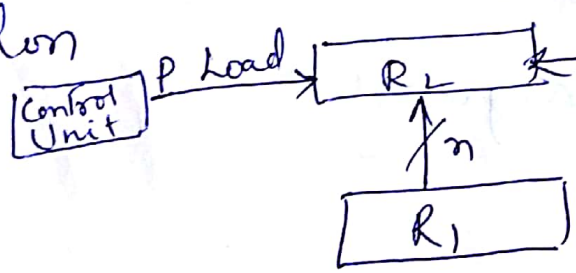
$$\text{If } (P = 1) \text{ then } (R_2 \leftarrow R_1)$$

where P is ~~Parallel load~~ a control signal generated in control unit.

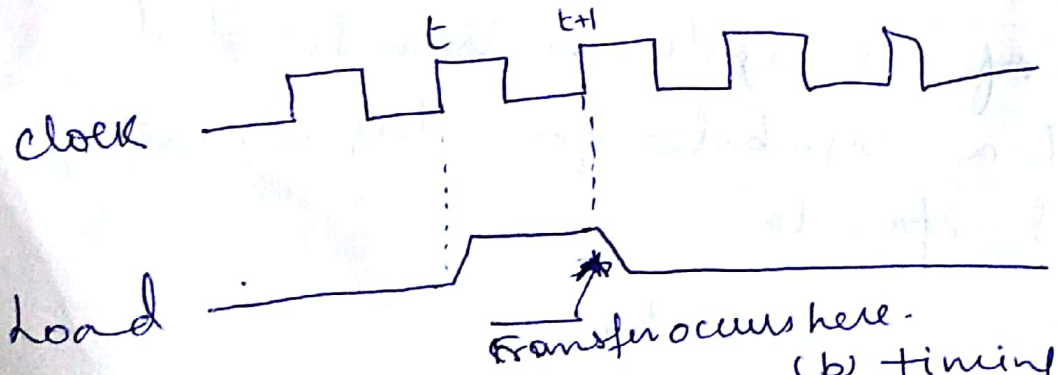
A <u>control function</u> is a boolean variable that is equal to 1 or 0.

$$P : R_2 \leftarrow R_1.$$

The control condition is terminated by (:) a colon



(a) Block diagram.



(b) timing diagram

P becomes active just after time t, the actual transfer does not occur until the register is triggered by the next positive transition of the clock at time t+1.

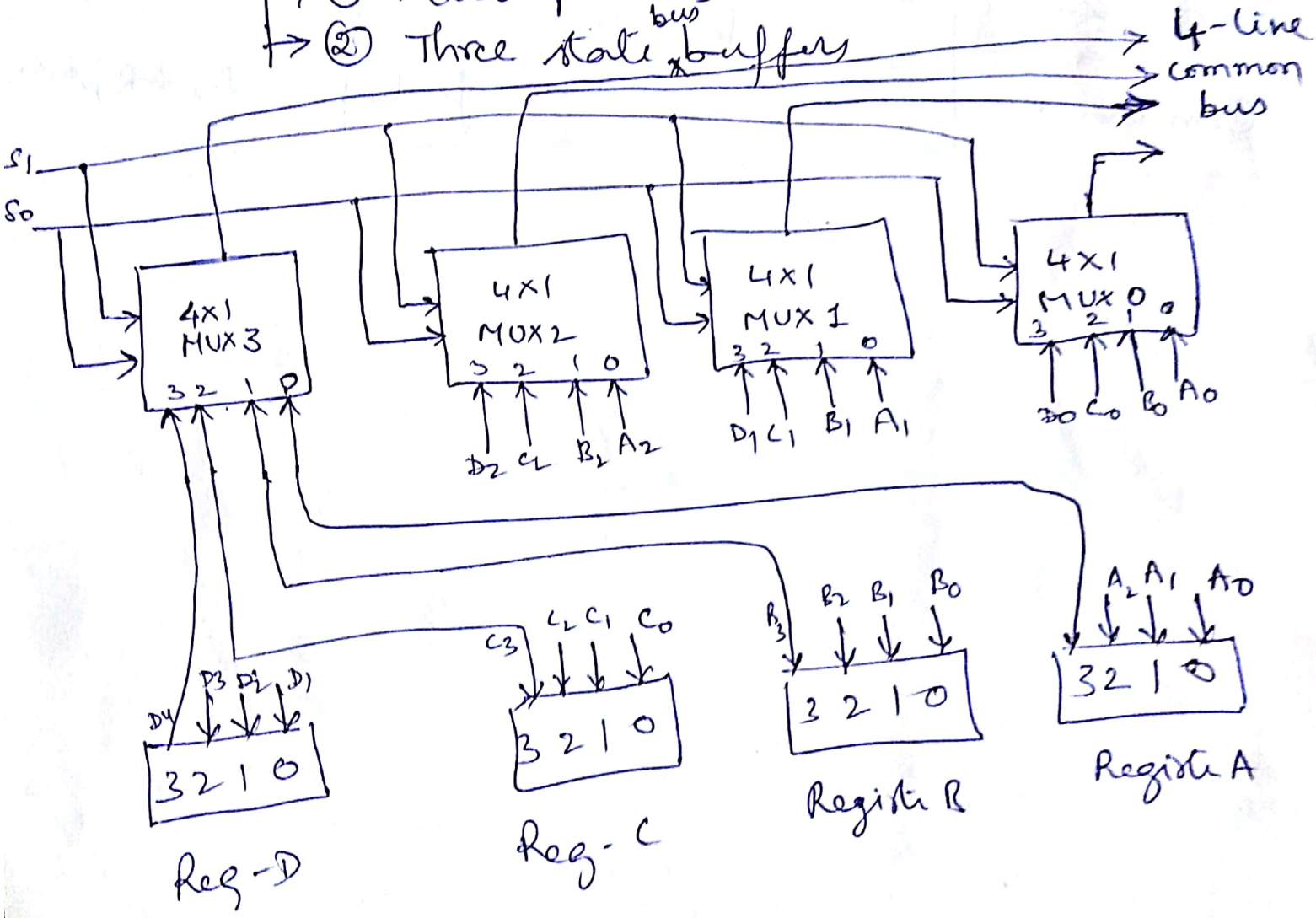
Symbol	Description	Examples
Letters and numerals	Denotes a register	MAR, R2
Parentheses ()	Denotes a part of register	$R_2(0-7)$, $R_2(L)$
Arrow ←	Denotes transfer of information	$R_2 \leftarrow R_1$
Comma ,	Separates two microoperation	$R_2 \leftarrow R_1, R_1 \leftarrow R_2$

Bus and Memory Transfers

Use common bus system, to avoid no of wires for transferring data from one register to another.

→ A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during each particular register transfer.

→ Two ways of constructing common bus
- → ① Multiplexers
- → ② Three state bus buffers



Bus System for four registers.

Each register has 4 bits, i.e 0-3. ~~and~~ The bus consists of ~~four~~ four 4X1 multiplexers each having four data inputs i.e 0-3. (Two selection input S_1 and S_0) The two selection lines S_1 and S_0 are connected to the selection inputs of all four multiplexers. The selection line choose the four bits of one register and transfer them into the four line common bus.

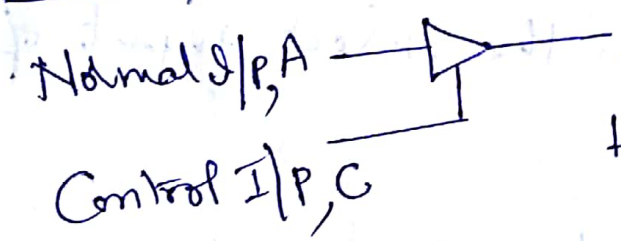
When $S_1 S_0 = 00$, Contents of A is transfered on Bus BUS ← A.

$S_1 S_0 = 01$, BUS ← B

$S_1 S_0 = 10$, BUS ← C

$S_1 S_0 = 11$ BUS ← D

② Three State Bus Buffer.



Output Y = A if C = 1

High Impedance if C = 0

→ A bus system can be constructed with three state gates. A three state gate is a digital circuit that exhibits three states.

Two states are 0 and 1. Third state is high impedance state, which means that the output is disconnected and does not have a logical significance.

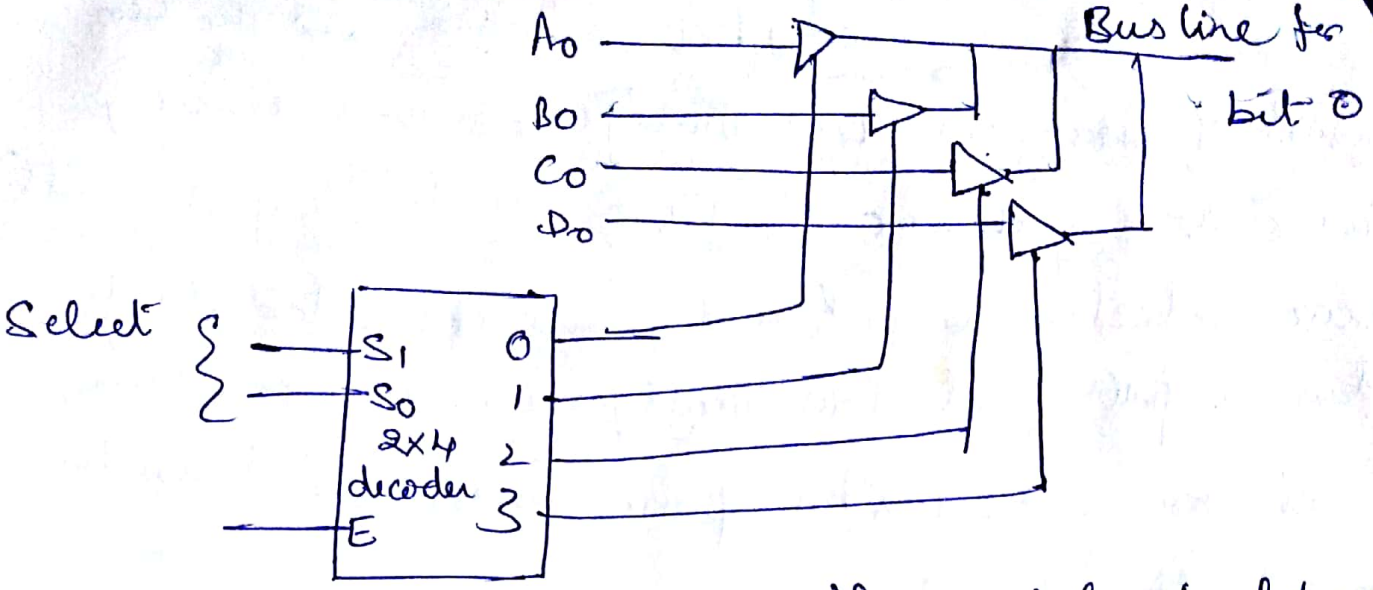


fig: Bus line with three state-buffers

When the enable, E input of the decoder is 0, all of its 4 inputs are 0, and the bus line is in a high impedance state.

When the enable E, input is active, one of the three-state buffers will be active, depending on the binary value in the select inputs of the decoder.

Memory Transfer :- ① Read ② Write.

① Read: DR ← M[AR]

→ This causes a transfer of information into DR from the memory word M selected by the address in AR.

② Write: M[AR] ← R1

→ Transfer of information from R1 into the memory word M selected by the address in AR.

Arithmetic Micro operations.

A microoperation is an elementary operation performed with the data stored in registers.

4 Categories.

1. Register transfer Micro operations $R_1 \leftarrow R_2$
2. Arithmetic microoperations (Numeric data)
3. Logical microoperations (Nonnumeric data
4. Shift microoperations

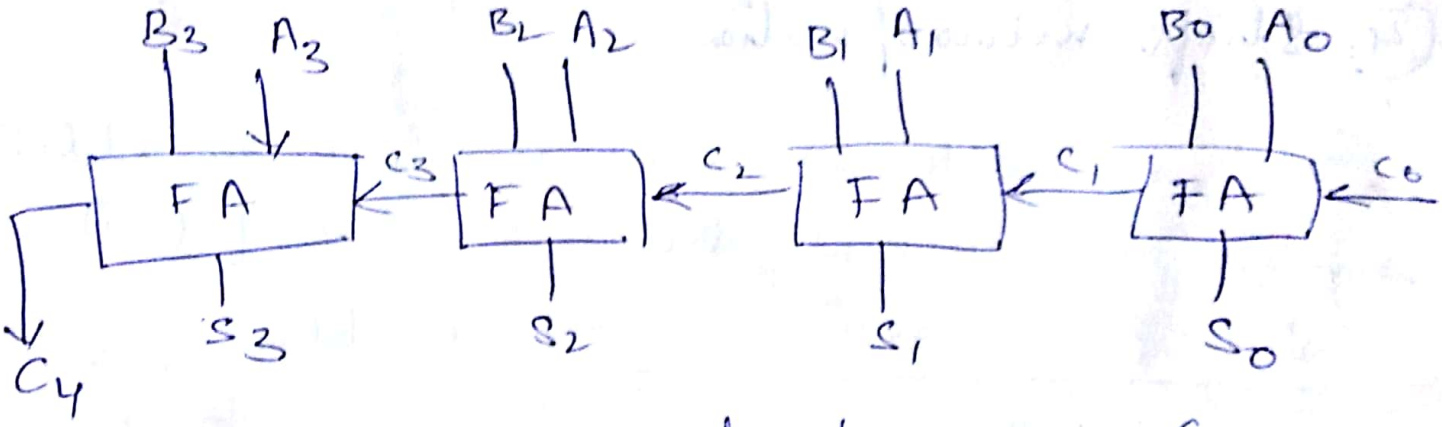
→ The Basic arithmetic micro operations are addition, subtraction, increment, decrement and shift (arithmetic)

Symbolic designation	Description
$R_3 \leftarrow R1 + R2$	
$R3 \leftarrow R1 - R2$	
$R2 \leftarrow \overline{R_2}$	
$R_2 \leftarrow \overline{R_2} + 1$	
$R_3 \leftarrow R1 + \overline{R_2} + 1$	
$R1 \leftarrow R1 + 1$	
$R1 \leftarrow R1 - 1$	

Binary Adder

The digital circuits that forms the arithmetic sum of two bits and a previous carry is called binary adder.

→ The binary adder is constructed with full-adder circuit connected ~~to the~~ in cascade, with the o/p carry from one full-adder connected to the input carry of the next full-adder as shown in fig.



The i/p carry is C_0 and o/p carry is C_4.
The S output of the full-adders generate the required sum bits

The addition and subtraction can be combined into one common circuit by including an exclusive-OR gate with each full-adder. The mode i/p M controls the operation.

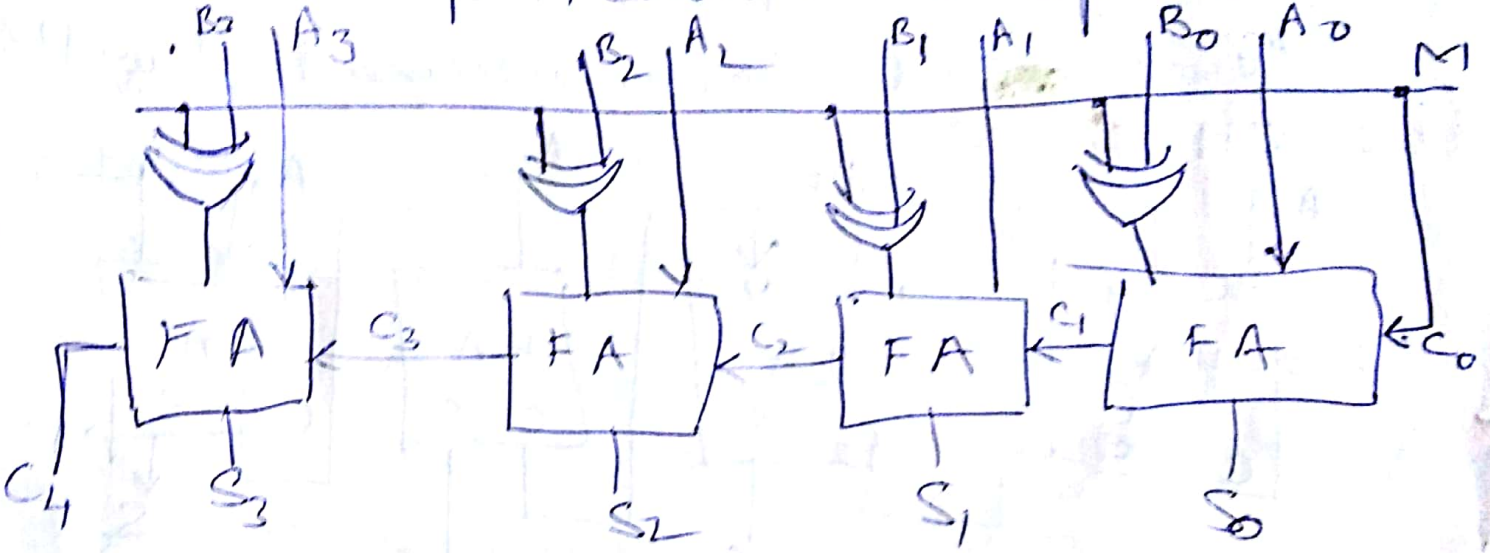


fig. binary adder & subtractor

→ When M=0 the circuit is an adder and when M=1 the circuit becomes a subtractor.

→ When M=0, we have $B \oplus 0 = B$. The full adders receive the value of B, the input carry is 0. and the circuit performs A+B.

→ When M=1, we have $B \oplus 1 = \overline{B}$ and $C_0 = 1$. The B inputs are all complemented and 1 is added through i/p carry and the circuit performs $A + \overline{B} + 1$.

Binary Incrementer

The increment microoperation adds one to a number in a register. This micro operation is easily implemented with a binary counter as shown in the figure.

For ex. 0110, after increment, it will be 0111.

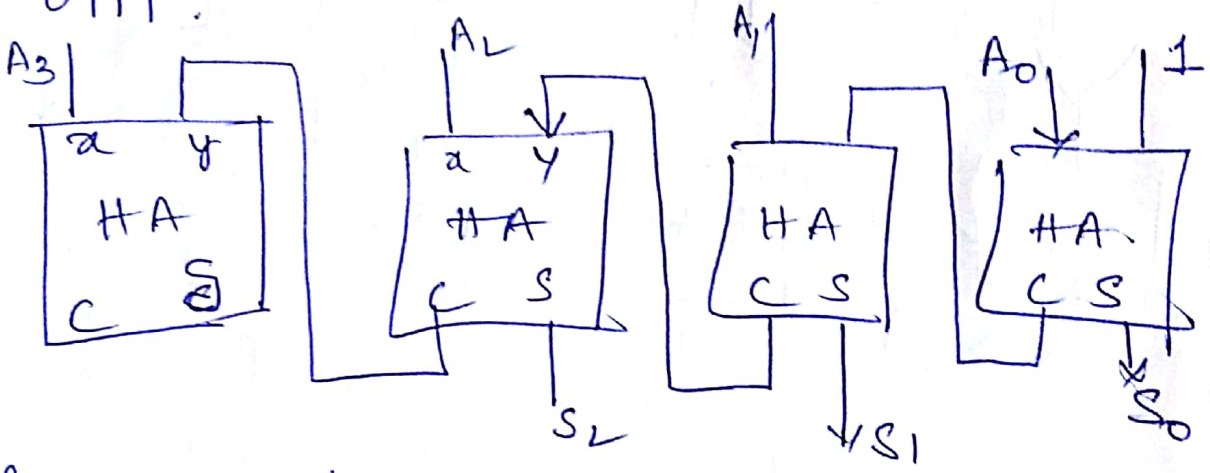


fig. 4-bit binary incrementer.

One of the inputs to least significant half-adder (HA is connected to logic 1 and the other input is connected to the Least Significant bit of the no. to be incremented. The o/p carry from one half-adder is connected to one of the inputs of the next-higher-order half-adder. The circuit receives the four bits from A_0 through A_3, adds one to it, and generates the incremented output in S_0 through S_3.

Arithmetic Circuit

All the arithmetic microoperation can be implemented in one composite arithmetic circuit. The basic component of an arithmetic circuit is the parallel adder. By controlling the data inputs to the adder, it is possible to obtain different types of arithmetic operations.

It has four full adder circuits that constitute 4-bit adder and four multiplexers for choosing different operations.

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

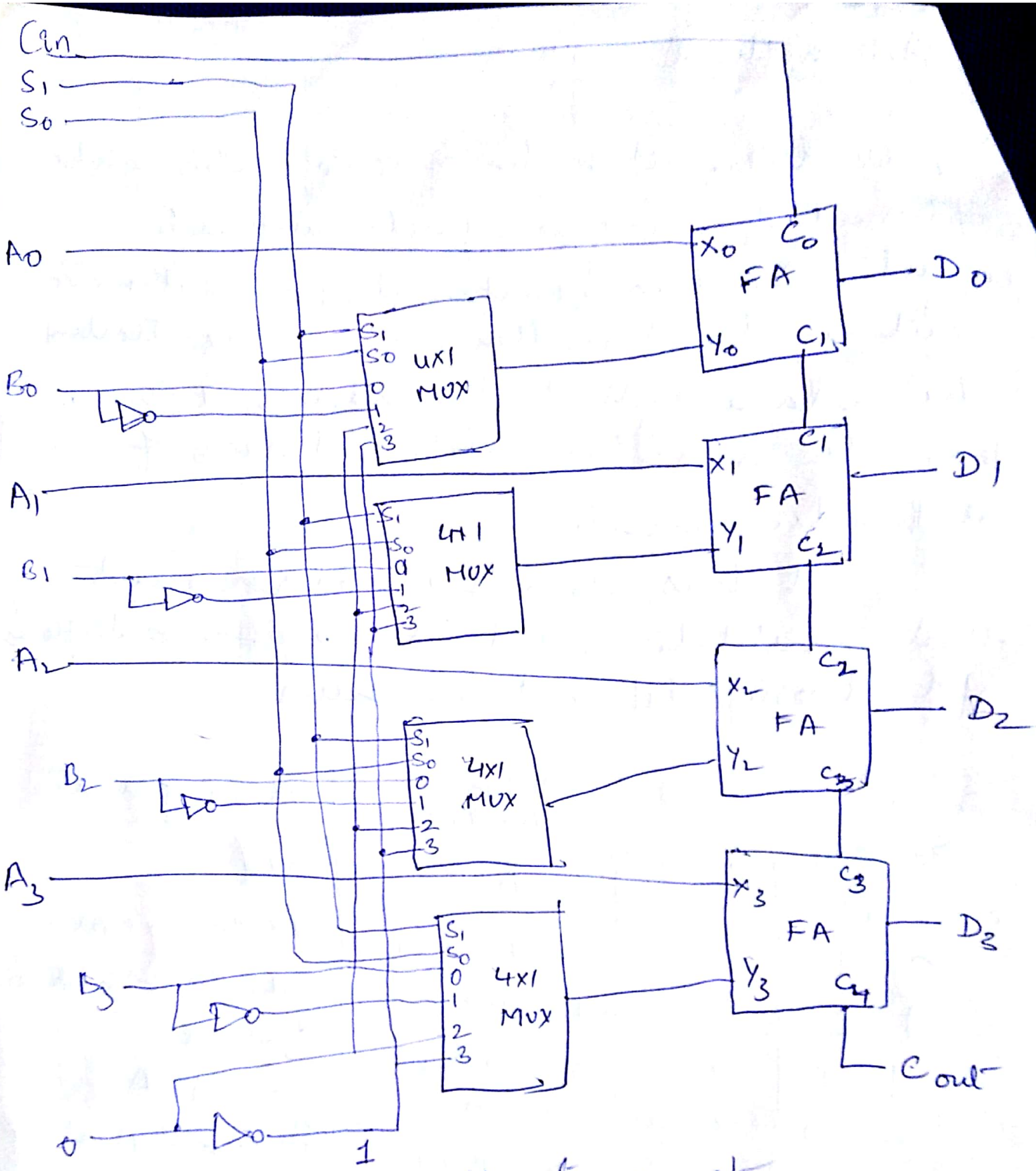


fig. 4-bit arithmetic circuit

There are two 4-bit inputs A and B and a 4-bit output D. The four inputs from A go directly to X inputs of the binary adder. Each of the four inputs from B are connected to the data inputs of the multiplexers. The multiplexers data inputs also receive the complement of B. The other two data inputs are connected to logic-0, and logic 1. The four multiplexers are controlled by two selection inputs, S_1 and S_0. The i/p carry C_{in} goes to the carry input of Full adder in the least significant position. The other carries are connected from one stage to the next.

The output of this circuit is

$$\boxed{D = A + Y + C_{in}}$$

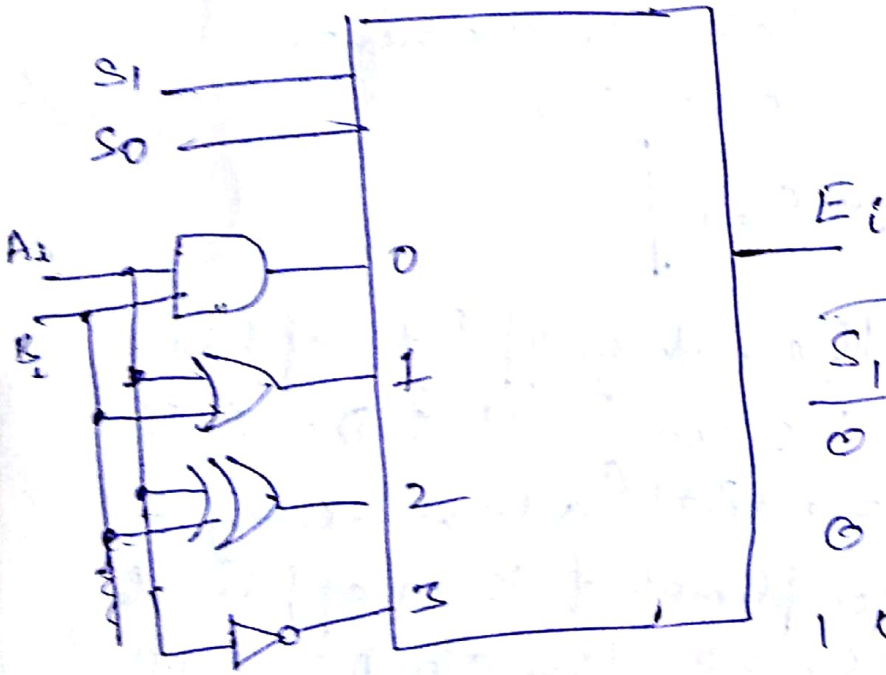
→ when $S_1 S_0 = 00$, the value of B is applied to Y inputs of adder. If $C_{in} = 0$, the o/p, $D = A + B$
if $C_{in} = 1$, the o/p, $D = A + B + 1$ (add with carry)

→ when $S_1 S_0 = 01$, the complement of B is applied to Y inputs of adder. If $C_{in} = 0$, then o/p $D = A + \bar{B}$ (subtract with borrow, ie $A - B - 1$)
if $C_{in} = 1$, then $D = A + \bar{B} + 1$ ie ($D = A - B$)

when $S_1 S_0 = 11$, all 1's are inserted to Y i/p of adder
when if $C_{in} = 0$ then $D = A - 1$. This is because
all 1's is equal to 2's complement of 1.
which acts like a decrements
when if $C_{in} = 1$ then $D = A - 1 + 1$, which is equal to
direct transfer of A.

when $S_1 S_0 = 10$, all 0's are inserted to Y inputs
$D = A + 0 + C_{in}$ (all B inputs are neglected)
if $C_{in} = 0$, then $D = A$.
if $C_{in} = 1$, then $D = A + 1$ which acts like
an increment.

Logic micro operation



S_1	S_0	O/P	operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \bar{A}$	complement

Logic microoperations are very useful for manipulating individual bits or a portion of a word stored in a register. They can be used to change bit values, delete a group of bits, or insert new bit values into a register.

(1) Selective set :- This operation sets to 1 the bits in register A where there are corresponding 1's in register B.

```
 1010 — A before
 1100 — B
 ----
 1110 → A after
```

(2) Selective - complement : This operation complement bits in A where there are corresponding 1's in B.

```
 1010 — A
 1100 — B
 ----
 0110 — A (after)
```

(3) Selective - clear :- It clears to 0 the bits in A only where there are corresponding 1's in B.

```
 1010 — A
 1100 — B
 ----
 0010 — A after
```

Mask :- It is similar to selective-clear opn except that the bits of A are cleared only where there are corresponding 0's in B.

```
 1010 - A
 1100 - B
 ------
 1000 - A after masking.
```

Clear :- It compares the words in A and B and produces an all 0's result if the two nos are equal.

```
 1010   A
 1010   B
 ------
 0000 - A.
```

Insert operation :- It inserts a new value into a group of bits. This is done first masking the bits and then OR ing them with the required ~~bits~~ value.

```
 0110  1010  A
 0000  1111  B(mask)
 ----------
 0000  1010  A after mask
```

Then insert the new value

```
 0000  1010 - A before
 1001  0000 - B(insert)
 ----------
 1001  1010 - A(After insertion)
```

Shift micro operation

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

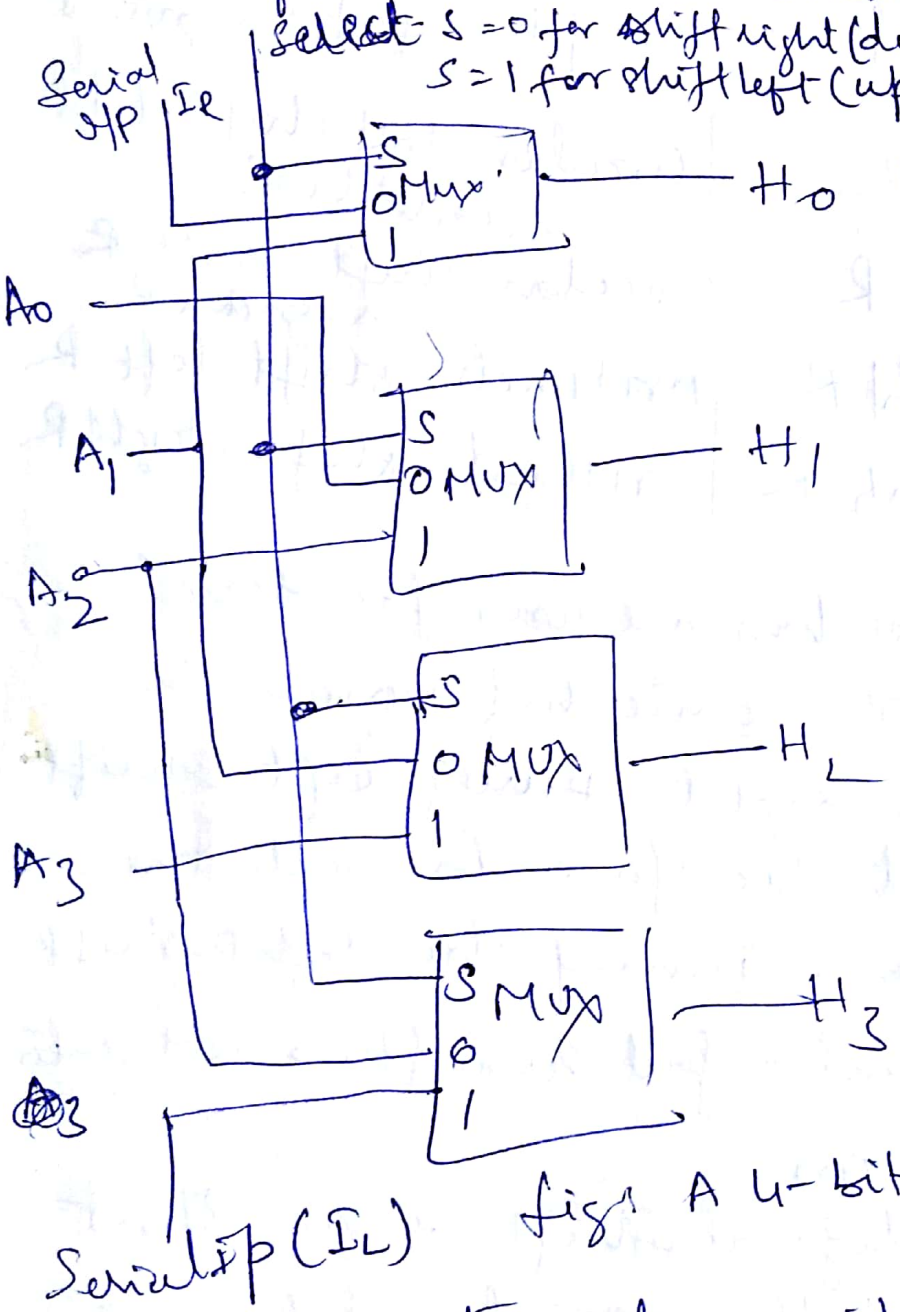
Shift microoperations are used for serial transfer of data. The contents of register is shifted to left or right. During left shift the serial input transfer a bit into the rightmost position. During the right shift operation the serial input transfers a bit into the left position.

Logical shift:- A logical shift is one that transfers 0 through the serial input.

Circular shift:- It is a rotate operation, which circulates the bits of the register around the two ends without loss of information.

This is done by connecting the serial output to its serial input.

<u>Arithmetic shift</u> :- It shifts a signed binary number to the left or right. It must leave the sign bit unchanged because the sign of the number remains the same.



Function Table

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

fig: A 4-bit combinational circuit shifter.

→ A combinational circuit shifter can be constructed with multiplexers as in fig.

A 4-bit shifter has four data inputs, A_0 through A_3 and four data o/ps H_0 thru' H_3.

There are two serial inputs, one for shift left (IL) and other for shift right (IR). when the selection input S = 0, the inputs data are shifted right (down in the diagram). when S = 1, the input data are shifted left (up in diagram).

Arithmetic Logic Shift Unit:

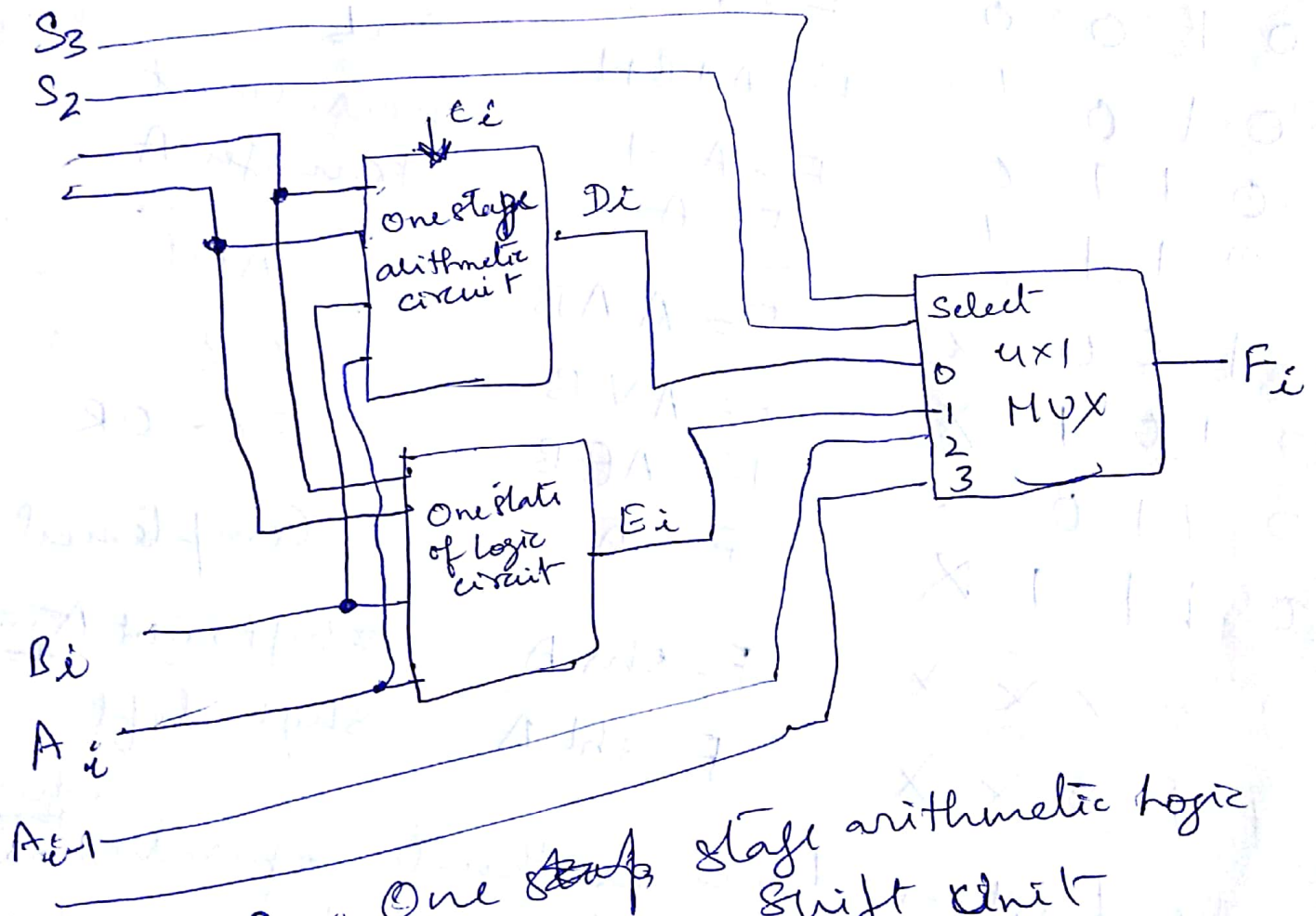


fig: One ~~stage~~ stage arithmetic logic shift unit

The circuit provides 8 arithmetic operations, four logic operations, and two shift operations. Each operation is selected with five variables C_1, S_0 and C_{in}. The input carry C_{in}

is used for selecting an arithmetic operation only.

Operation Select S_3	S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A + 1$	Increment A
0	0	0	1	0	$F = A + B$	Add
0	0	0	1	1	$F = A + B + 1$	Add with carry
0	0	1	0	0	$F = A + \overline{B}$	subtract with borrow
0	0	1	0	1	$F = A + \overline{B} + 1$	Sub.
0	0	1	1	0	$F = A - 1$	Decrement
0	0	1	1	1	$F = A$	transfer A
0	1	0	0	X	$F = A \wedge B$	AND
0	1	0	1	X	$F = A \vee B$	OR
0	1	1	0	X	$F = A \oplus B$	EX-OR
0	1	1	1	X	$F = \overline{A}$	Complement A
1	0	X	X	X	$F = shr\ A$	shift right ~~A into~~ ~~B~~
1	1	X	X	X	$F = shl\ A$	shift left

The first eight are arithmetic operation. which ~~that~~ are selected with $S_3S_2 = 00$, the next four are logic operations and are selected with $S_3S_2 = 01$. The input carry has no effect and marked as dont care condition, X. The last two operations are shift opns and are selected with $S_3S_2 = 10$ and 11 and the other three selection inputs have no effect on the shift.

Basic Computer Organization and Design

Morris Mano
Chapter 5.

Instruction codes :- An instruction code is a group of bits that instruct the computer to perform a specific operation.

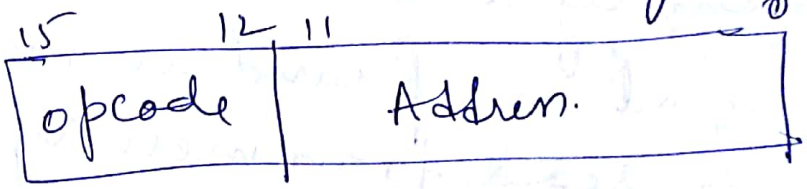
It is divided into two parts.

1. Opcode — Ex : add, sub, multiply, divide, shift, complement

   The no. of bits required for the opcode depends on the total no. of operations available in the computer.

   The opcode consists 'n' bits for 2^n distinct operations

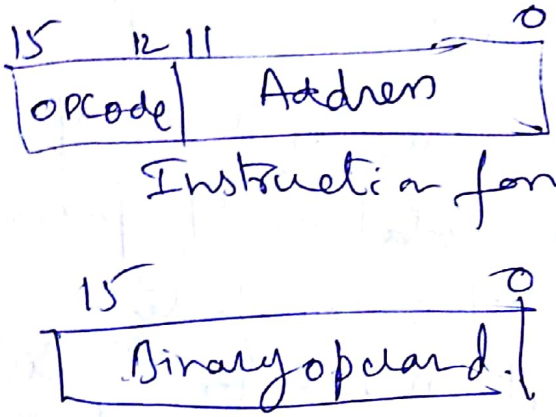
2. Address :- Memory words can be specified in instruction codes by their address.



2^{12} — 4096 memory locations.

Stored Program Concept

The simplest way to organize a computer is to have one processor register and an instruction code of format with two parts. The first part specifies the operation to be performed (OPCode) and the second specifies an address.



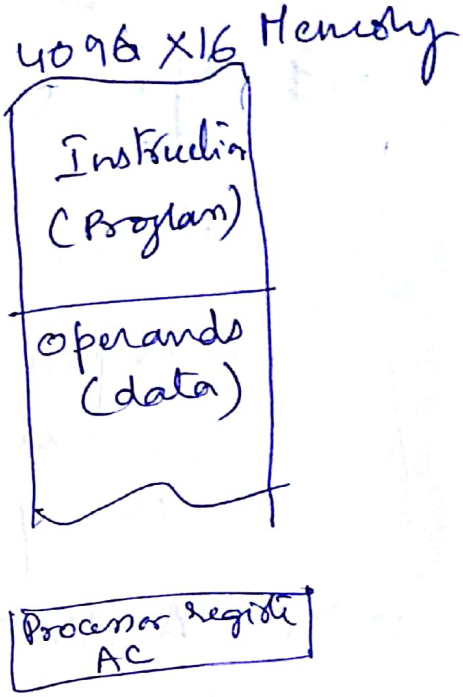


fig. Stored Program organization

The memory address tells to the control where to find an operand in memory. This operand is read from memory and used as the data to be operated on together with the data stored in the processor register (AC).

→ Instructions are stored in one section of memory and data in another.

→ 4 bits are reserved for opcode. ∴ 2^4 distinct operation are available (16)

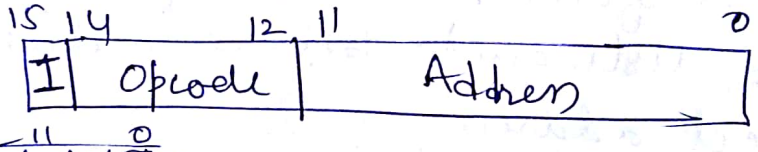
12 bits for address. ∴ 2^{12} ie 4096 address location. Each address loc. contain 16 bit word.

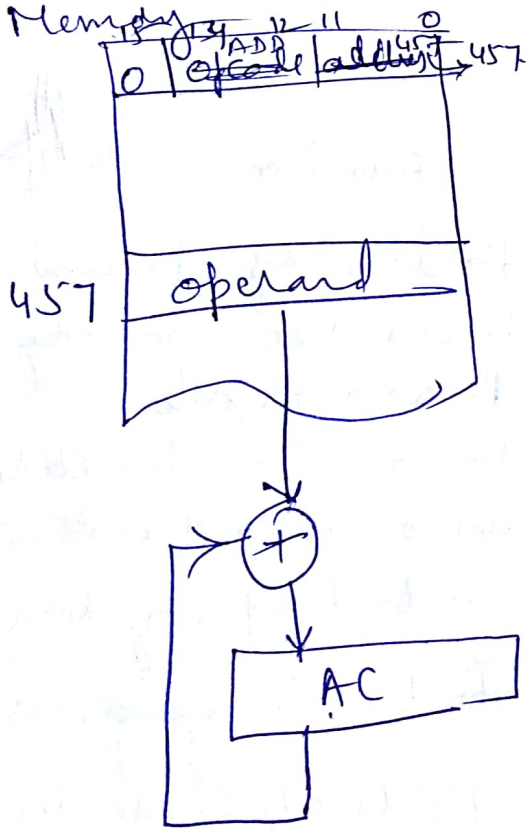
Accumulator :- Computers that have a significant single processor register usually assign to it the name Accumulator (AC).

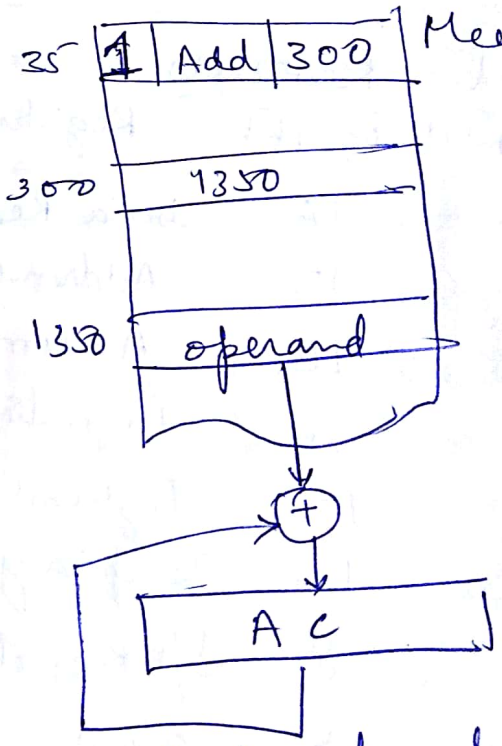
→ The operation is performed with memory operand and the content of a AC.

→ operations such as clear AC, increment/decrement AC, complement AC, operate on data stored in the AC register. For these operations, the second part of instruction code (bits 0 – 11) is not needed for specifying a memory address.

Direct and Indirect address :-

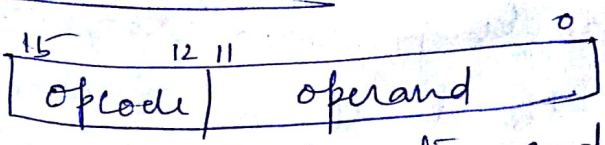






(b) Indirect address

1. Immediate Address :-



Instruction code for immediate address

→ Sometimes actual operand is present in the instruction code instead of address is called a immediate address.

2. Direct Address :- when the second part of specifies the address of an operand, then it is direct address

3. Indirect address :- when the second part of the instruction code specifies an address of a memory word in which the address of the operand is found. One bit of information is used to distinguish between a direct and an indirect address.

Computer Registers

Register Symbol	No. of bits	Register Name	Function
DR	16	Data Register	Hold memory operand
AR	12	Address Register	Hold address for memory
AC	16	Accumulator	Processor Register
IR	16	Instruction Register	Holds Instruction code
PC	12	Program Counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data
INPR	8	I/P Register	Holds I/P character
OUTR	8	O/P Register	Holds O/P character

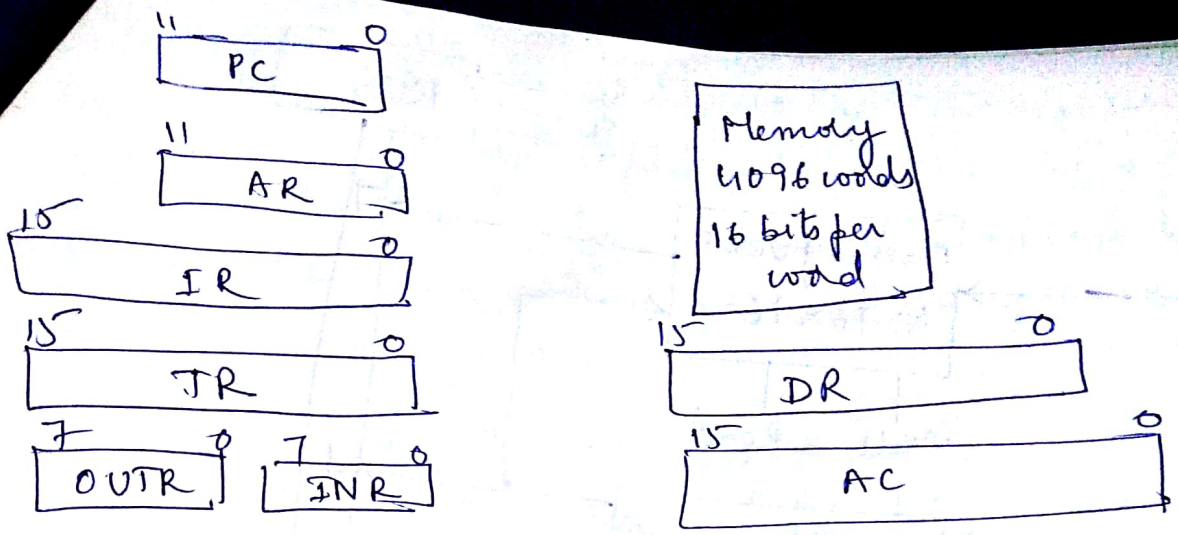
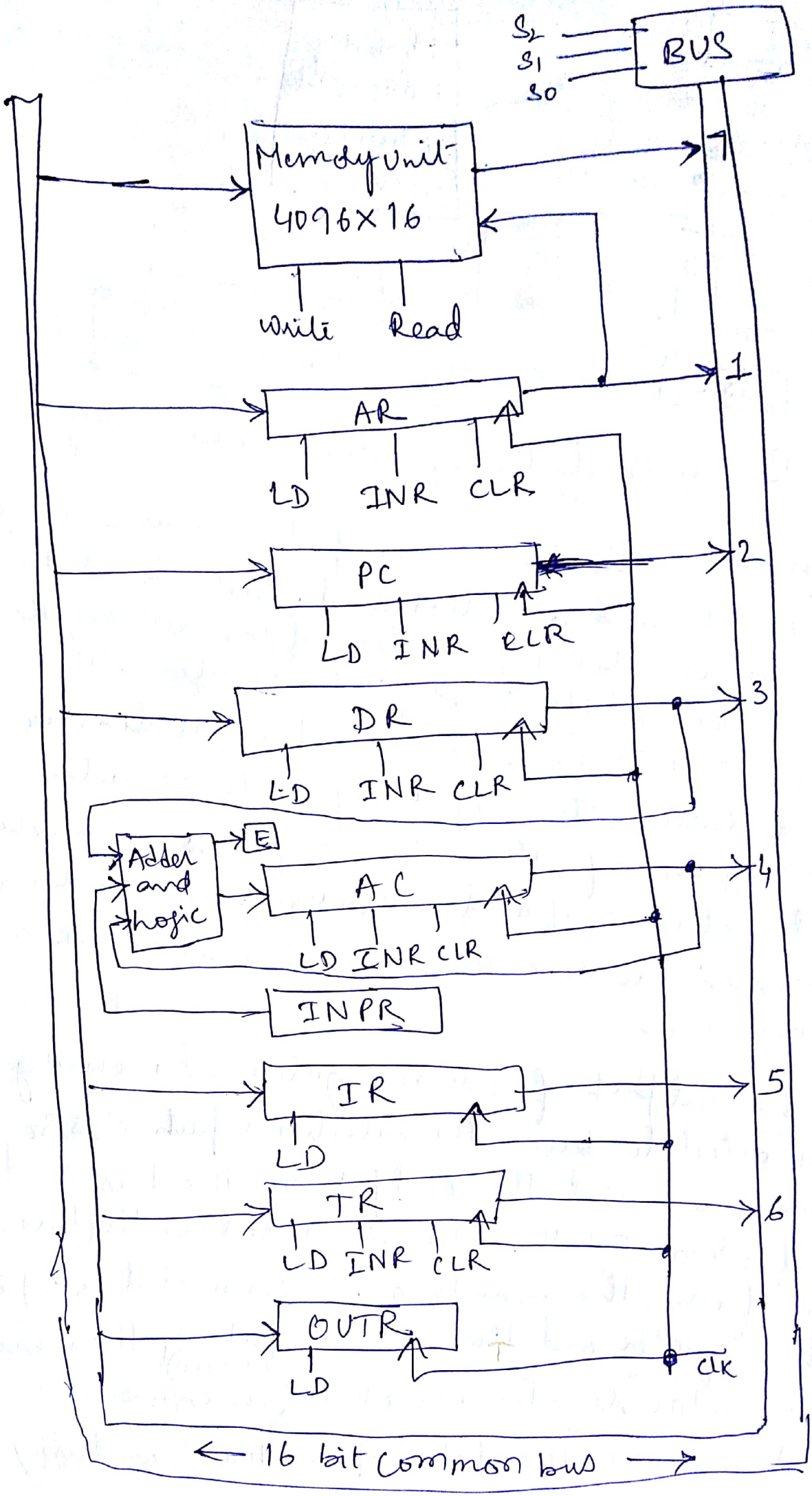


fig: Basic Computer registers and memory

Common Bus:- The basic computer has 8 registers a memory unit and a control unit. Paths must be provided to transfer information from one register to another and between memory and registers. The no of wires will be excessive if connections are made between the outputs of each register and the inputs of the other registers. A more efficient scheme for transferring information in a system with many registers is to use a common bus.

The output of seven registers and memory are connected to bus. The selection inputs $S_2 S_1 S_0$ are used to select the output on the bus. For ex: If $S_2 S_1 S_0 = 011$ then DR is placed on the bus. The lines from the ~~com~~ bus are connected to inputs of each register and the data inputs of the memory. The particular register whose LD (LOAD) is ~~equal~~ enabled receives the data from the bus during


S2
S1
S0
BUS
Memory unit
4096×16
Write
Read
7
AR
LD
INR
CLR
1
PC
LD
INR
CLR
2
DR
LD
INR
CLR
3
Adder and logic
E
AC
LD
INR
CLR
4
INPR
IR
LD
5
TR
LD
INR
CLR
6
OUTR
LD
CLK
16 bit common bus

the next clock pulse transition. The memory receives the contents of the bus when its write input is activated. The memory places its 16-bit O/P onto the bus when the read input is activated and $S_2 S_1 S_0 = 111$

Four registers, DR, AC, IR, and TR have 16-bits each. Two register AR and PC have 12 bits each since they hold memory address.

→ when AR and PC are applied on 16-bits bus, the four Most significant bits are 0.

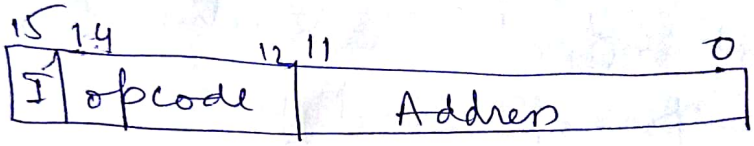
Five registers have three control I/Ps : LD(load) INR (increment), and CLR (clear). This type of register is equivalent to a binary counter with parallel load and synchronous clear.

→ The content of any register can be applied onto the bus and an operation can be performed in the adder and logic circuit during the same clock pulse.

For ex: DR ← AC, AC ← DR can be executed at the same time.

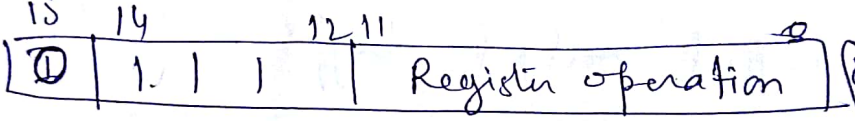
Computer Instructions

The basic computer has three instruction code formats.



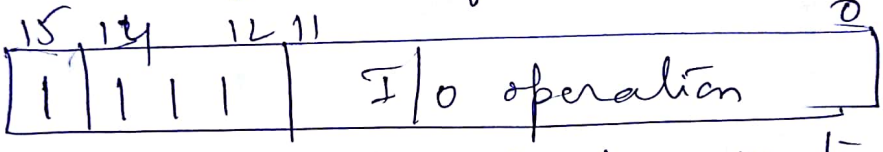
(opcode = 000 thru 110)

(a) Memory - reference instruction



(opcode = 111, I = 0)

(b) Register - reference instruction



(opcode = 111, I = 1).

(c) Input - output instruction

Basic computer instruction formats.

The type of instruction is recognized by the computer control from the four bits in position 12 through 15 of the instruction. If the three opcode bits in positions 12 through 14 are not equal to 111, the instruction is a memory-reference type and the bit in position 15 is taken as the addressing mode I.

If the 3-bit opcode is equal to 111, control then inspects the bit in position 15. If this bit is '0', the instruction is a register-reference type.

If the bit is 1, the instruction is an input-output type.

Instruction set completeness.

The set of instructions are said to be complete if the computer includes a ~~significant~~ sufficient no. of instructions in each of the following categories

1. Arithmetic, logic & shift instructions
2. Instruction for moving data to and from memory & processor registers
3. Program control instructions
4. I/p and o/p instructions

Instruction cycle: Each instruction cycle consists of a following phases.

1. Fetch an instruction from memory.
2. Decode the instruction.
3. Read the effective address from memory if the instruction has an indirect address.
4. Execute the instruction.

after completion of step 4, the control goes back to step 1 and execute till a Halt instruction is encountered.

→ PC is loaded with the first instr of the program. The sequence counter is set to '0' providing a decoded timing signal T_0, after each clock pulse SC is incremented by one. so that the timing signals go through a sequence T_0, T_1, T_2 and so on. The microoperation for the fetch and decode phases can be specified by

$$T_0 : AR \leftarrow PC$$

$$T_1 : IR \leftarrow M(AR),\ PC \leftarrow PC+1$$

$$T_3 : D_0 \ldots D_7 \leftarrow \text{Decode}(IR(12-14)),\ AR \leftarrow IR(0-11),$$
$$I \leftarrow IR(15).$$

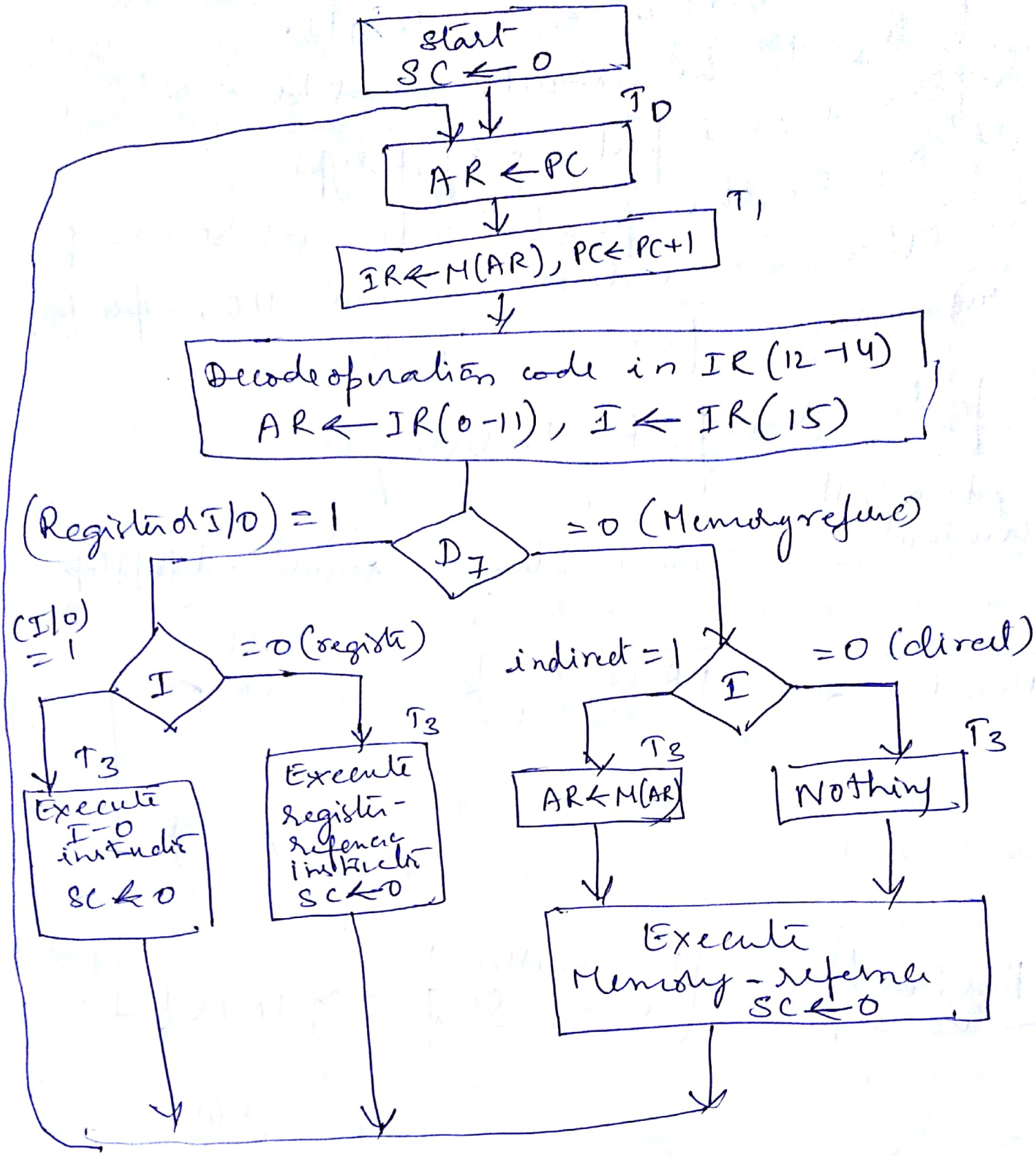


fig: Flowchart for instruction cycle.

Decoder output D_7 is equal to 1 if the operation code is equal to 111.

If $D_7 = 1$, the instruction must be register-reference or input-output type.

If $D_7 = 0$, the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction.

Input - Output Interrupt

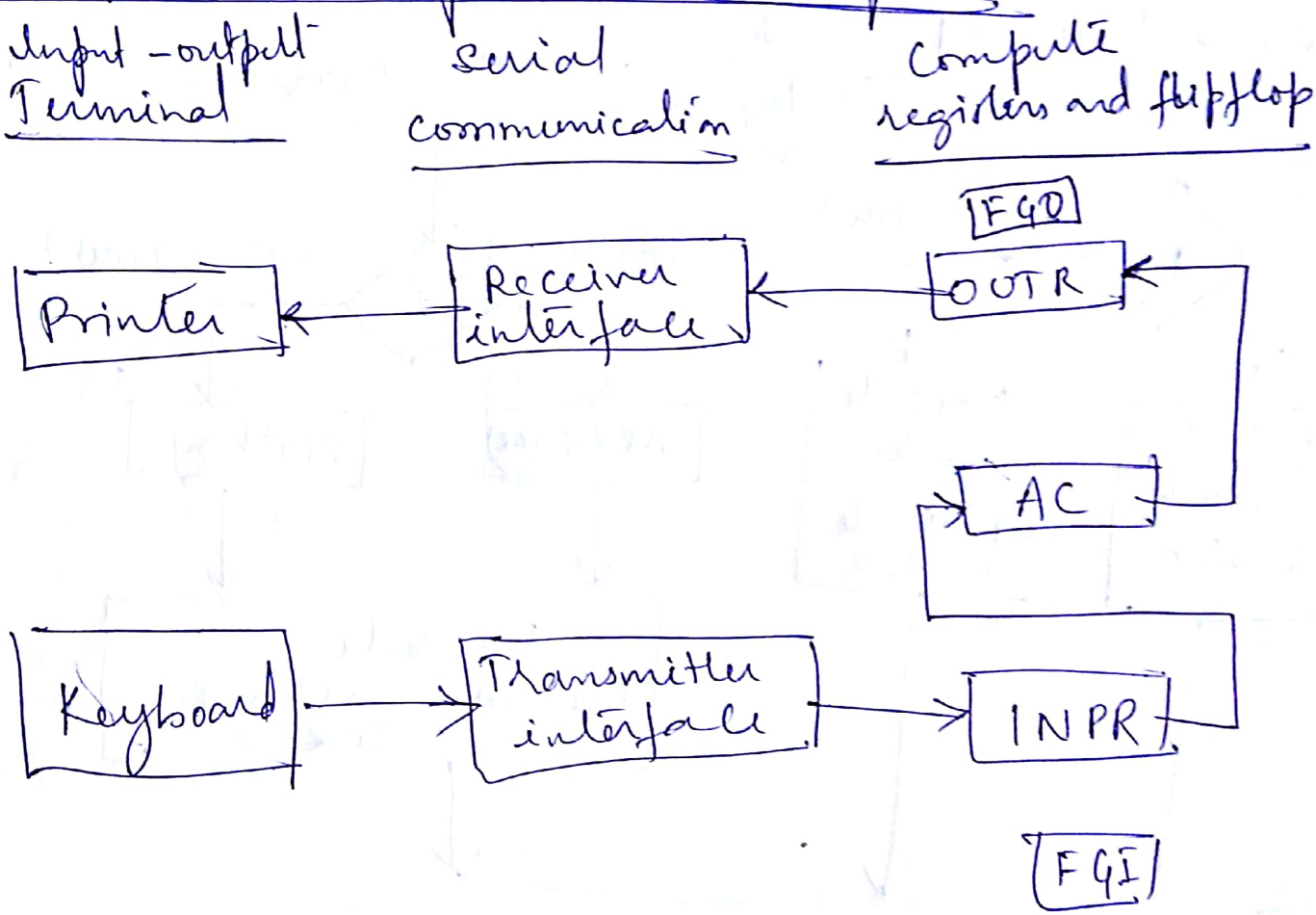


fig input - output configuration

The terminal sends and receives serial information. Each quantity of info. has eight bits of a alphanumeric code. The transmitter interface receives serial info from the keyboard and transmits it to INPR. The receiver interface receives information from OUTR and sends it to Printer serially.

The 1-bit input flag FGI is a control flipflop. the flag bit is set to 1 when new info. is available ~~int~~ in the I/P device and is cleared to 0 when the info is accepted by computer.

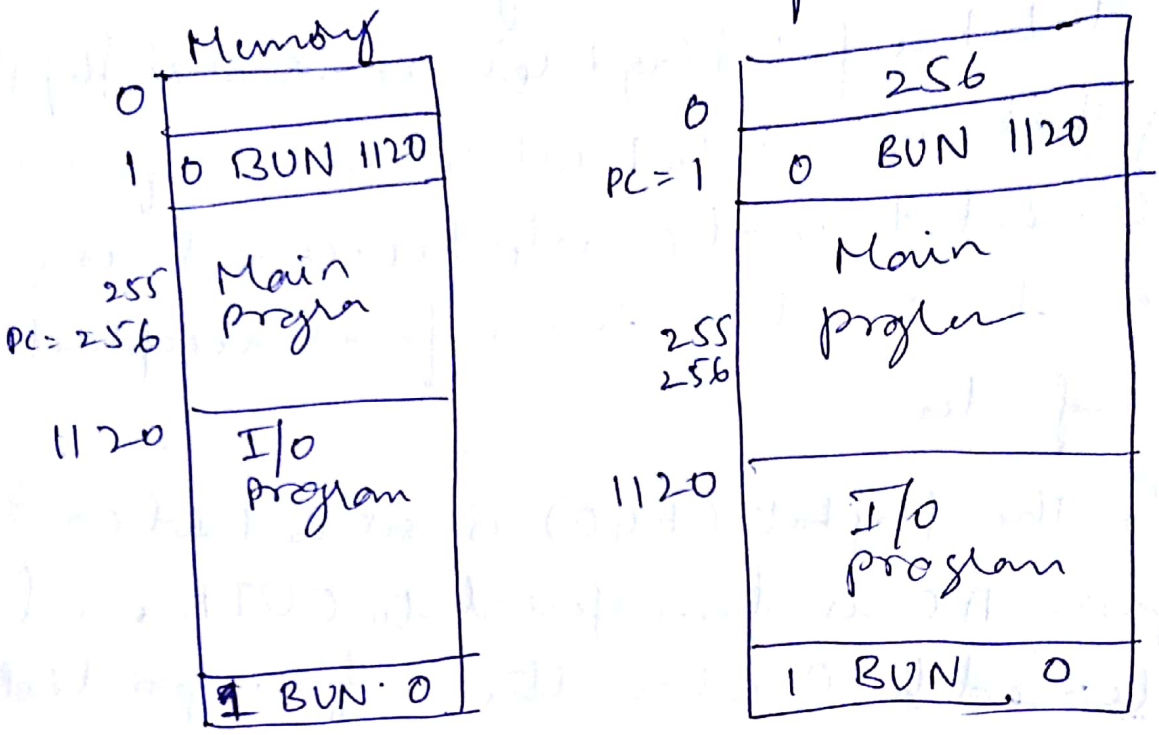
Ill^ly, The flag bit (FGO) is set to 1 when info from AC is transferred to OUTR and it is cleared to 0 when the info is printed by printer.

∴ The computer keeps checking the flag ~~bit~~ bit and when it finds it set, it initiates an info. transfer. The difference of info flow rate between the computer and that of I-O device makes this type of transfer inefficient.

An alternative procedure is let the external device inform the computer when it is ready for the transfer.

This type of transfer uses Interrupt

The interrupt cycle is a hardware implementation of a branch and save return address operation. The return address available in PC is stored in a specific location where it can be found later when the program returns to the instruction at which it was interrupted.



(a) Before Interrupt (b) After interrupt

A interrupt occurs while the control is executing the instruction at address 255. At this time, the return address 256 is in PC. The Programmer has previously placed an I-O service program in memory at 1120 address location and a BUN 1120 instruction at address 1.

Now, the content of PC (256) is stored at mem. loc '0' and PC is set to 1. The branch instruction at address 1 causes the program to transfer to the I-O program at 1120 address loc. Once it is done, the program returns to the location where it was interrupted.

Morris Mano – 8 chapter 5

Stack Organization

A stack is a storage device that stores information in such a way that the item stored last is the first item retrieved.

→ The ~~operation~~ function of stack is Last-in, first-out (LIFO).

→ The register that holds the address of stack is called stack Pointer (SP).

The two operation of a stack are
- → Push (insert)
- → POP (delete).

These operations are stimulated by incrementing or decrementing the stack pointer register.

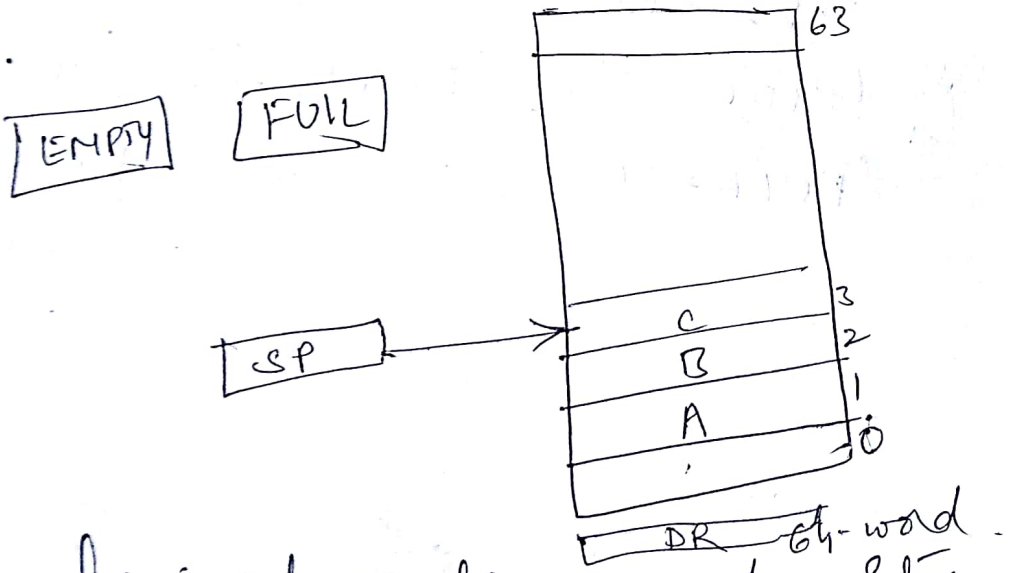


fig: Block diagram of a Stack

Initially, stack pointer is cleared to 0, EMPTY is set 1 and FULL is cleared to 0, so that SP points to address '0'. a new item is inserted with a PUSH opn: like

SP ← SP + 1 (increment stack Pointer)

M[SP] ← DR (write item on top of stack)

If (SP = 0) then (FULL ← 1) (check the if stack is full)

EMPTY ← 0 (Mark the stack not empty).

A new item is deleted from the stack if the stack is not empty.

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

SP ← SP − 1 (Decrement SP)

If (SP = 0) then (EMPTY ← 1) (check if stack is empty)

FULL ← 0 (Mark stack not full).

A stack organization is very effective for evaluating arithmetic expressions.

Infix Notation :- Operator is placed between the operands.
Ex: A+B.

Prefix Notation :- operator is placed before operands
Ex: +AB

Post fix (or) Polish Notation :- operator is placed after operands.
Ex: AB+

Reverse Polish Notation

(A*B) + (C*D)

AB* CD* +

3*4 +5*6

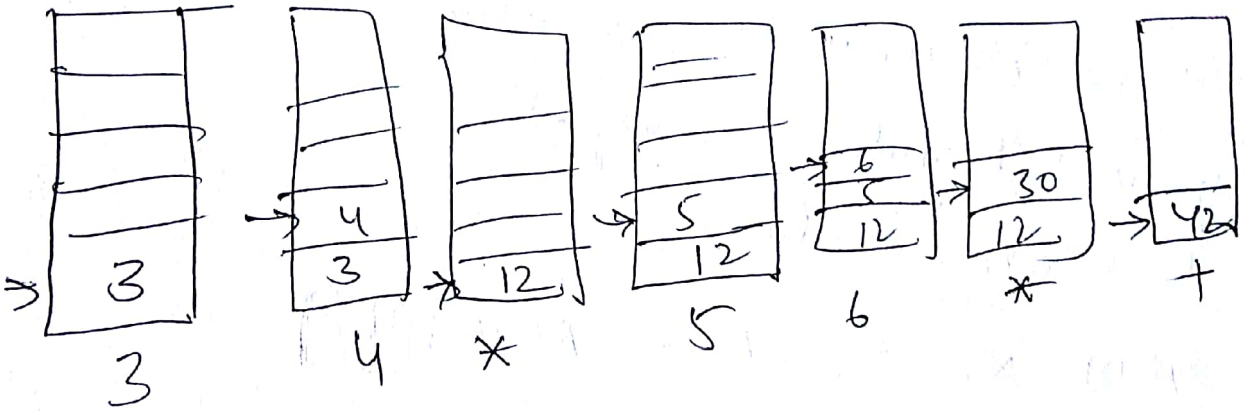


fig: Stack operations to evaluate (3*4) +(5*6)

Instruction formats

The most common field in instruction formats are:

1. An operation code
2. Address field
3. Mode field.

The no of address fields in a instruction format of a computer depends on the internal organization of its registers. Most computers fall into one of three types of CPU organization

1. Single Accumulator organization
2. General Register organization
3. Stack Organization.

→ Ex: ADD X. ⇒ AC ← AC + M[X].

where X is the address of operand.

→ Ex: ① ADD R_1, R_2, R_3 ⇒ $R_1 \leftarrow R_2 + R_3$

② ADD R_1, R_2 ⇒ $R_1 \leftarrow R_1 + R_2$

→ Ex: PUSH X.

where X is top of stack.

There are four different instruction formats

5

(1) zero address instruction

(2) One - address instruction

3. Two - address instruction

4. Three - address instruction.

Now, we evaluate the arithmetic expression $X = (A+B) * (C+D)$ using zero, one, two, and three address instruction.

(1) ~~Zero~~ Three address instruction

ADD R_1, A, B	$R_1 \leftarrow M[A] + M[B]$
ADD R_2, C, D	$R_2 \leftarrow M[C] + M[D]$
MUL X, R_1, R_2	$M[X] \leftarrow R1 * R2$.

Advantage is short program

Disadvantage is too many bits to specify three addresses.

(2) Two - address instruction — Address field specify processor registers or memory.

MOV R1, A	$R_1 \leftarrow M[A]$
ADD R1, B	$R_1 \leftarrow R1 + M[B]$
MOV R2, C	$R_2 \leftarrow M[C]$
ADD R_2, D	$R_2 \leftarrow R_2 + M[D]$
MUL R1, R2	$R1 \leftarrow R1 * R2$
MOV X, R1	$M[X] \leftarrow R1$

One-address Instruction :- uses implied AC register for all data manipulation.

LOAD	A	AC ← M[A]
ADD	B	AC ← AC + M[B]
STORE	T	M[T] ← AC
LOAD	C	AC ← M[C]
ADD	D	AC ← AC + M[D]
MUL	T	AC ← AC * M[T]
STORE	X	M[X] ← AC.

→ Here we need a temporary register to store intermediate results.

Zero address Instruction :- A stack organization does not need an address field for instruction

PUSH A	TOS ← A
PUSH B	TOS ← B
ADD	TOS ← (A+B)
PUSH C	TOS ← C
PUSH D	TOS ← D
ADD	TOS ← (C+D)
MUL	TOS ← (C+D) * (A+B)
POP X	M[X] ← TOS

TOS stands for Top of stack

Here we have to convert the expression into reverse polish notation ie X = AB+CD+*

RISC instructions :- A program for

RISC - type consists of LOAD and STORE instructions that have one memory and one register address. [Ex LOAD, R1, A
STORE X, R1]

→ Computation - type instructions have three addresses with all specifying processor register (Ex: ADD R_1, R_2, R_3)

LOAD R1, A	R1 ← M[A]
LOAD R_2, B	R_2 ← M[B]
LOAD R_3, C	R_3 ← M[C]
LOAD R_4, D	R4 ← M[D]
ADD R1, R1, R2	R1 ← R1 + R2
ADD R3, R3, R2	R3 ← R3 + R4
MUL R1, R1, R3	R1 ← R1 × R3
STORE X, R1	M[X] ← R1

Addressing Modes

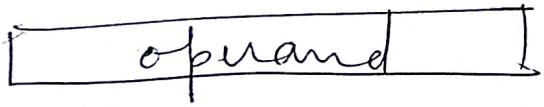
The way the operands are chosen during program execution is dependent on the addressing mode of instruction. The addressing mode specifies a rule ~~of~~ for interpreting & modifying the address field of the instruction before the operand is actually referenced.

① **Implied mode:-** The operands are specified implicitly in the definition of instruction.

Ex:- Complement AC, Zero-address instructions like PUSH, POP. (Top of stack)

② **Immediate mode:-** The operand is specified in the instruction itself. (or) An immediate-mode instruction has an operand field rather than address field.

Instruction



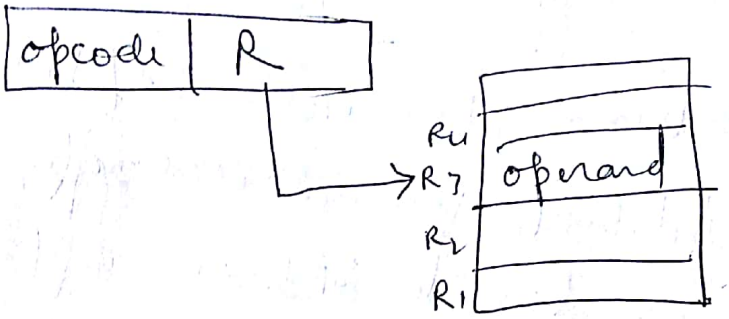
EA = operand

Effective Address: It is defined to be the memory address obtained from the computation dictated by the given addressing mode.

Register Mode:- [Usually, the address a field of an instruction may specify either a memory word or a processor register.]

The operands are in registers that reside with in the CPU.

~~Register Ind~~

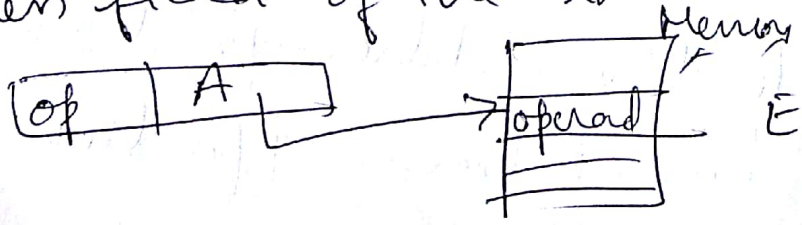


EA = R.

Register Indirect:- The ~~register~~ instruction specifies a register in the CPU whose contents give the address of the operand in memory.

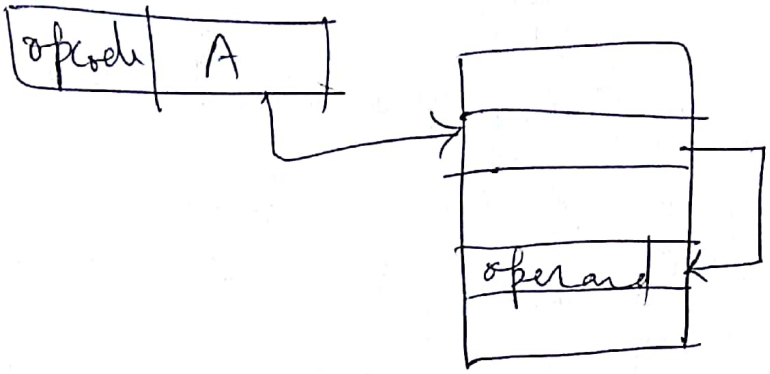
EA = (R)

Direct Address:- The operand resides in memory and its address is given directly by the address field of the instruction



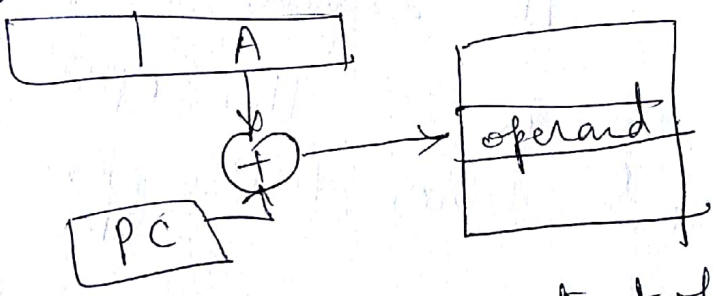
EA = A.

Indirect address :- The address field of the instruction gives the address where the effective address is stored in memory. 10



$EA = (A)$

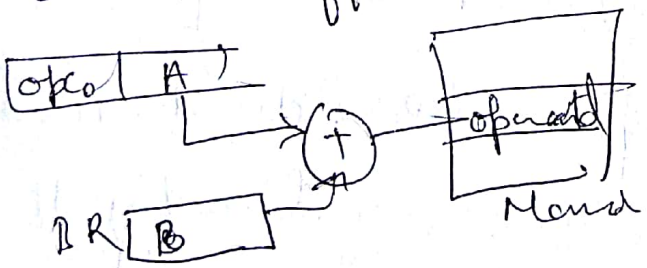
Relative address :- The content of the Program counter is added to the address part of the instruction in order to obtain effective address



$EA = A + (PC)$

Base Register :- The content of base register is added to the address part of the instruction to obtain effective address

$EA = A + (BR)$



Index Register : the content of an Index register is added to the address part of the instruction to obtain effective address

$$EA = A + (IR)$$ Memory

Data Transfer and Manipulation

Computer instruction can be classified into three categories

1. Data Transfer instructions
2. Data Manipulation Instructions
3. Program control Instruction.

Data Transfer

Name	Mnemonic	
LOAD	LD	— is used memory to Processor Reg usually AC.
Store	ST	— Processor Reg into memory.
Move	MOV	— one Reg to another reg. between CPU register & memory & between two memory words.
Exchange	XCH	
INPUT	IN	
OUTPUT	OUT	
Push	PUSH	
Pop	POP	

Data Manipulation Instructions: They are divided into three types.

① Arithmetic ② logic & bit manipulation ③ Shift.

Arithmetic Instruction

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate (2's complement)	NEG.

Logical Instructions

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive OR	X-OR
clear carry	CLRC
Set carry	SETC
complement carry	COMC
Enable input	EI
Disable input	DI

Shift Instructions

Name	Mnemonic
Logical shift right	SHR
" " left	SHL
Arithmetic shift right	SHRA
" " left	SHLA
Rotate Right	ROR
Rotate left	ROL
Rotate right thru' carry	RORC
Rotate left thru' carry	ROLC

Program Control

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (by subtraction)	CMP
Test (by AND Diy)	TST

UNIT - III

Micro Programmed Control

Control Memory :-

The function of the control unit in a digital computer is to initiate sequence of micro operations. The complexity of the digital system is derived from the number of sequences of micro operations that are performed.

When the control signals are generated by hardware using conventional logic design techniques, the control unit is said to be 'hardwired'. Microprogramming is a second alternative for designing the control unit of a digital computer. The principle of microprogramming is an elegant and systematic method for controlling the microoperation sequences in a digital computer.

The control function that specifies a microoperation is a binary variable. When it is in one binary state, the corresponding microoperation is executed. The active state of a control variable may be either the '1' state or the '0' state. In a bus-organized system, the control signals that specify microoperations are group of bits that select the paths in multiplexers, decoders & ALU's.

The control unit initiates a series of sequential steps of microoperations. During any given time, certain microoperations are to be initiated, while others remains idle. The control variables at any given time can be represented by string of

www.jntuworldupdates.org

of 1's & 0's called a "control word". A control unit whose binary variables are stored in memory is called a "microprogrammed control unit". Each word in control memory contains with in it a "micro instruction". A sequence of micro instructions constitutes a "microprogram". The control memory is a read-only memory (ROM), because no alterations are to be taken. ROM words are made permanent during the h/w production of the unit. The content of the word in ROM at a given address specifies a micro instruction

A more advanced development known as dynamic microprogramming permits a microprogram to be loaded initially from an auxiliary memory such as a magnetic disk. Control units that use dynamic microprogramming employ a writable control memory. This type of memory can be used for writing but is mostly for reading. A memory that is part of a control unit is referred to as a "Control memory".

A computer that employs a microprogrammed control unit have two separate memories.

1) Main Memory.

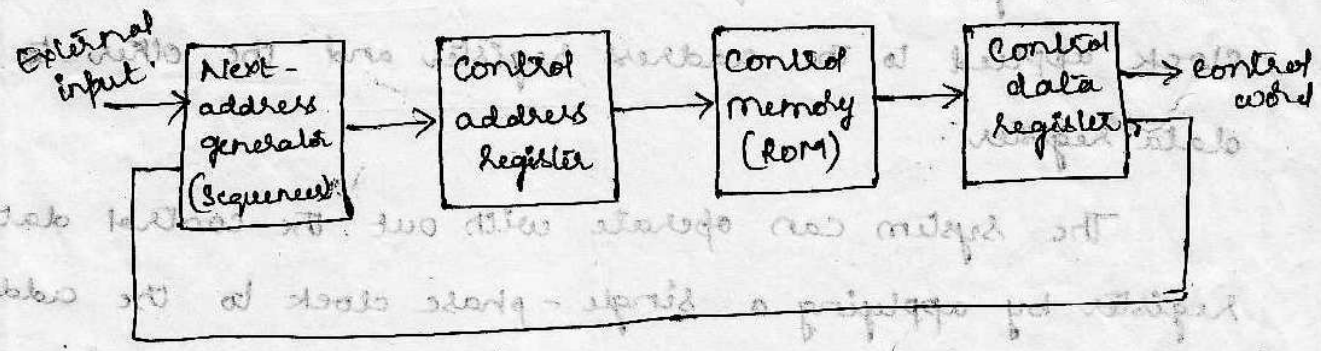
2) Control Memory.

The Main Memory is available to the user for storing the programs.

The Control Memory holds a fixed microprogram that can't be altered by the occasional user.

www.jntuworldupdates.org

The general configuration of a microprogrammed control unit is shown in below fig.



The control memory is assumed to be a ROM, within which all control information is permanently stored. The control address register specifies the address of the micro instruction and the control data register holds the micro-instruction read from memory.

The microinstruction contains a control word that specifies one or more microoperations for the data processor. Once these operations are executed, the control must determine the next address. The next address may also be a function of external input conditions. The microoperations are being executed, the next address is computed in the next address generator circuit and then transferred into the control address register to read the next microinstruction.

The next address generator is sometimes called a microprogram "sequencer". The functions of a microprogram sequencer are incrementing the control address register by one.

www.specworld.in
www.smartzworld.com

www.jntuworldupdates.org

The control data register holds the present micro instruction while the next address is computed and read from memory. The data register is sometimes called a "pipeline register". This configuration requires a two-phase clock, with one clock applied to the address register and the other to the data register.

The system can operate with out the control data register by applying a single-phase clock to the address register. The control word and next-address information are taken directly from the control memory. It must be realized that a ROM operates as a combinational circuit and the content of the specified word in ROM remains in the o/p wires as long as its address value remains in the address register.

The other two components: the sequencer and the control memory are combinational circuits & do not need a clock.

The main advantage of microprogrammed control is the fact that once the hardware configuration is established, there should be no need for h/w or wiring changes.

RISC architecture concept use hardwired control rather than a control memory with a microprogram.

www.jntuworldupdates.org

Address Sequencing :-

Micro instructions are stored in control memory in groups, with each group specifying a routine. Each computer has its own microprogram routine in control memory to generate the micro operations that execute the instruction. The hardware that controls the address sequencing of the control memory must be capable of sequencing the microinstruction with in a routine and be able to branch from one routine to another.

An initial address is loaded into the control address Register. The address is usually the address of the first microinstruction that activates the instruction fetch routine. The fetch routine is incrementing the control address register. At the end of fetch routine, the instruction is in the instruction register of the computer.

The control memory must go through the routine that determines the effective address of the operand. The next step is to generate the micro operations that execute the instruction fetched from memory. The transformation from the instruction code bits to an address in control memory where the routine is located is referred to as a "mapping process".

Micro programs that employ subroutines will require an external register for storing the return address. Return addresses can't be stored in ROM.

When the execution of the instruction is completed, control must return to the fetch routine.

The address sequencing capabilities required in a control memory are:

www.specworld.in

www.smartzworld.com

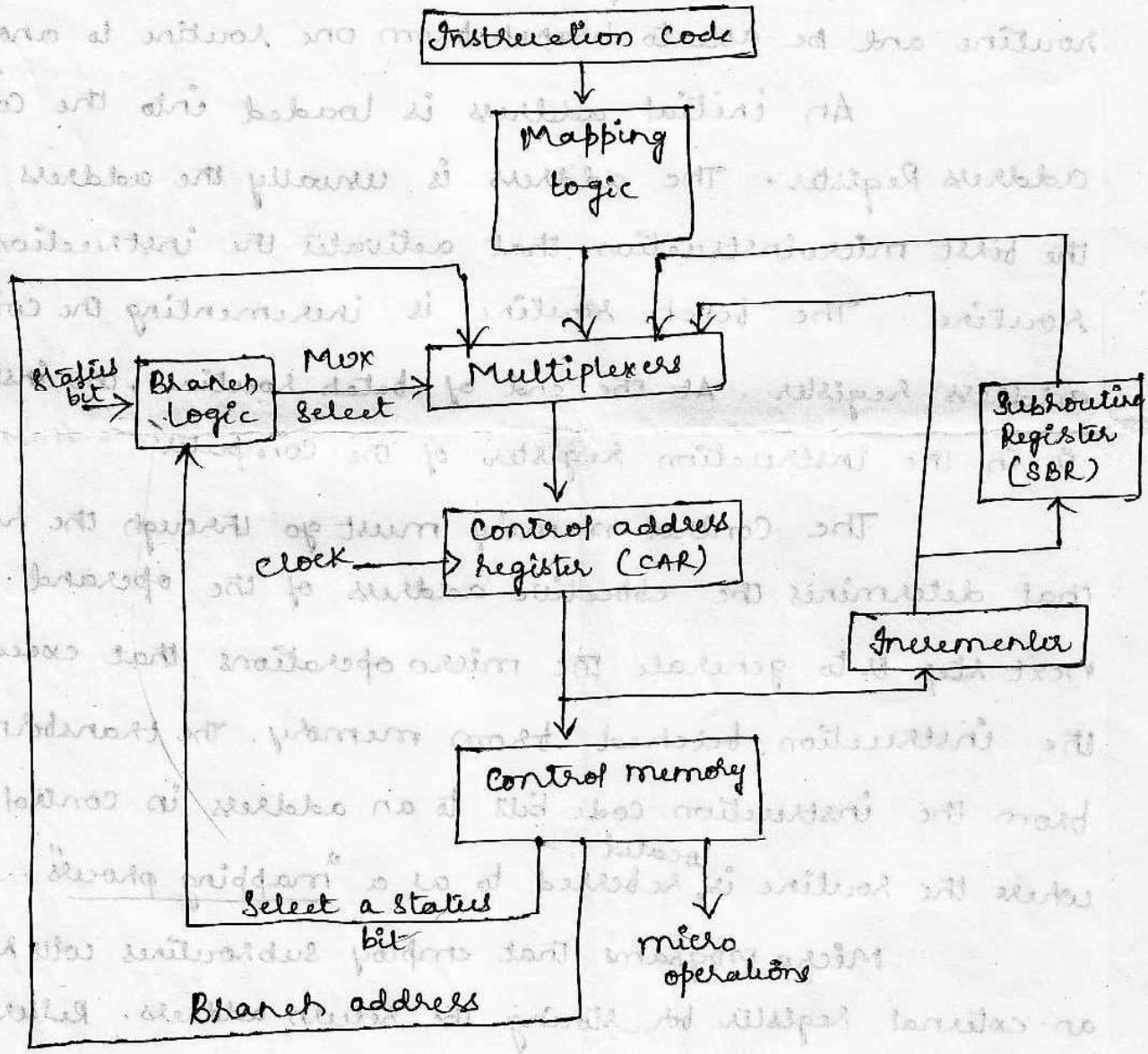
www.jntuworldupdates.org

1) Incrementing of the control address register

2) unconditional branch or conditional branch, depending on status bit conditions.

3) A mapping process from the bits of the instruction to an address for control memory.

4) A facility for subroutine call & return.



The fig. shows a block diagram of a control memory and the associated h/w needed for selecting the next micro instruction address.

The diagram shows four different paths from which the CAR receives the address. The incrementer increments

www.specworld.in
www.smartzworld.com

the content of the CAR by one, to select the next micro-instruction in sequence. Branching is achieved by specifying the branch address in one of the fields of the micro instruction. Conditional branching is obtained by using part of the micro instruction to select a specific status bit. An external address is transferred into control memory via mapping logic circuit. The return address for a subroutine is stored in a special reg. whose value is then used when the micro program wishes to return from the subroutine.

Conditional Branching :-

The status conditions are special bits in the system, such as the carry-out of an adder, the sign-bit of a number, the mode bits of an instruction & i/o status conditions. Information in these bits can be tested and actions initiated based on their condition : their value is 1 or 0. The conditional branch decisions generated in the branch logic.

The branch logic h/w may be implemented in a variety of ways. The simplest way is to test the specified condition & branch to the indicated address ; otherwise, the address reg. is incremented. This can be implemented with a multiplexer.

Eg: There are eight status bit conditions in the system. Three bits in the micro instr are used to specify the selection variables for the multiplexer. If the selected status bit is in the '1' state, the o/p of the multiplexer

www.specworld.in
www.smartzworld.com

www.jntuworldupdates.org

is 1, otherwise it is 0. A '1' output in the multiplexer generates a control ~~register~~ signal to transfer the branch address from the micro instruction into the control address register. A 0 o/p in the multiplexer causes the address reg is to be incremented.

An unconditional branch micro instruction can be implemented by loading the branch address from control memory into the control address reg.

Sub-routines :

Sub-routines are programs that are used by other routines to accomplish a particular task.

Micro programs that use subroutines must have a provision for storing the return address during a sub-routine call and restoring the address during a sub-routine return. The subroutine reg. can then become the source for transferring the address for the return to the main ~~memory~~ routine. The best way to structure a reg. file that stores addresses for sub-routine is to organize the reg. in a last-in, first-out (LIFO) stack.

Mapping Instruction :-

A special type of branch exists when a micro instruction specifies a branch to the first word in control memory where a micro pgm routine for an instruction.

For eg, a computer with a simple instruction format as shown in fig.

www.jntuworldupdates.org

Computer instruction: | 1 0 1 1 | Address |

Mapping bits : 0 | x x x x | 00

Micro instruction address: | 0 1 0 1 1 0 0 |

Assume further that the control memory has 128 bits. This simple mapping process that converts the 4-bit operation code to a 7-bit address of control memory as shown in fig.

This mapping consists of placing a '0' in the MSB of the address, transferring the four operation code bits & clearing the two least significant bits of the control address register.

If the routine needs more than four micro instructions, it can use addresses 1000000 through 1111111. If it uses fewer than four micro instructions, the unused memory locations would be available for other routines.

One can extend this concept to a more general mapping rule by using a ROM to specify the mapping function. The bits of the instr specify the address of a mapping ROM. The contents of the mapping ROM give the bits for the control address register.

The mapping concept provides flexibility for adding instructions for control memory.

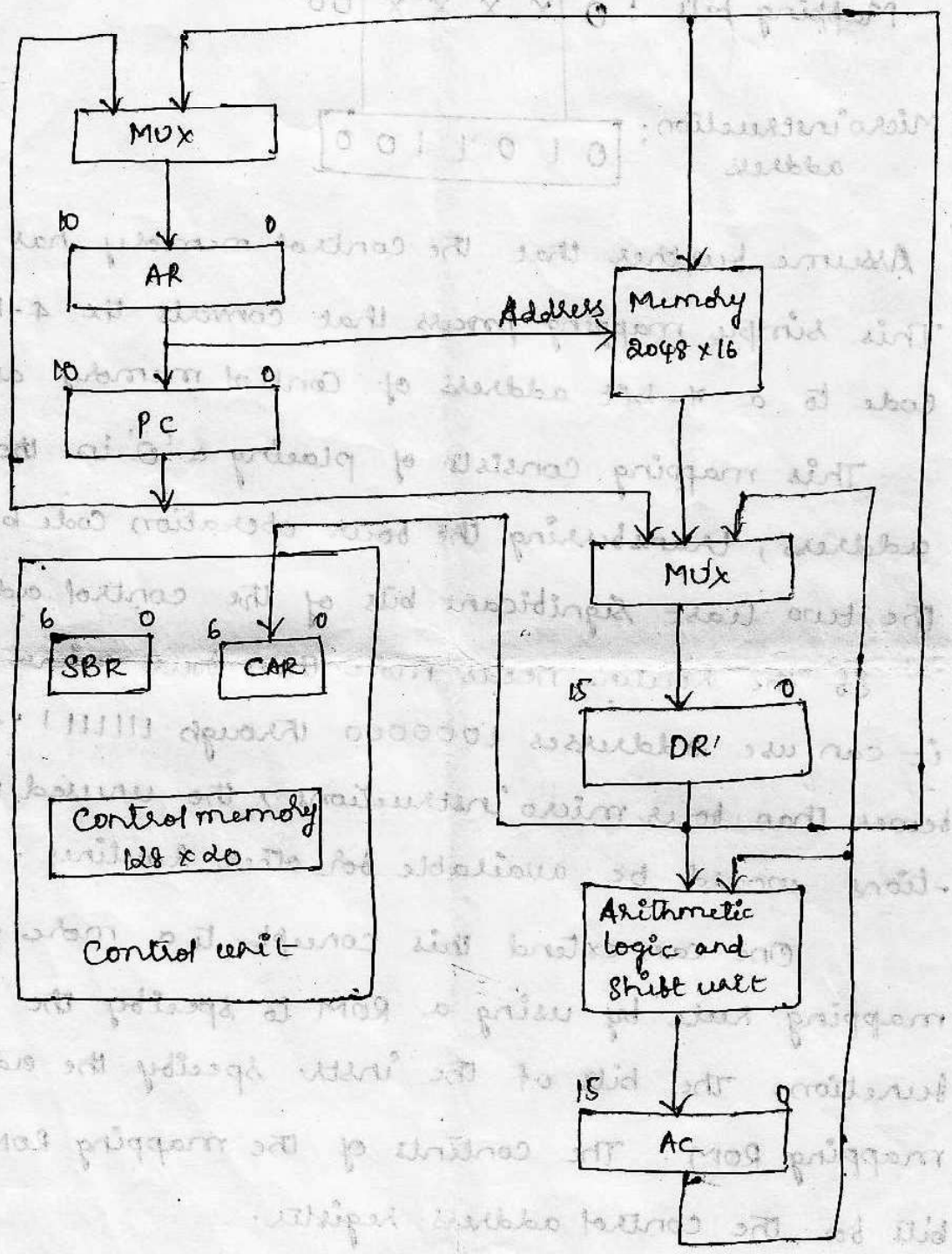
The mapping function is implemented by means of an integrated circuit called PLD. This can be expressed in Boolean expressions.

www.jntuworldupdates.org

Microprogram Example :-

Computer configuration :-

The block diagram of the computer is shown in below fig.



It consists of two memory units : a main memory for storing instructions and data, and a control memory for storing the micro program. Four registers are associated with the processor unit & two with the control unit. The processor

registers are program counter (PC), address register (AR), data register (DR) and accumulator (AC).

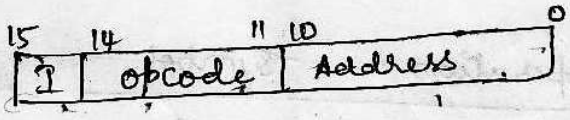
The control unit has a control address register CAR and a subroutine register (SBR).

The transfer of information among the registers in the processor is done thro' multiplexer

DR can receive information from AC, PC & memory.

AR can receive information from PC & DR. PC can receive information from AR. The arithmetic, logic and shift unit performs micro operations with data from AC and DR and places the result in AC. Input data written to memory come from DR, and data read from memory can go only to DR.

The computer instruction format is shown in beq. It consists of three fields.



1) 'I' bit is for indirect A.M. 2) 4-bit opcode and 3) 11-bits for address.

The below lists four of the sixteen possible memory-reference instructions.

Symbol	opcode	Description
ADD	0000	$AC \leftarrow AC + M[EA]$
BRANCH	0001	If $(AC < 0)$ then $(PC \leftarrow EA)$
STORE	0010	$M[EA] \leftarrow PC$.
EXCHANGE	0011	$AC \leftarrow M[EA]$, $M[EA] \leftarrow AC$.

www.jntuworldupdates.org

Microinstruction format :-

The microinstruction format for the control memory is shown in fig.

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

The 20-bits of the microinstruction are divided into four functional parts. The three fields F1, F2 & F3 specify microoperations for the computer. The CD is condition for branching which is used to select the status bit conditions. The BR is Branch field specifies the type of branch to be used. The AD field contains a branch address (Address field). The address field is seven bits, since the control memory has $128 = 2^7$ words.

The microoperations are subdivided into three fields of three bits.

The each three bits specify the seven distinct micro operations

F1	microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC + DR$	ADD
010	$AC \leftarrow 0$	CLRAC
011	$AC \leftarrow AC + 1$	INCAC
100	$AC \leftarrow DR$	DRTAC
101	$AR \leftarrow DR(0-10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE

www.specworld.in

www.smartzworld.com

F2	Micro operation	Symbol
000	None	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR + 1$	INCDR
111	$DR(0-10) \leftarrow PC$	PCTDR.

F3	microoperation	Symbol.
000	None	NOP
001	$AC \leftarrow AC \oplus DR$	XOR
010	$AC \leftarrow \overline{AC}$	COM
011	$AC \leftarrow shl\ AC$	SHL
100	$AC \leftarrow shr\ AC$	SHR
101	$PC \leftarrow PC + 1$	INCPC
110	$PC \leftarrow AR$	ARTPC
111	Reserved.	

CD	Condition	Symbol	Comments
00	Always = 1	U	unconditional branch
01	DR(15)	I	Indirect address bit
10	AC (15)	S	Sign bit of AC
11	AC=0	Z	Zero value in AC.

www.jntuworldupdates.org

BR	Symbol	Function
00	JMP	CAR ← AD if Condition = 1
		CAR ← CAR + 1 if Condition = 0.
01	CALL	CAR ← AD, SBR ← CAR + 1 if Condition = 1
		CAR ← CAR + 1 if Condition = 0.
10	RET	CAR ← SBR (Return from Subroutine)
11	MAP	CAR (2-5) ← DR (11-14),
		CAR (0,1,6) ← 0.

We can specify two simultaneous operation from F2 & F3 and none from F1.

DR ← M[AR] with F2 = 100

PC ← PC+1 with F3 = 101.

The nine bits of the microoperations fields will then be 000 100 101.

All transfer type micro operations symbols use five letters. The first two letters represents the source register, the third letter is T and the last two letters represents the destination register.

Eg: DRTAC. means DR to AC.

The CD (Condition) field consists of two bits needed to specify four status bit conditions.

We will use the symbols U, I, S & Z for the four status bits when we write micro programs in symbolic form.

The BR (Branch) field consists of two bits. It is used in conjunction with the address field (AD) to choose the address of the next micro instruction.

www.jntuworldupdates.org

Symbolic microinstructions :-

A symbolic microprogram can be translated into its binary equivalent by means of an assembler. The simplest way to formulate an assembly language for a microprogram is to define symbols for each field of the microinstruction and defining their own symbolic address.

Each symbolic microinstruction is divided into five fields : label, microoperations, CD, BR & AD.

1) The label field may be empty or it may specify a symbolic address. Label is terminated with colon.

2) The microoperations field consists of one, two, three symbols, separated by commas. There may be no more than one symbol from each F field. The NOP symbol is used when there is no microoperation and will be translated by the assembler to nine zeros.

3) The CD field has one of the letters U, I, S or Z.

4) The BR field contains one of the four symbols.

5) The AD field specifies a value for the address field

of the microinstruction in one of three possible ways.

a) with symbolic address, which must also appear as label.

b) with the symbol NEXT to designate the next address in sequence

c) When the BR field contains a RET or MAP symbol, the AD field is left empty & is converted to 7 zeros by assembler.

We will also use the pseudo instruction ORG to define the origin (first address). The ORG 64 informs the assembler to place the next micro instruction in control memory at decimal address 64, which is equivalent to the binary address 1000000.

The Fetch Routine:-

The control memory has 128 words and each word contains 20 bits. ~~To~~ The first 64 words are to be occupied by the routines for the 16 micro instructions and the last 64 words may be used for another purpose.

The micro instructions needed for the fetch routine are:

$AR \leftarrow PC$

$DR \leftarrow M[AR],\ PC \leftarrow PC+1$

$AR \leftarrow DR(0\text{-}10),\ CAR(2\text{-}5) \leftarrow DR(11\text{-}14),\ CAR(0,1,6) \leftarrow 0.$

The address part is transferred to AR and then control is transferred to one of 16 routines by mapping the operation code part of the instruction from DR into CAR.

The fetch routine needs three micro instructions which are placed in control memory at addresses 64, 65 & 66.

www.jntuworldupdates.org

We can write the symbolic micro program for the fetch routine as follows:

```
        ORG 64
FETCH : PCTAR          U   JMP   NEXT
        READ, INCPC    U   JMP   NEXT
        DRTAR          U   JMP
```

The translation of the symbolic microprogram to binary produces the following binary microprogram.

Binary address	F1	F2	F3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	000	000	00	11	0000000

The three microinstructions that constitute the fetch routine have been listed in three different representations. The symbolic rep. is useful for writing microprograms in an assembly language format. The binary rep. is actually stored in control memory.

Symbolic microprogram :-

The execution of the MAP micro instruction in the fetch routine results in a branch to address 0XXX00, where XXX are the four bits of the operation.

The first address for the BRANCH and STORE routines are 0 0001 00 (decimal 4) and 0 0010 00 (decimal 8) respectively. The first address for the other 13 routines are at address values 12, 16, 20, ---- 60.

The indirect address mode is associated with all memory-reference instructions. The subroutine is shown in fig

www.jntuworldupdates.org

INDRCT :	READ	U	JMP	NEXT
	DRTAR	U	JMP	NEXT.

The transfer and return from the indirect subroutine occurs, assume that MAP instruction at the end of the fetch routine caused a branch to address 01 where the ADD routine is stored. The first micro instruction call subroutine INDRCT conditioned on status bit.

The INDRCT subroutine has two micro instructions.

	ORG 64			
FETCH :	PCTAR	U	JMP	NEXT
	READ, INCPC	U	JMP	NEXT
	DRTAR	U	~~JMP~~ MAP	
INDRCT :	READ	U	JMP	NEXT
	DRTAR	U	RET	

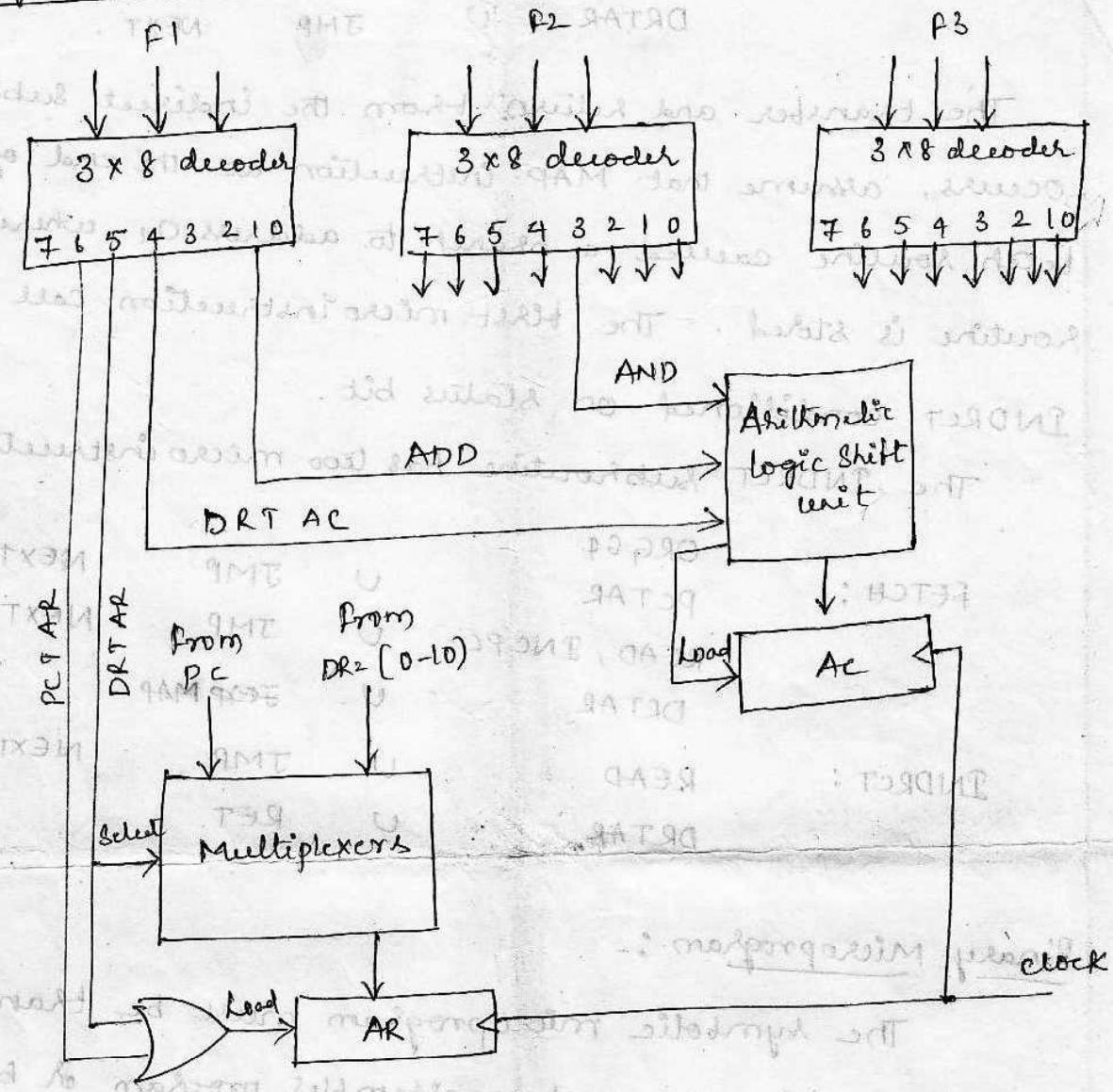
Binary Microprogram :-

The symbolic microprogram must be translated to binary either by means of an assembler program or by the user if the microprogram is simple.

The equivalent binary form of the microprogram is listed in table

www.jntuworldupdates.org

Design of a control unit :-



The bits of the microinstruction are usually divided into fields, with each field defining a distinct, seperate function. The various fields encountered in instruction formats provide control bits to initiate microoperations in the system, special bits to specify the way that the next address is to be evaluated, and an address field for branching. The number of control bits that initiate microoperations can be reduced by grouping mutually exclusive variables into fields and encoding the K bits in each field to provide 2^K microoperations.

Each field requires a decoder to produce the corresponding control signals. This method reduces the size of the microinstruction bits but requires additional hardware external to the control memory. It also increases the delay time of the control signals because they must propagate through the decoding circuits.

The encoding of control bits was demonstrated in the programming example of the preceding section. The nine bits of the microoperation field are divided into three subfields of three bits each. The control memory output of each subfield must be decoded to provide the distinct microoperations. The outputs of the decoders are connected to the appropriate inputs in the processor unit.

Fig. shows the three decoders and some of the connections that must be made from their outputs. Each of the three fields of the microinstruction presently available in the output of control memory are decoded with a 3x8 decoder to provide eight outputs. Each of these outputs must be connected to the proper circuit to initiate the corresponding microoperation as specified.

For example, when F1 = 101 (binary 5), the next clock pulse transition transfers the content of DR(0-10) to AR. Similarly, when F1 = 110 (binary 6) there is a transfer from PC to AR. As shown in fig. outputs 5 and 6 of decoder F1 are connected to the load input of AR so that when either one

www.jntuworldupdates.org

of these outputs is active, information from the multiplexers is transferred to AR.

The multiplexers select the information from DR when output 5 is active and from PC when 5 is inactive. The transfer into AR occurs with a clock pulse transition only when output 5 or output 6 of the decoder are active. The other outputs of the decoders that initiate transfers between registers must be connected in a similar fashion.

The arithematic logic shift unit can be designed as in figs ~~5-19~~ and ~~5-20~~. Instead of using gates to generate control signals marked by the symbols AND, ADD, and DR ~~in fig. 6-19~~ these inputs will now come from the outputs of the decoders associated with the symbols AND, ADD and DRTAC, respectively as shown in fig. The other outputs of the decoders that are associated with an AC operation must also be connected to the arithematic logic shift unit in a similar fashion.

Microprogram Sequencer :-

The basic components of a microprogrammed control unit are the control memory and the circuits that select the next address. The address selection part is called a microprogram sequencer.

A microprogram sequencer can be constructed with digital functions to suit a particular application. However, just as there are large ROM units available in integrated circuit packages, so are general-purpose sequencers suited for the construction of microprogram control units. To guarantee a wide range of acceptability, an integrated circuit sequencer must provide an internal organization that can be

www.jntuworldupdates.org

adapted to a wide range of applications. The purpose of a microprogram sequencer is to present an address to the control memory so that a microinstruction may be read and executed.

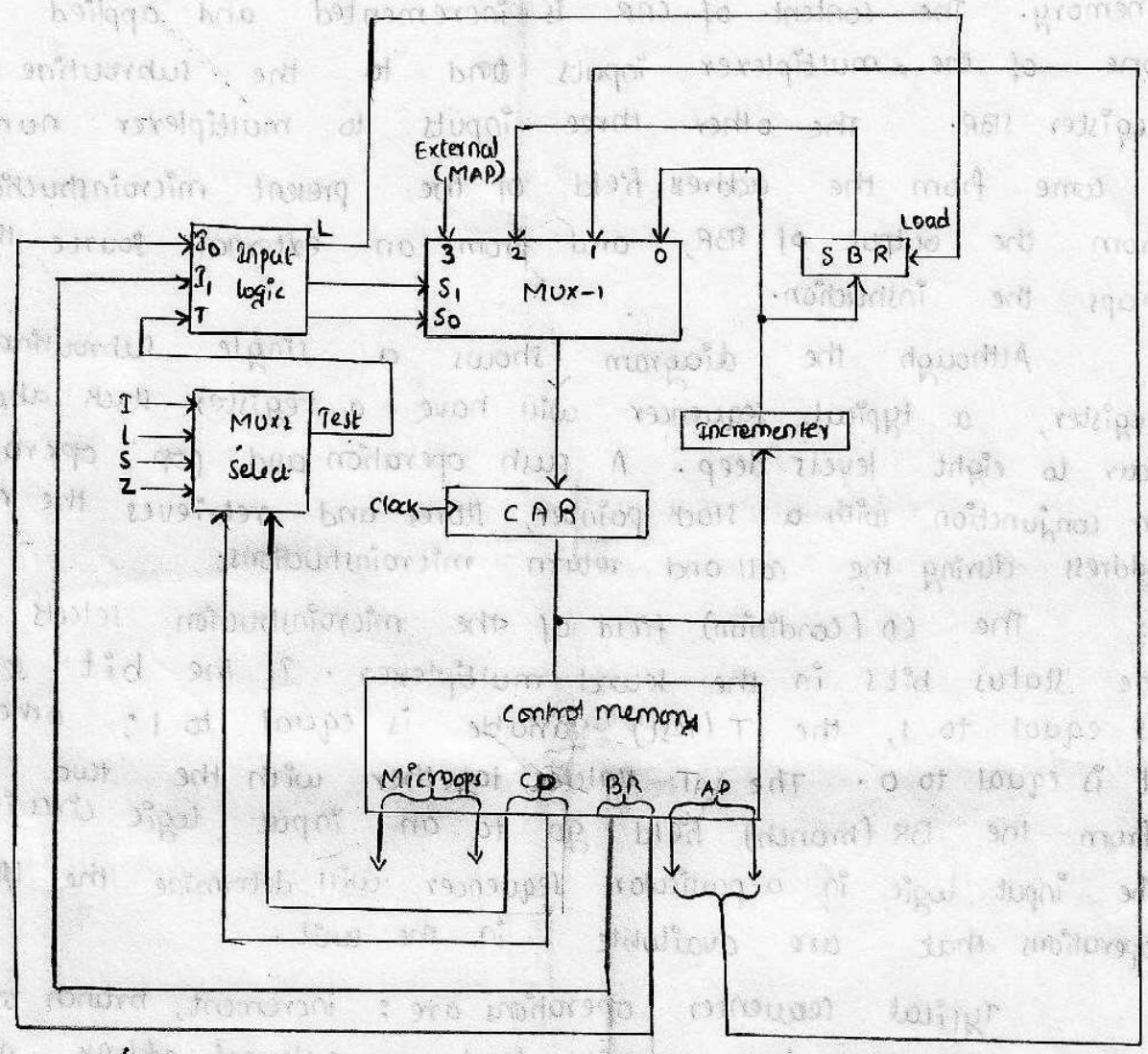


fig:- Microprogram sequencer for a control memory.

The control memory is included in the diagram to show the interaction between the sequencer and the memory attached to it. There are two multiplexers in the circuit. The first multiplexer selects an address from one of four sources and routes it into a control address register CAR. The second multiplexer tests the value of a selected status bit and the result of the test is applied to an input logic circuit.

www.jntuworldupdates.org

The output from CAR provides the address for the control memory. The content of CAR is incremented and applied to one of the multiplexer inputs and to the subroutine register SBR. The other three inputs to multiplexer number 1 come from the address field of the present microinstruction, from the output of SBR, and from an external source that maps the instruction.

Although the diagram shows a single subroutine register, a typical sequencer will have a register stack about four to eight levels deep. A push operation and pop operation, in conjunction with a stack pointer, stores and retrieves the return address during the call and return microinstructions.

The CD (condition) field of the microinstruction selects one of the status bits in the second multiplexer. If the bit selected is equal to 1, the T (test) variable is equal to 1; otherwise it is equal to 0. The T value together with the two bits from the BR (branch) field go to an input logic circuit. The input logic in a particular sequencer will determine the type of operations that are available in the unit.

Typical sequencer operations are: increment, branch or jump, call and return from subroutine, load an external address, push or pop the stack, and other address sequencing operations. With three inputs, the sequencer can provide up to eight address sequencing operation. The input logic circuit in fig. has three inputs I_0, I_1 and T, and three outputs, S_0, S_1, and L.

Variables S_0 and S_1 select one of the source addresses for CAR. Variable L enables the load input in SBR. The binary values of the two selection variables determine the path in the multiplexer. For example $S_1, S_0 = 10$ mux input number 2 is selected and establishes a transfer path from SBR to CAR.

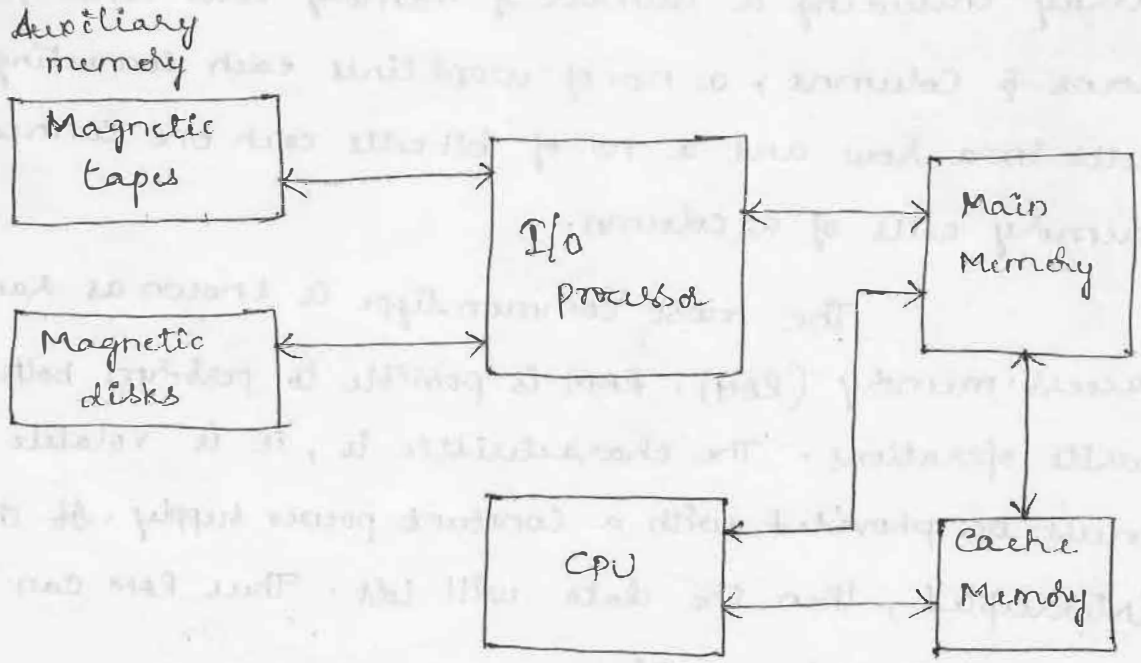
www.jntuworldupdates.org

MEMORY ORGANIZATION

Memory Hierarchy :-

The memory unit that communicates directly with the CPU is called the main memory. Devices that provide backup storage are called auxiliary memory. The most common auxiliary devices used in computer systems are magnetic disks and tapes. They are used for storing system programs, large databases and other back up information. Only programs and data currently needed by the processor reside in main memory and all other information is stored in auxiliary memory and transferred to main memory when it is needed.

The below fig. shows the components in a memory hierarchy.



The Memory hierarchy system consists of all storage devices employed in a computer system from the slow but high-capacity auxiliary memory to a relatively faster main memory and

www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org

cache memory accessible to the high-speed processing logic.

At the bottom of the hierarchy are the relatively slow magnetic tapes used to store removable files. Next are the magnetic disks are used as backup storage. The main memory occupies a central position be able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor.

When programs not residing in main memory are needed by the CPU, they are brought in from auxiliary memory.

A special very-high-speed memory called a cache. It is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate.

Semiconductor RAM Memory :-

The semiconductor memory includes a memory cell array including a number of memory cells arranged in a rows & columns, a no. of word lines each connecting all memory cells in a row and a no. of bit cells each one connecting all memory cells of a column.

The most common type is known as random access memory (RAM). RAM is possible to perform both read and write operations. The characteristic is, it is volatile. A RAM must be provided with a constant power supply. If the power is interrupted, then the data will lost. Thus RAM can be used only as temporary storage.

RAM technology has divided into two technologies

(i) Static RAM

(ii) Dynamic RAM.

www.jntuworldupdates.org

Static RAM :-

In a static RAM, binary values are stored ~~as~~ using traditional flip-flop logic gate configurations. A SRAM will holds its data as long as power is supplied to it. SRAM is used as cache memory.

Dynamic RAM :-

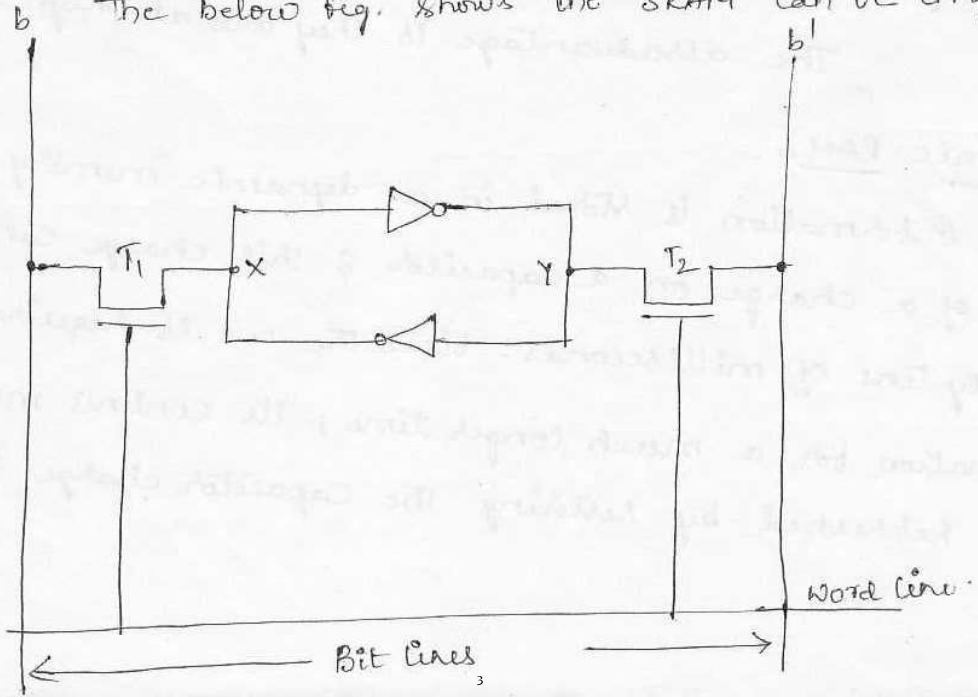
A DRAM is made with cells that store data as charge on capacitors.

DRAM's are among the densest VLSI circuit interms of transistors perchip.

Both SRAM's and DRAM's are volatile, that is the stored information is lost when the power is removed. DRAM is used as main memory. A dynamic memory cell is simpler and hence smaller than a static memory cell. Thus, a dynamic RAM is more dense and less expensive than SRAM. But SRAM's are generally faster than DRAM's.

There is an another type of memory is known as Read-only memory (ROM).

The below fig. shows the SRAM can be implemented.



In the fig., the two inverters are cross-connected to form a latch. The latch is connected to two bit lines by transistors T_1 & T_2. These transistors act as switches that can be opened or closed under the control of the word line. When the word line is at ground level, the transistors are turned off and the latch retains its state. For eg, let us assume that the cell is state 1 if the logic value at point X is 1 and at point Y is 0.

Read operation:

To read in SRAM cell, the wordline is activated to close switches T_1 & T_2. If the cell is in state 1, the signal on bit line 'b' is high & the signal on bit line b' is low.

Write operation:-

The state of the cell is set by placing the appropriate value on bit line 'b' and its complement on b' and then activating the wordline.

A major advantage of SRAM's is their very low consumption because current flows in the cell only when the cell is being accessed and also they are fast.

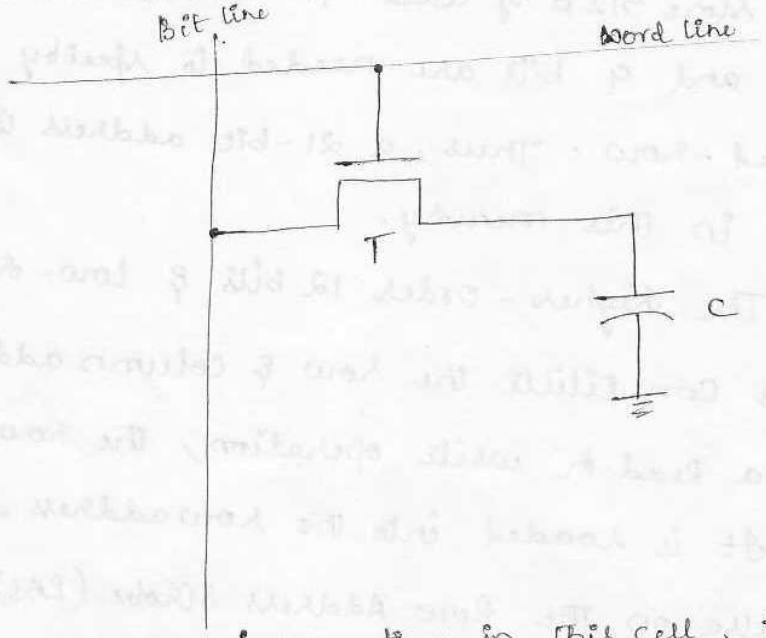
The disadvantage is they are at high cost.

Dynamic RAM's:

Information is stored in a dynamic memory cell in the form of a charge on a capacitor & this charge can be maintained for only tens of milliseconds. Since the cell is required to store information for a much longer time, its contents must be periodically refreshed by restoring the capacitor charge to its full value.

www.jntuworldupdates.org

The below fig. shows the dynamic memory cell that consists of a capacitor C and a transistor 'T'.

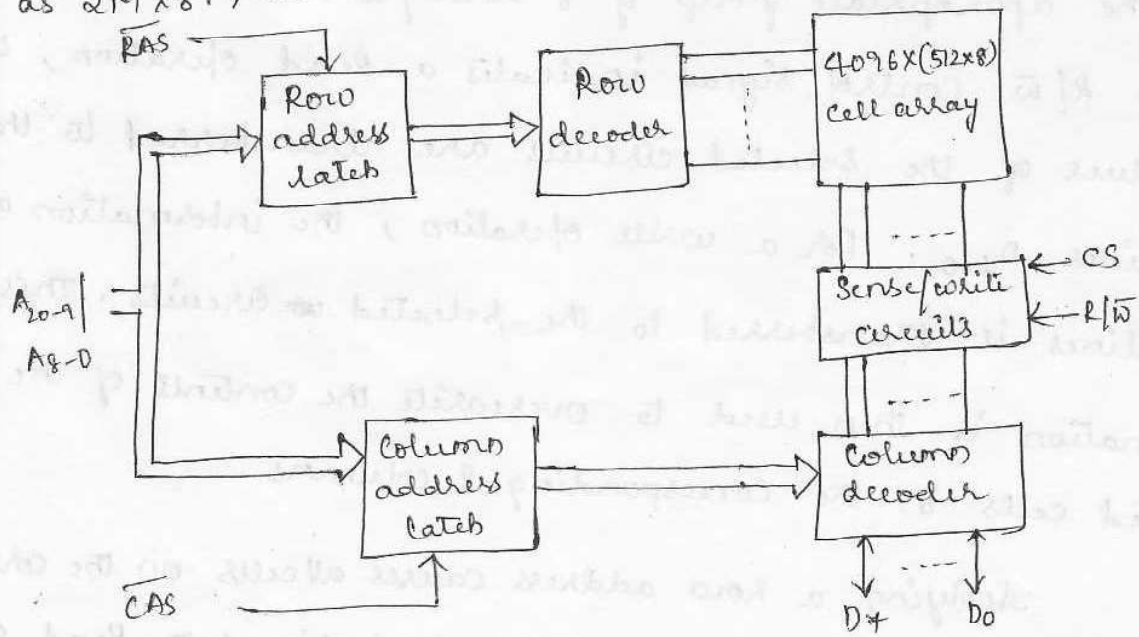


Inorder to store information in this cell, the transistor T is turned on and an appropriate voltage is applied to the bit line. This causes a known amount of charge to be stored in the capacitor.

After the transistor is turned off, the capacitor begins to discharge.

Asychronous DRAM's :-

The below fig. shows the 16-Megabit DRAM chip, configured as 2M x 8.,



www.jntuworldupdates.org

The cells are organized in the form of a 4K x 4K array. The 4096 cells in each row are divided into 512 groups of 8, so that each row can store 512B of data. 12 address bits are needed to select a row and 9 bits are needed to specify a group of 8 bits in the selected-row. Thus, a 21-bit address is needed to access a byte in this memory.

The higher-order 12 bits & low-order of 9 bits of the address constitute the row & column address of a byte.

During a Read or write operation, the row address is applied first. It is loaded into the row address latch in response to a signal pulse on the Row Address Strobe (RAS) in the chip. Then a Read operation is initiated, in which all cells on the selected row are read & refreshed.

After the row address is loaded, the column is applied to the address pins and loaded into the column address latch under the control of the column Address Strobe (CAS) signal.

The information in the latch are decoded and the appropriate group of 8 sense/write are selected. If the R/$\bar{W}$ control signal indicates a Read operation, the o/p values of the selected circuits are transferred to the data lines D_{7-0}. For a write operation, the information on the D_{7-0} lines is transferred to the selected circuits. This information is then used to overwrite the contents of the selected cells in the corresponding 8 columns.

Applying a row address causes all cells on the corresponding row to be read and refreshed during both Read &

www.jntuworldupdates.org

while operations. To ensure that the contents of a DRAM are maintained, each row of cells must be accessed periodically. A refresh counter performs this function automatically.

The timing of the memory device is controlled asynchronously. A specialized memory controller circuit provides the necessary control signals, RAS & CAS signals. The processor must take into account the delay in the response of the memory. Such memories are referred to as asynchronous DRAM's.

Advantages :-

1) It has high density & low cost, these are widely used.
2) Available chips are of size 1M to 256 Mbits & larger chips are developed.
3) A DRAM chip is organized to read or write a number of bits in parallel.
4) It provides flexibility in designing memory systems.

Fast page Mode :-

The contents of all 4096 cells in the selected row are sensed, but only 8 bits are placed in the data lines D_{7-0}. This byte is selected by the column address bits A_{8-0}.

A simple modification can make it possible, is, to access the other bytes in the same row with out having to reselect the row. A latch can be added at the o/p of the sense

www.jntuworldupdates.org

amplifier in each column.

The application of a row address will load the latches corresponding to all bits in the selected row. Then, it is only necessary to apply different column address to place the different bytes on the data lines.

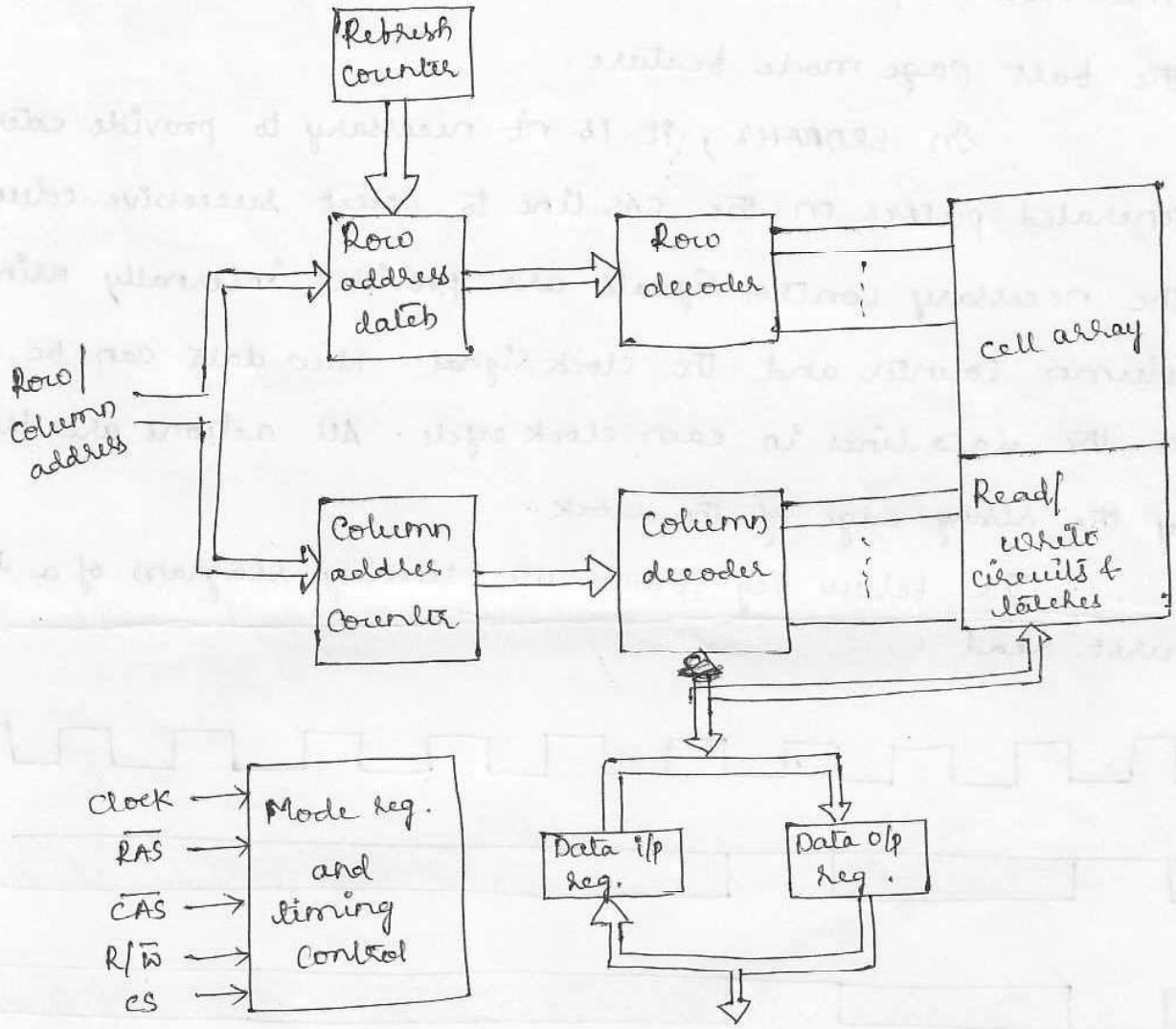
The most useful arrangement is to transfer bytes in sequential order, which is applied by applying a consecutive sequence of column address under the control of successive CAS signals.

This scheme allows transferring a block of data at a much faster rate than can be achieved for transfers involving random addresses. The block transfer capability is referred to as 'fast page mode' features.

Synchronous DRAM's :-

The operation is directly synchronized with a clock signal, such memories are known as synchronous DRAM's.

The below figure shows the structure SDRAM.



The address & data connections are buffered by means of registers. The o/p of each sense amplifier is connected to a latch. A read operation causes the contents of all cells in the selected row to be loaded into these latches. If an access is made for refreshing purposes but it will not change the contents of these latches and it will refresh the contents of the cells. Data held in the latches that correspond to the selected columns are transferred into the data o/p reg.

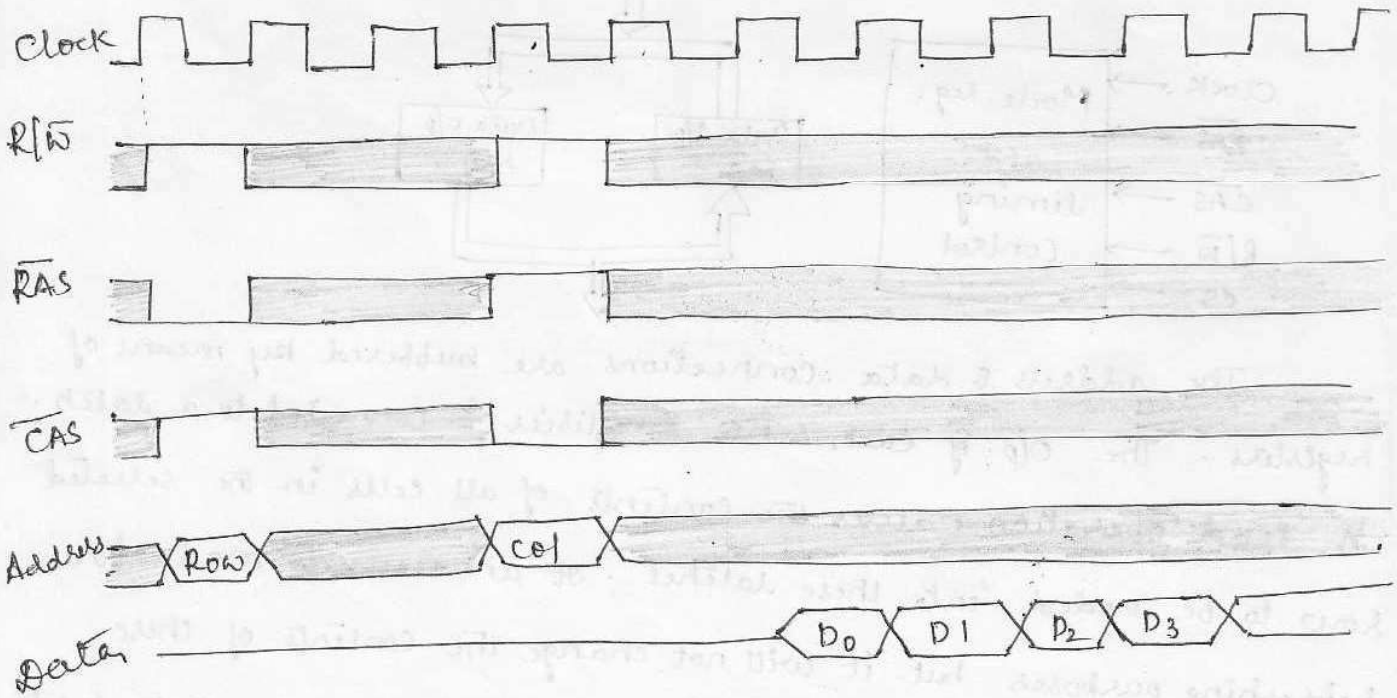
www.jntuworldupdates.org

SRAM's have several different modes of operation, which can be selected by writing control information into 'mode' register.

For eg, burst operations of different lengths can be specified. The burst operations use the block transfer capability as in the fast page mode feature.

In SRDRAM's, it is n't necessary to provide externally generated pulses on the CAS line to select successive columns. The necessary control signals are provided internally using a column counter and the clock signal. New data can be placed on the data lines in each clock cycle. All actions are triggered by the rising edge of the clock.

The below fig shows the timing diagram of a typical burst read



First, the row address is latched under control of the RAS signal. The memory typically takes 2 or 3 clock cycles to activate the selected row. Then, the column address is

www.jntuworldupdates.org

latched under control of $\overline{CAS}$ signal. After a delay of one clock cycle, the first set of data bits is placed on the data lines.

The SDRAM automatically increments the column address to access the next three sets of bits in the selected row, which are placed on the data lines in the next 3 clock cycles.

SDRAMs have built-in refresh circuitry. A part of this circuitary is a refresh counter, which provides the addresses of the rows that are selected for refreshing.

The commercial SDRAM's can be used with clock speeds above 100 MHz.

~~Asynchronous DRAM's~~ :-

Internal organization of Memory chips :-

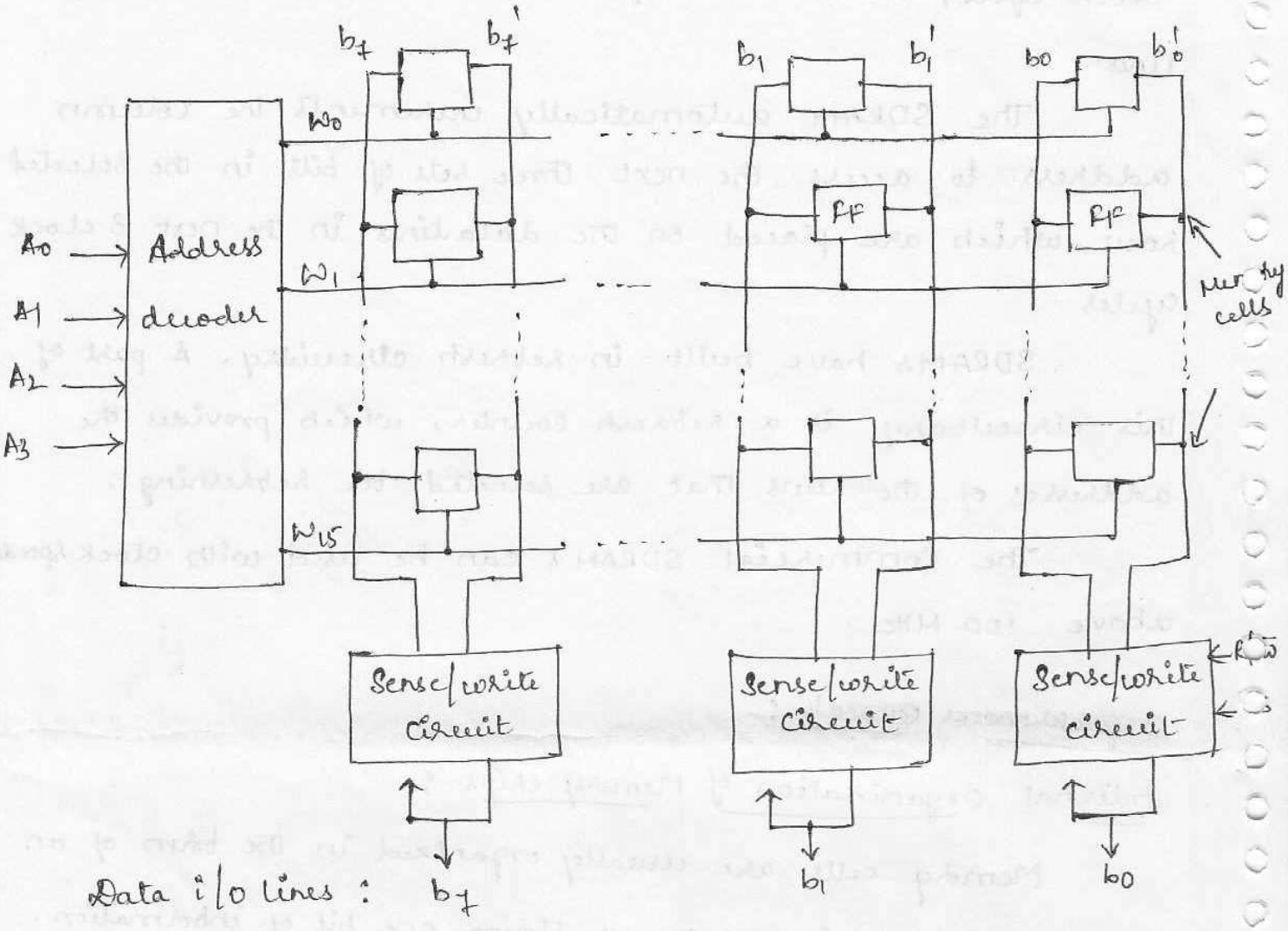
Memory cells are usually organized in the form of an array, each cell is capable of storing one bit of information.

The fig. is an example of a very small memory chip consisting of 16 words of 8 bits each.

Each row of cells constitutes a memory word and all cells of a row are connected to a common line is referred to as the word line, which is driven by the address decoder on the chip. The cells in each column are connected to a sense/write circuit by two bit lines.

The sense/write circuits are connected to the data i/o lines of the chip. During read operation, these circuits sense, or read the information stored in the cells selected by a

www.jntuworldupdates.org



word line and transmit this information to the o/p data bus. During a write operation, the sense/write circuit receive the information and store the cells of the selected ~~row~~ word.

The data i/p & the data o/p of each sense/write circuit are connected to a single bidirectional data line that can be connected to the data bus of a computer.

Two control lines, R/$\bar{W}$ & CS are provided in addition to the address & data lines. The R/$\bar{W}$ input specifies the required operation & the CS (chip select) i/p selects a given chip in a multi chip memory system.

www.jntuworldupdates.org

PROM :-

For small quantities, we are going to use a type of a ROM called programmable Read only Memory.

In this, the programmability is achieved by inserting a fuse at point p (shown in fig.). Before it is programmed, the memory contains all 0's.

The user can insert 1's at the required locations by burning out the fuses at these locations using high-current pulses.

PROM's provide flexibility & convenience

PROM's provide faster and considerably less expensive because they can be programmed directly by the user. The ROM & PROM are irreversible.

EPROM :-

This an type of ROM, which allows the stored data to be erased and new data to be loaded. Such an erasable, reprog-rammable ROM is usually called an EPROM.

EPROM's are capable of retaining stored information for a long time, they can be used inplace of ROM's.

The EPROM & ROM has similar structure. In EPROM cell, the connection to ground is always made at point 'p' and a special transistor is used, which has the ability to function as either as a normal transistor or as a disabled transistor that is always turned off. i.e., the transistor can be programmed to behave as a permanently open switch.

The advantage is that their contents can be erased and reprogrammed. The disadv. is that a chip is removed physically from the circuit for reprogramming and the entire contents is erased by exposure to U.V. light.

www.jntuworldupdates.org

EEPROM :-

The another version of erasable PROM's that can be both programmed and erased electrically. Such chips are called EEPROM's. It is possible to erase the cell contents selectively.

The disadv. is that different voltages are needed for erasing, writing and reading the stored data.

Flash Memory :-

Flash Memory is intermediate b/w EPROM & EEPROM. Like EEPROM, flash memory uses an electrical erasing Technology. An entire flash memory can be erased in one or a few sector's which is much faster than EPROM.

Flash Memory uses only one transistor per bit and so achieves the high density.

Cache memories :-

The speed of the mainmemory is very low, because for good performance, the processor cannot spend much of its time waiting to access instructions and data in main memory.

An efficient solution is to use a fast cache memory which essentially makes the main memory appear to the processor to be faster.

The effectiveness of cache mechanism is based on a property of computer programs called 'locality of reference'. Analysis of programs show that most of their execution time is spent on routines in which many instructions are executed repeatedly.

The instructions in localized areas of the program are

executed repeatedly during some time period and the remainder of the program is accessed relatively infrequently. This is referred to as 'locality of reference'. This is divided into two types 1) Temporal. 2) Spatial.

The temporal means that a recently executed instruction is likely to be executed again.

The spatial means that instructions in close proximity to a recently executed instruction (Based on instruction address)

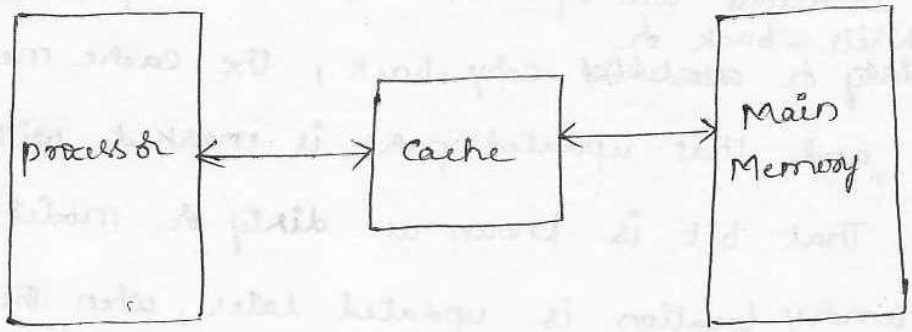
If the active segments of a program can be placed in a fast cache memory, then the total execution time is reduced.

The memory control circuitary is designed to take adv. of the property of locality of reference. The temporal aspect of the locality of reference, that whenever an information item is first needed, this item should be brought into cache where it is needed.

The spatial aspect suggests that instead of fetching one item from the main memory to the cache, it is useful to fetch several items that reside at adjacent addresses.

The term 'block' to refer to a set of contiguous address locations of some size. Another term is refer to a cache block is 'cache line'.

The below fig. shows the use of a cache memory



www.jntuworldupdates.org

When a Read Request is received from the processor, the contents of a block of memory words containing the location specified are transferred into the cache one word at a time. When the program references any of the locations in this block, the desired contents are read directly from the cache. The cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory.

The correspondence b/w the main memory blocks and those in the cache is specified by a mapping function.

When the cache is full and a memory word that is not in the cache is referenced, the cache control h/w must decide which block should be removed to create space for the new block that contains the referenced word. The collection of rules for making this decision is "replacement algorithm".

The cache control circuitary determines whether the requested word currently exists in the cache. The Read or write operation is performed on the appropriate cache location. In Read operation, the main memory is not involved.

The write operation may be proceed into two ways

1) Write-through protocol 2) ~~dirty or modified bit~~ Write-back or copy-back

In write-through protocol, both cache memory and the main memory locations are updated simultaneously.

In ~~dirty or modified~~ Write-back or copy-back, the cache memory is only updated and that updated part is marked with its associated flag bit. That bit is known as dirty or modified bit. The main memory location is updated later, when the block containing

www.jntuworldupdates.org

this marked word is to be removed from the cache to make room for a new block.

When the addressed word in a Read operation is not in the cache, a "read miss" occurs. The block of words that contains the requested word copies from the main memory into the cache. After the entire block is loaded into the cache, the particular word requested is forwarded to the processor.

The latter approach, which is called load-through or early restart. In this, it reduces the processor's waiting period, but it is more complex circuitary.

During a write operation, if the addressed word is not in the cache, a "write miss" occurs. Then, the write-through is used, the information is written directly into the main memory.

In the case of write-back protocol, the block containing the addressed word is first brought into the cache, and then the desired word in the cache is overwritten with the new information.

Mapping functions :-

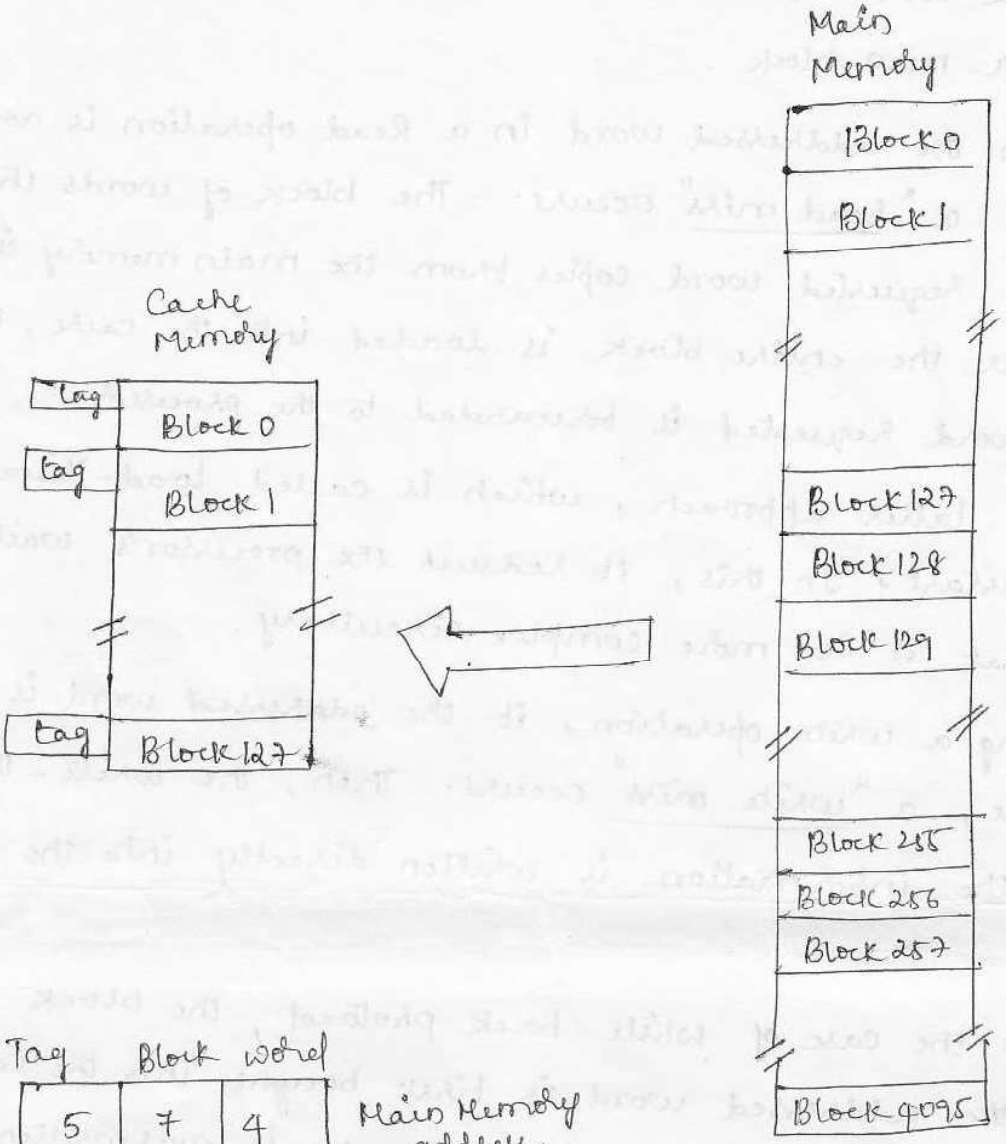
Direct Mapping :-

The simplest way to determine cache locations in which to store memory blocks is the direct-mapping technique.

In this technique, block j of the main memory maps onto block j modulo 128 of the cache, as depicted in fig.

One of the main memory blocks 0, 128, 256 ... are loaded into ~~main memory~~ cache, it is stored in cache block 0. 1, 129, 257 -- are stored in cache block 1 and so on.

www.jntuworldupdates.org



Tag	Block	word
5	7	4

Main Memory address.

Since more than one memory block is mapped onto a given cache block position, contention may arise for that position even when the cache is not full.

For eg, instructions program may start in block 1 and continue in block 129, after a branch. As this program is executed, both of these blocks are transferred to the block-1 position in the cache. Contention is resolved by allowing the new block to overwrite the currently resident block.

Placement of a block in the cache is determined from the memory address. The memory address can be divided into three fields as shown in fig.

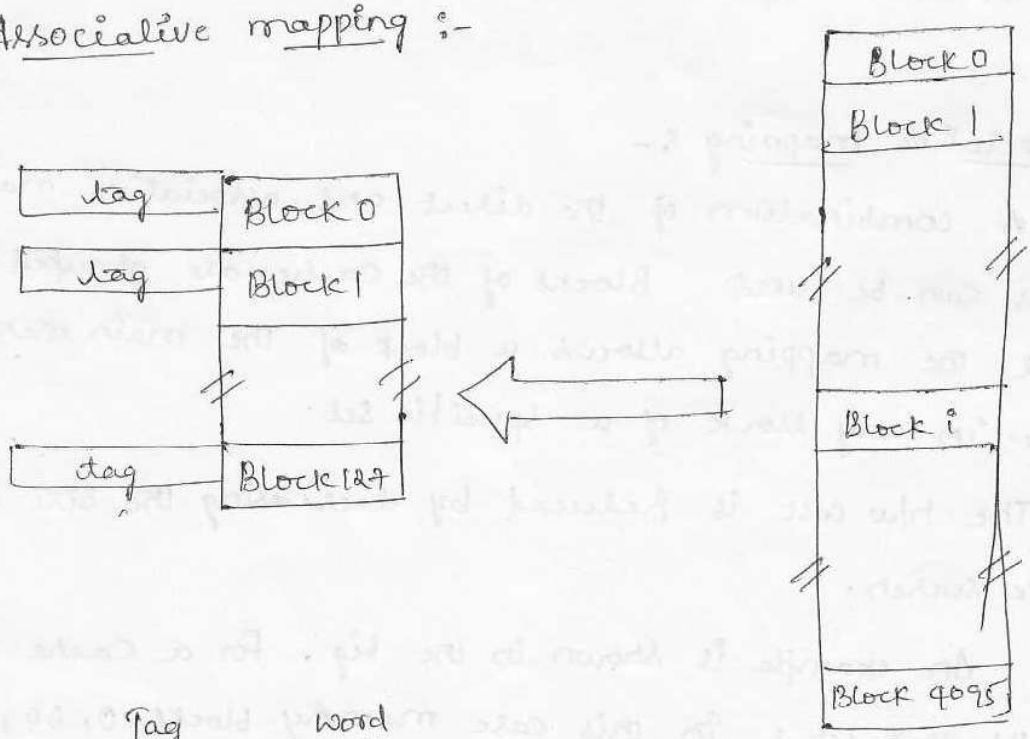
www.jntuworldupdates.org

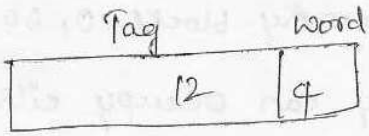
The low-order 4 bits select one of 16 words in a block. When a new block enters the cache, the 7-bit cache block field determines the cache position in which this block must be stored. The high-order 5 tag bits associated with its location in the cache. They identify the 32 blocks that are mapped into cache.

As execution proceeds, the 7-bit cache block field of each address generated by the processor points to a particular block location in the cache. The high-order 5 bits of the address are compared with the tag bits associated with the cache location. If they match, then the desired word is in that block of cache. If there is no match, then the block containing the required word must first be read from the main memory and loaded into the cache.

The direct-mapping is easy to implement but it is not very flexible.

Associative mapping :-





Main Memory address

www.jntuworldupdates.org

This is a much more flexible mapping method, in which a main memory block can be placed into any cache block position

In this 12 tag bits are required to identify a memory block when it is resident in the cache. The tag bits of an address received from the processor are compared to the tag bits of each block of the cache, if the desired block is present. This is called the 'associative-mapping' technique.

A new block that has to be brought into the cache has to replace an existing block only if the cache is full. In this case we need an algorithm to select that particular block to be replaced.

The cost of an associative cache is higher than the cost of a direct-mapped cache because of the need to search all 128 tag patterns to determine whether a given block is in the cache.

A search of this kind is called an associative search.

Set-Associative mapping :-

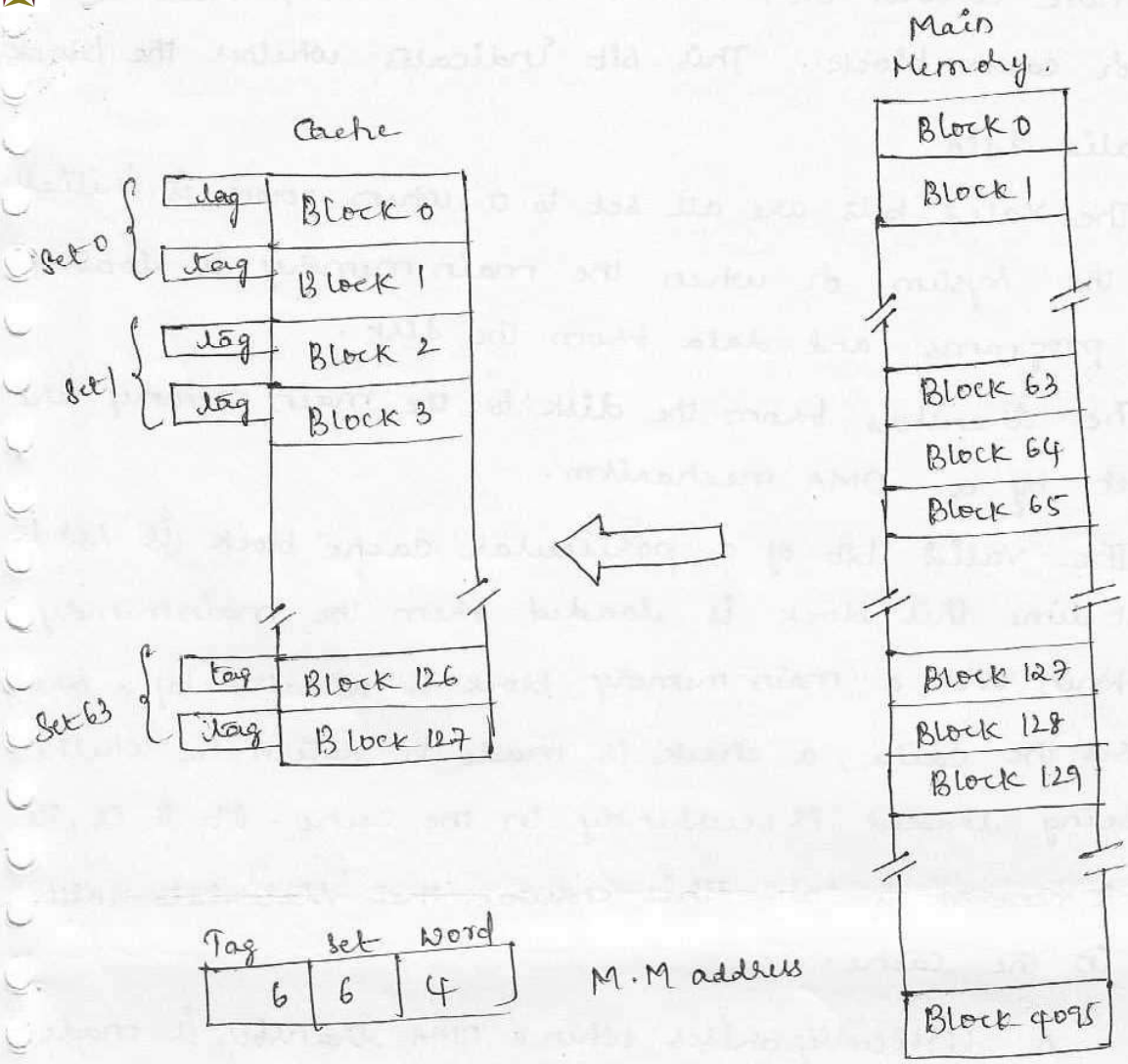
A combination of the direct and associative mapping techniques can be used. Blocks of the cache are grouped into sets and the mapping allows a block of the main memory to reside in any block of a specific set.

The h/w cost is reduced by decreasing the size of the associative search.

An example is shown in the fig. For a cache with two blocks per set. In this case memory blocks 0, 64, 128 ... 4032 map into cache set 0 and they can occupy either of the two block positions within this set.

www.specworld.in
www.smartzworld.com

www.jntuworldupdates.org



Tag	set	Word
6	6	4

M.M address

Having 64 sets means that the 6-bit set field of the address determines which set of the cache might contain the desired block. The tag field of the address must then be associatively compared to the tags of the two block of the set to check if the desired block is present.

For the main memory and cache sizes, four blocks per set can be accommodated by a 5-bit set field, eight blocks per set can be accommodated by a 4-bit set field and so on. The extreme condition of 128 blocks per set requires no set bits & corresponds to the fully associative technique, with 12 tag bits. The other extreme of one block per set is the direct mapping method. A cache that has K blocks per set is referred to as a K-way set-associative cache.

One more control bit, called the valid bit, must be provided for each block. This bit indicates whether the block contains valid data.

The valid bits are all set to 0 when power is initially applied to the system or when the main memory is loaded with new programs and data from the disk.

The transfers from the disk to the main memory are carried out by a DMA mechanism.

The valid bit of a particular cache block is set to '1' the first time this block is loaded from the main memory.

When ever a main memory block is updated by a source that bypasses the cache, a check is made to determine whether the block being loaded is currently in the cache. If it is, its valid bit is cleared to '0'. This ensures that 'stale' data will not exist in the cache.

A difficulty arises when a DMA transfer is made and the cache uses the write-back protocol.

One solution to this problem is to flush the cache by forcing the dirty data to be written back to the memory before the DMA transfer takes place.

The need to ensure that two different entities (processor & DMA) uses the same copies of data is referred to as cache-coherence problem.

<u>Replacement algorithms</u> :-

When a block is to be overwritten, it is sensible to overwrite the one that has gone the longest time without being referenced. This block is called the last recently used (LRU) block and the technique is called the LRU Replacement algorithm.

www.jntuworldupdates.org

LRU Algorithm :-

To use the LRU algorithm, the cache controller must track references to all blocks. To track the LRU block of a four-block set in a set associative cache. A 2-bit counter can be used for each block. When a 'hit' occurs, the counter of the block that is referenced is set to 0.

When a 'miss' occurs and the set is not full, the counter associated with the new block loaded from the main memory is set to 0 and the values of all other counters are increased by one.

When a 'miss' occurs and the set is full, the block with the counter value 3 is removed, the new block is put in its place and its counter is set to '0'. The other three block counters are incremented by one.

The LRU algorithm has been used extensively. Although it performs well for many patterns, it can lead to poor performance in some cases. For eg, LRU produces bad results, when accesses are made to sequential elements of an array that is too large to fit into the cache. performance of the LRU algorithm can be improved by introducing a small amount of randomness in deciding which block to replace.

www.jntuworldupdates.org

Performance Considerations :-

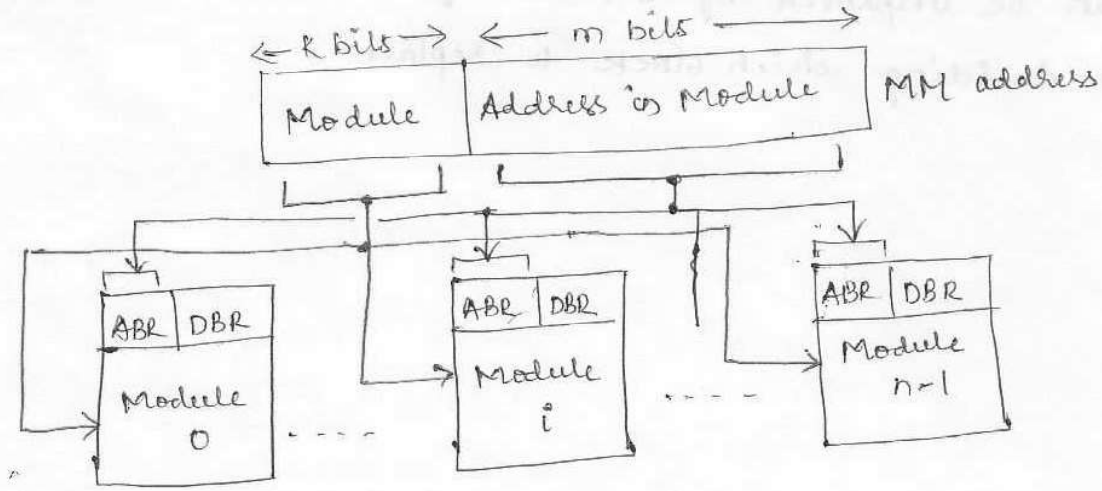
Two key factors of a computer are performance & cost. performance depends on how fast machine instructions can be brought into the processor for execution & how fast they can be executed.

An effective way to introduce parallelism is to use an interleaved organization.

Interleaving :-

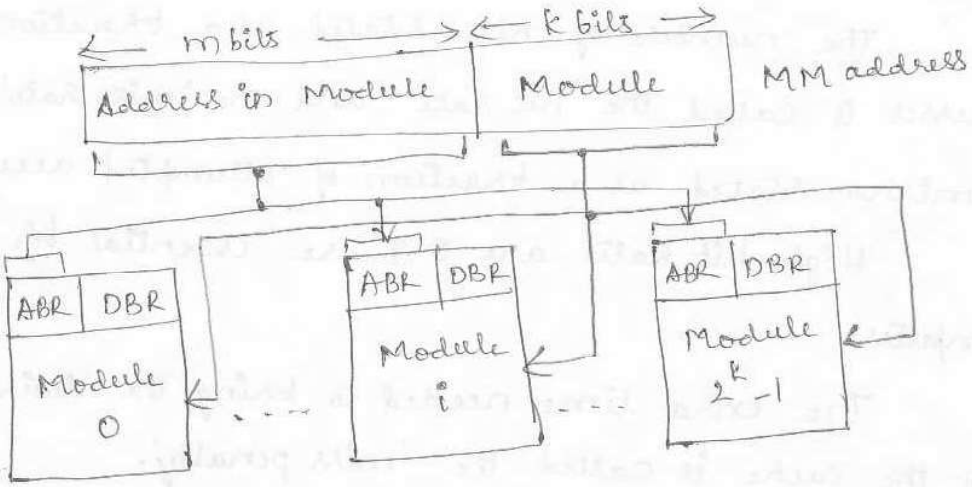
If the main memory of a computer is structured as collection of physically separate modules, each with its own address buffer register (ABR) and data buffer register (DBR). The memory access operations may proceed in more than one module at the same time. Thus, the aggregate rate of transmission of words to and from the main memory system can be increased.

Two methods of address layout are shown below.



a) Consecutive words in a module.

www.jntuworldupdates.org



b) Consecutive words in consecutive Modules

In fig (a), the memory address generated by the processor is decoded. The high-order K bits name one of 'n' modules and the low-order 'm' bits name a particular word in that module. When consecutive locations are accessed, when a block of data is transferred to a cache, only one module is involved. At the same time, however, devices with DMA ability may be accessing information in other memory Modules.

The fig (b) is a more effective way to address the modules is called Memory interleaving. The low-order K bits of the memory address select a module, and the high-order m bits name a location with in that module. The consecutive addresses are located in successive modules.

Any component of the system that generates requests for access to consecutive memory locations can keep several modules busy at any one time.

This results in both faster access to a block of data & higher avg. utilization of the memory system as a whole. To implement the interleaved structure, there must be 2^k modules otherwise, there will be gaps of nonexistent locations in the memory address space.

www.jntuworldupdates.org

Hit rate & Miss penalty :-

The number of hits stated as a fraction of all attempted accesses is called the 'hit rate' and the 'miss rate' is the number of misses stated as a fraction of attempted accesses.

High hit rates are 0.9 are essential for high-performance computers.

The extra time needed to bring the desired information into the cache is called the 'miss penalty'.

In general, the miss penalty is the time needed to bring a block of data from a slower unit in the memory hierarchy to a faster unit.

The miss penalty is reduced if efficient mechanism for transferring data b/w the various units of the hierarchy are implemented.

Virtual memories :-

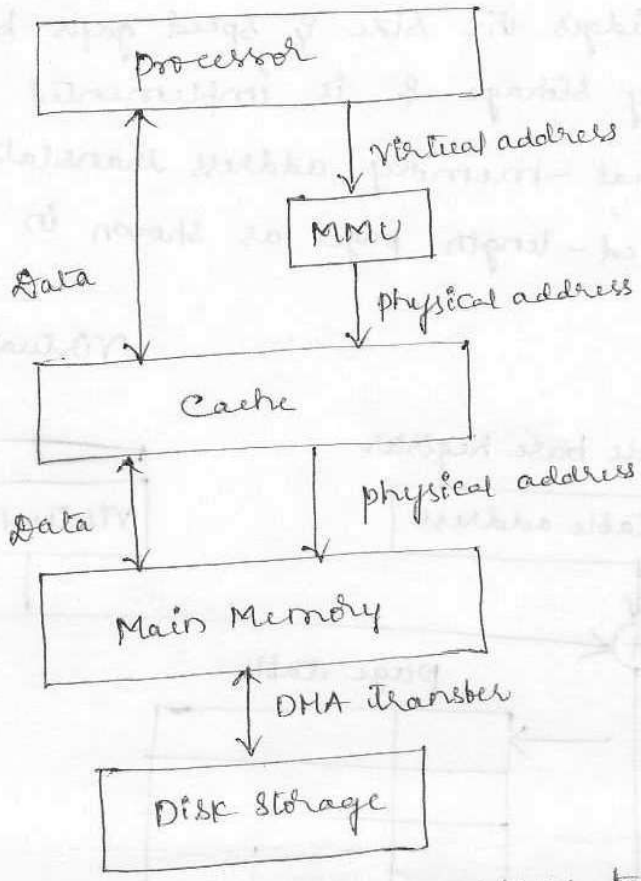
Techniques that automatically move program and data blocks into the physical main memory, when they are required for execution is called virtual techniques.

The binary addresses that the processor issues for either instructions or data are called virtual or logical addresses. These addresses are translated into physical addresses by a combination of h/w & s/w components.

If a virtual address refers to a part of the program or data space that is currently in the physical memory, then the contents of the appropriate location in the main memory are accessed immediately. If the referenced address is not in the main memory, its contents must be brought into a suitable location in the memory.

www.jntuworldupdates.org

The below fig. shows a organization that implements virtual memory. A special h/w unit called the memory Management



unit (MMU), translates virtual address to physical address. When the desired data (or instructions) are in the main memory the data is fetched and transfers to cache memory.

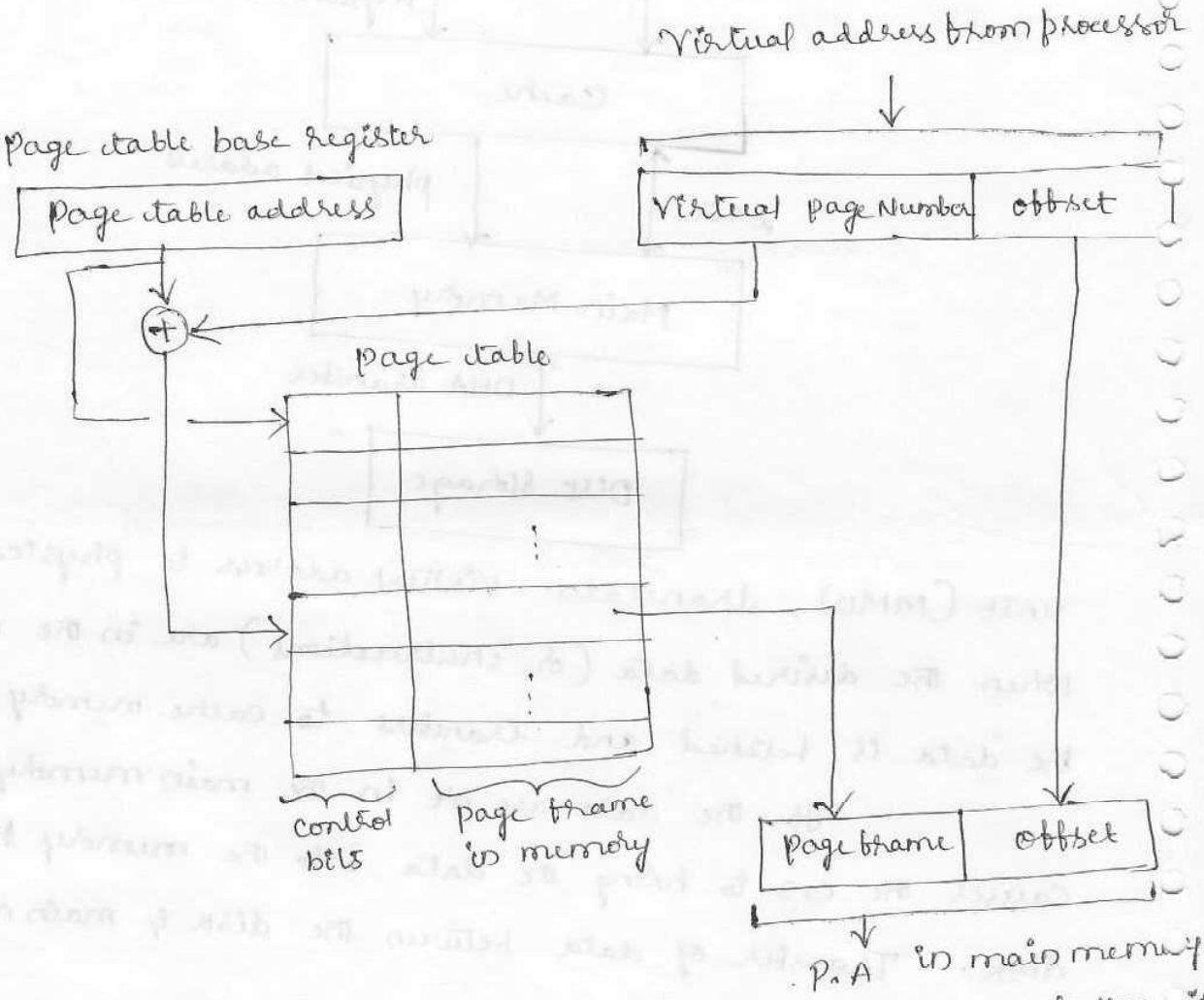
If the data are n't in the main memory, the MMU causes the O.S to bring the data into the memory from the disk. Transfer of data between the disk & main memory is performed using DMA.

Address Translation :-

A simple method for translating virtual addresses into physical addresses is to assume that all programs and data are composed of fixed-length units called pages, each of which consists of a block of words that occupy contiguous locations in the main memory.

The cache memory bridges the speed gap b/w the processor and the main memory and is implemented in h/w. The virtual-memory mechanism bridges the size & speed gaps b/w the main memory and secondary storage & is implemented in s/w.

A virtual-memory address translation is based on the concept of fixed-length pages as shown in fig.



Each virtual address generated by the processor, whether it is for an instruction fetch or an operand fetch / store operation is interpreted as a virtual page Number (High-order bits) followed by an offset (low-order bits) that specifies the location of a particular byte (or word) with in a page.

~~This information that precedes~~ An area in the main memory that can hold one page is called a 'page frame'.

The starting address of the page table is kept in a 'page table base reg'.

www.jntuworldupdates.org

By adding the virtual page number to the contents of this register, the address of the corresponding entry in the page table is obtained.

Each entry in the page table also includes some control bits that describe the status of the page while it is in the main memory.

One bit indicates the validity of the page, whether the page is loaded in main memory.

Another bit indicates whether the page has been modified during resident in the memory.

Other control bits indicates various restrictions that may be imposed on accessing the page.

The page table information is used by the MMU for every read & write access, so the page table should be situated with in the MMU.

The portion consists of the page table entries that correspond to the most recently accessed pages. A small cache, usually called the Translation Look aside Buffer (TLB).

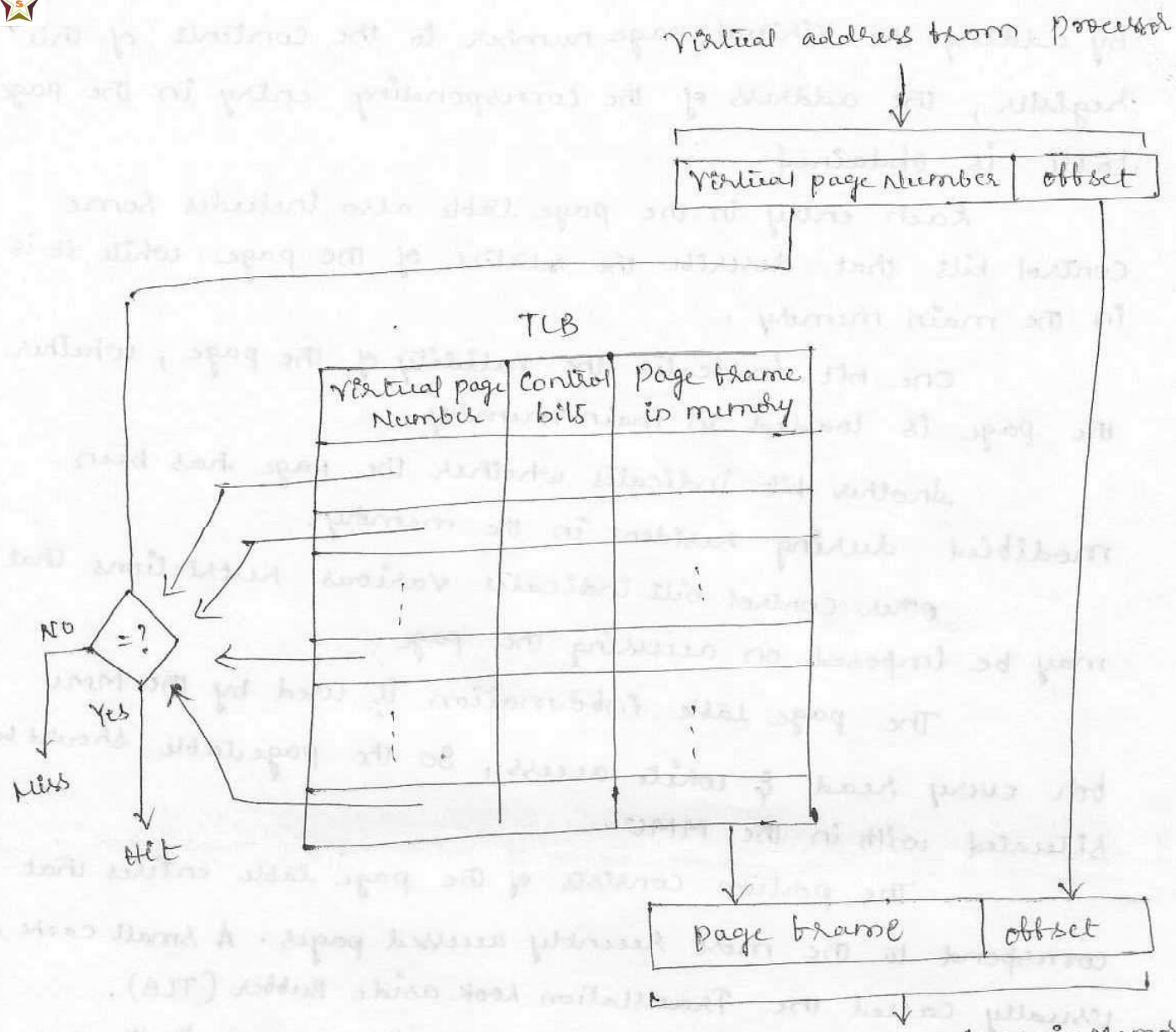
The operation of the TLB with respect to the page table in the main memory is same as the cache memory. The information that constitutes a page table entry, the TLB must also include the virtual address of the entry.

The fig. shows the organization of a TLB where the associative mapping technique is used.

The contents of the TLB be coherent with the content of page tables in the memory. When the O.S changes the contents of page tables, it must simultaneously invalidate the corresponding entries in the TLB.

One of the control bits in the TLB is provided for this purpose. When an entry is invalidated, the TLB will

www.jntuworldupdates.org



acquire the new information as a part of the MMU's normal response to access misses.

Address translation may follows in this way

1) Given a virtual address, the MMU looks for this referenced page in the TLB.

2) If the page table entry for this page is found in the TLB, the physical address is obtained immediately.

3) If there is a miss in the TLB, then the required entry is obtained from the page table in the main memory and the TLB is updated

www.jntuworldupdates.org

When a program generates an access request to a page, if that page is n't in the main memory, then a 'page fault' is occured.

A page fault can be detected, by the MMU asks the O.S. to intervene by raising an exception (interrupt). processing of the active task is interrupted, the control is transferred to the O.S.

The O.S. then copies the requested page from the disk into the main memory and returns control to the interrupted task. A long delay occurs while the page transfer takes place.

If a new page is brought from the disk when the main memory is full, it must replace one of the resident pages.

The problem of choosing which page to remove is critical, because main memory has larger memory. We should large portions in main memory. This will reduce the frequency of transfers to & from the disk.

Here also, the LRU replacement algorithm is used. and the control bits in the page table entries can indicate usage.

One simple scheme is, the control bit is set to '1' whenever the corresponding pages is referenced. The O.S, clears this bit in all page-table entries, thus providing a simple way of determining which have not been used recently.

A modified page has to be written back to the disk before it is removed from the main memory.

The address translation process in the MMU requires some time to perform, mostly dependent on the time needed to look up entries in the TLB.

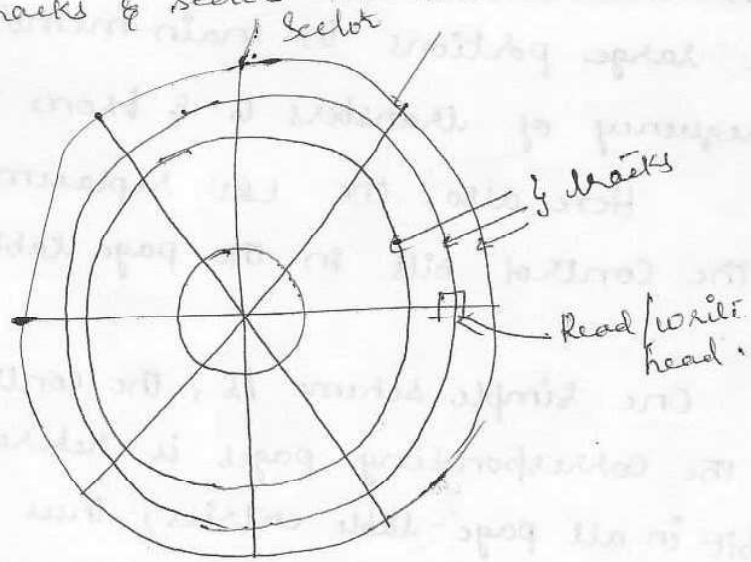
www.jntuworldupdates.org

We can reduce the average translation time by including one or more special registers that retain the virtual page number and the physical page frame of the most recently performed translations. The information in these registers can be accessed more quickly than the TLB.

Magnetic Disks :-

- Magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material. The both sides of the disk are used & several disks may be stacked on one spindle with read/write heads available on each surface.

All disks rotate together at high speed and are n't stopped. Bits are stored in the magnetized surface in spots along concentric circles called tracks. The tracks are commonly divided into sections called sectors. The min. quantity of information which can be transferred is a sector. The subdivision of one disk surface into tracks & sectors are shown in below fig



Some units uses a single read/write head for each disk surface. The track address bits are used by mechanical assembly to move the head into the specified track position before read/write. In other disk systems, separate read/write heads are provided for each track in each surface. The address bits can then select a particular track electronically thro' a decoder circuit.

www.jntuworldupdates.org

This type of unit is more expensive & is found only in very large computer systems.

permanent timing tracks are used in disks to synchronize the bits & recognize the sectors. A disk system is addressed by address bits that specify the disk number, disk surface, the sector number and track with in the sector.

Information transfer is very fast. Disks may have multiple heads & simultanious transfer of bits from several tracks at the same time.

Disks that are permanently attached to unit & can't be removed by the user are called hard disks. A disk drive with removable disks is called a floppy disk.

Magnetic tape :-

A magnetic tape transport consists of the electrical, mechanical and electronic components to provide the parts & control mechanism for a magnetic tape unit.

The tape itself is a strip of plastic coated with a magnetic recording. Bits are recorded as magnetic spot on the tape along several tracks. Seven or nine bits are recorded simultaneously to form a character together with a parity bit. Read/write heads are mounted one in each track so that data can be recorded & read as a sequence of characters.

Magnetic tape units can be ~~started to~~ stopped, started to move forward or in reverse. The information is recorded in blocks referred to as records. Gaps of unrecorded tape are inserted b/w records where the tape can be stopped. The tape starts moving while in a gap and attains its constant speed.

www.jntuworldupdates.org

By reading the bit pattern at the end of the record, the control recognizes the beginning of a gap.

A tape unit is addressed by specifying the record numbers & no. of records. Records may be fixed or variable length.

RAID :-

RAID (Redundant Array of Inexpensive Disks) is a Technology where multiple independent disks are maintained. In this, a single large file is partitioned & each partition is placed in each of these independent disks. When ever any process requires these partitions, it can easily access these disks parallely & acquire the required data.

When these disks are accessing parallely, hence the given accessing time is decreased where as bandwidth of data transfer is increased.

The data belonging to a single large file, when gets partitioned then, each partition is referred as 'stripe' used the way each of these strips are placed in these disks is referred as stripping.

Hence RAID is dependent on two factors.

1) Stripping of data in order to access all the disks parallely &
2) Data redundancy to increase reliability.

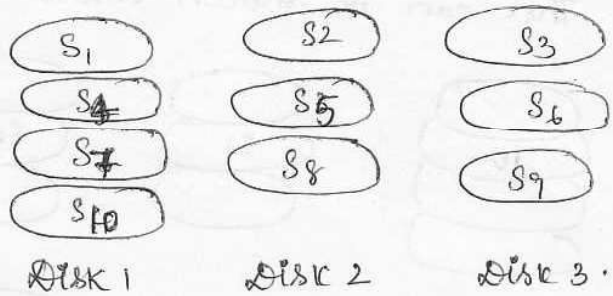
RAID has different levels which are explained below.

RAID Level 0 :-

In this level, stripes of data are belonging to the files is introduced in each disks, such that first strip is added in first disk, second strip is added in disk2 and so on. As the no. of disks ends, the same procedure is repeated by considering disk1 as initiating disk.

The stripping procedure ends only when all strips are filled in these disks. In Raid level 0, there is no provision for data redundancy.

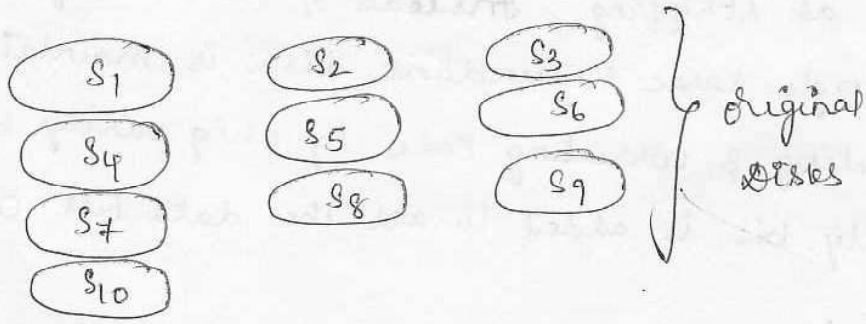
Various consequences occuring at RAID level 0 can be represented below:

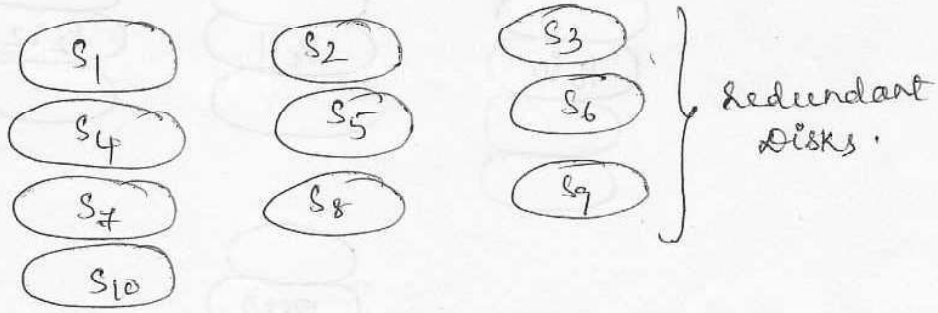


RAID level 1:-

At this level data redundancy is considered to be of primary focus. RAID level 0 is applied here, inorder to achieve redundancy each physical disk is maintained twice. As a result each stripe of data will have it copy in a separate identical disk. Hence reliability can be achieved.

This can be shown below.



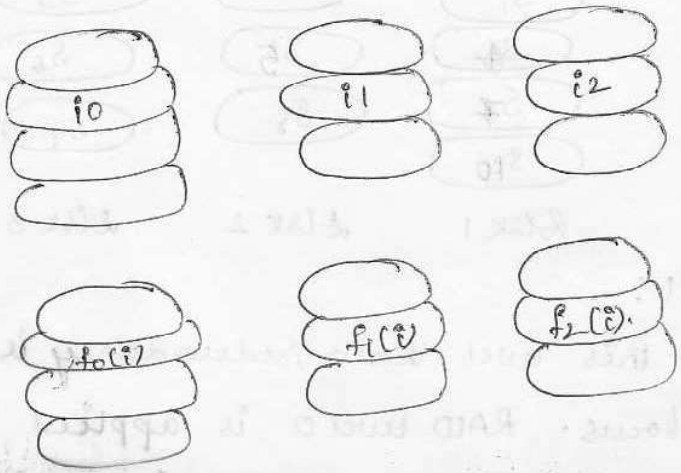


www.jntuworldupdates.org

RAID level 2:

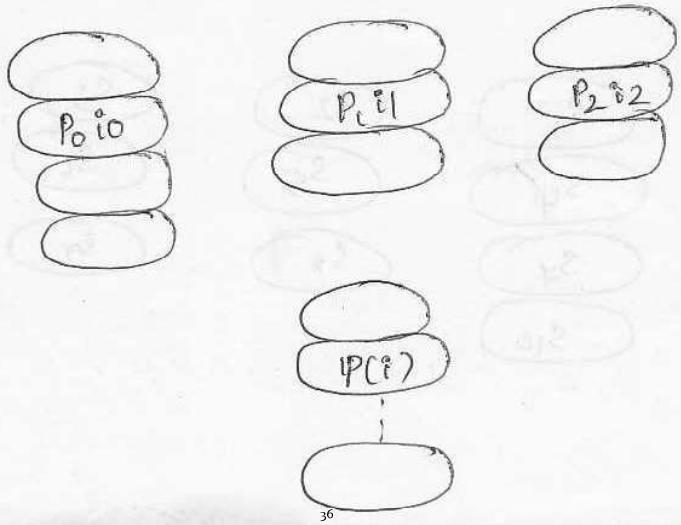
RAID level2 is a combination of level 0 & level 1, Instead of looking the frequently occuring errors, an error detecting as well as correcting code is maintained across all the disks. The frequently used error detecting code is "Hamming code."

This can be shown below



RAID level 3:

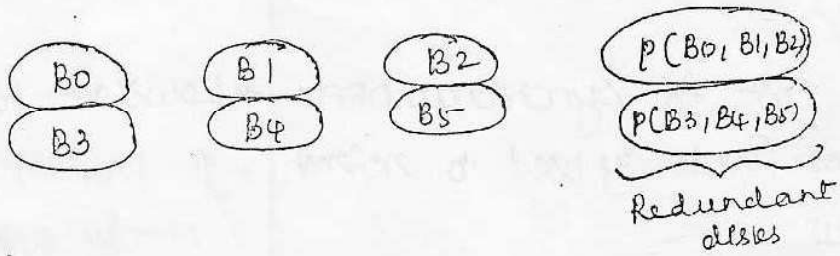
RAID level3 reflects other levels interms of redundancy as well as stripping. Instead of maintaing equal no. of redundant a single large redundant disk is maintained. Also, the error detecting & correcting code by using parity bit concept (extra bit) parity bit is added to all the data bits corresponding to each disk.



www.jntuworldupdates.org

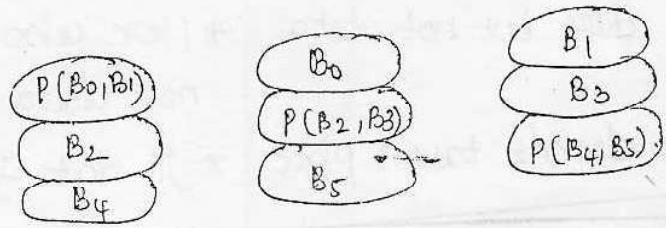
RAID level 4 :

Most of the data strips are combined to form a t of data. A single redundant disk is maintained. For ce combination of blocks, a parity bit is added and the code placed in the redundant disk. In this level, the redunda disk with parity added data blocks turn out to be a dis



RAID level 5 :

In order to overcome the limitation of level 4 i.e., instead of placing all the parity enabled blocks in one disk, the are distributed to all the disk. Hence all the disks now responsible for maintaining parity enabled blocks.



SRAM	DRAM
1) Complex CKt	1) Simple CKt
2) Costlier	2) Cost effective
3) Use latch & translational ff's	3) Capacitors used
4) Canbe used as Cache	4) Can be used as MM.
5) ~~Temp~~ storage permanent	5) Temp storage
6) Current flow when it is accessed only	6) Charging Capacitors.
7) Hasn't latch	7) Hasn't latch
8) less pwr consumption	8) High pwr Consumption.
9) we store automatically in latch	9) Store data in auxilary memory after discharging

www.jntuworldupdates.org

Synchronous DRAM	Asynchronous DRAM
* In this it is directly synchronize with clock signal	* In this it is controlled asynchronously by using timing signal
* The complexity of ckt is not happened	* The ckt complexity is here
* High cost for synchronous DRAM	* Low cost for asynchronous DRAM
* Refresh counter is used to refresh the cell	* It provides flexibility in designing memory systems.
* Low density	* High density
* Time delay is present	* Time delay is not here
* No data lines are used	* Data lines D0 to D7 are used
* over write doesn't occur	* over write occurs
* Here add is diff but not data	* Here also the add is diff but not data.
* If over write doesn't takes place	* If data is liar then only the over write of data takes place
* Here data i/p & data o/p regs are present.	* Here data i/p & data o/p regs are not present.
* It can be used with clock above 100MHz	

SRAM	DRAM
10) Manipulated by selection of bits	10) Manipulated by giving power supply.
11) Continous pwr supply is req'd	11) Charging is req'd to capacitors
12) fastest manipulation speed	12) Slower manipulation

www.jntuworldupdates.org

UNIT-4

Input - Output Organization

Peripheral Devices :-

The input-output subsystem of a computer is commonly referred to as I/O which provides an efficient mode of communication between the central system and the outside environment.

Devices that are under the direct control of the computer are said to be connected on-line. These devices are designed to read information into or out of the memory unit upon command from the CPU and are considered to be part of the total computer system.

Input or output devices attached to a pcs computer are also called 'peripherals'. There are three types of peripherals 1) input 2) output 3) Input-output. These peripherals may be analog or digital and serial or parallel.

The common peripherals are keyboards, display units and printers.

peripherals that provide auxiliary storage for the system are magnetic disks & tapes.

Monitor and Keyboard :-

The most commonly used peripheral is 'Video monitors'. They consist of a keyboard as the input device and a display unit as the O/P device.

There are different types of video monitors, the most popular use is Cathode Ray Tube (CRT). The CRT contains

www.jntuworldupdates.org

an electronic gun that sends an electronic beam to a phosphorescent screen in front of the tube. The beam can be deflected horizontally and vertically. To produce a pattern on the screen, a grid inside the CRT receives a variable voltage that causes the beam to hit the screen and make it glow at selected spots.

A characteristic feature of display devices is a cursor that marks the position in the screen where the next character will be inserted. The cursor can be moved to any position in the screen, to a single character, beginning of a word, ending of a word etc.

Edit keys add or delete information based on the cursor position. The display terminal can operate in a single character mode where all characters entered on the screen through the keyboard.

In the block mode, the edited text is first stored in a local memory inside the terminal. The text is transferred to the computer as a block of data.

Printer :-

printers provide a permanent record on paper of computer output data or text.

There are three different types of character printers.
1) Daisy wheel printer 2) Dot matrix 3) laser printer.

The daisy wheel printer contains a wheel with the characters placed along the circumference. To print a character, the wheel rotates to the proper position and a magnet then presses the letter against the ribbon.

www.jntuworldupdates.org

The dot matrix printer contains a set of dots along the printing mechanism. Each dot can be printed or not, is depending on the specific characters that are printed on the line.

The laser printer uses a rotating photographic drum that is used to imprint the character images. The pattern is then transferred onto paper in the same manner as a copying machine.

Magnetic tape :-

Magnetic tapes are used mostly for storing files of data.

Magnetic disk :-

Magnetic disks have high-speed rotational surfaces coated with magnetic material.

Input - output Interface :-

I/O interface provides a method for transferring information between internal storage and external I/O devices.

The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral.

The major differences are

1) peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory which are electronic devices. so, a conversion of signal values may be required.

2) The data transfer of peripherals is usually slower than the

www.jntuworldupdates.org

transfer rate of the CPU and a synchronization mechanism may be needed.

3) Data codes and formats in peripherals differ from the word format in the CPU and memory.

4) The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

To resolve these differences, computer systems include special h/w components between the CPU and peripherals to supervise and synchronize all i/p & o/p transfers. These components are called 'interface' units.

In general, the word "interface" is the point of contact between two parts of a system.

Two main types of interface are 1) CPU interface that corresponds to the system bus.

2) input-output interface that corresponds to the input output device.

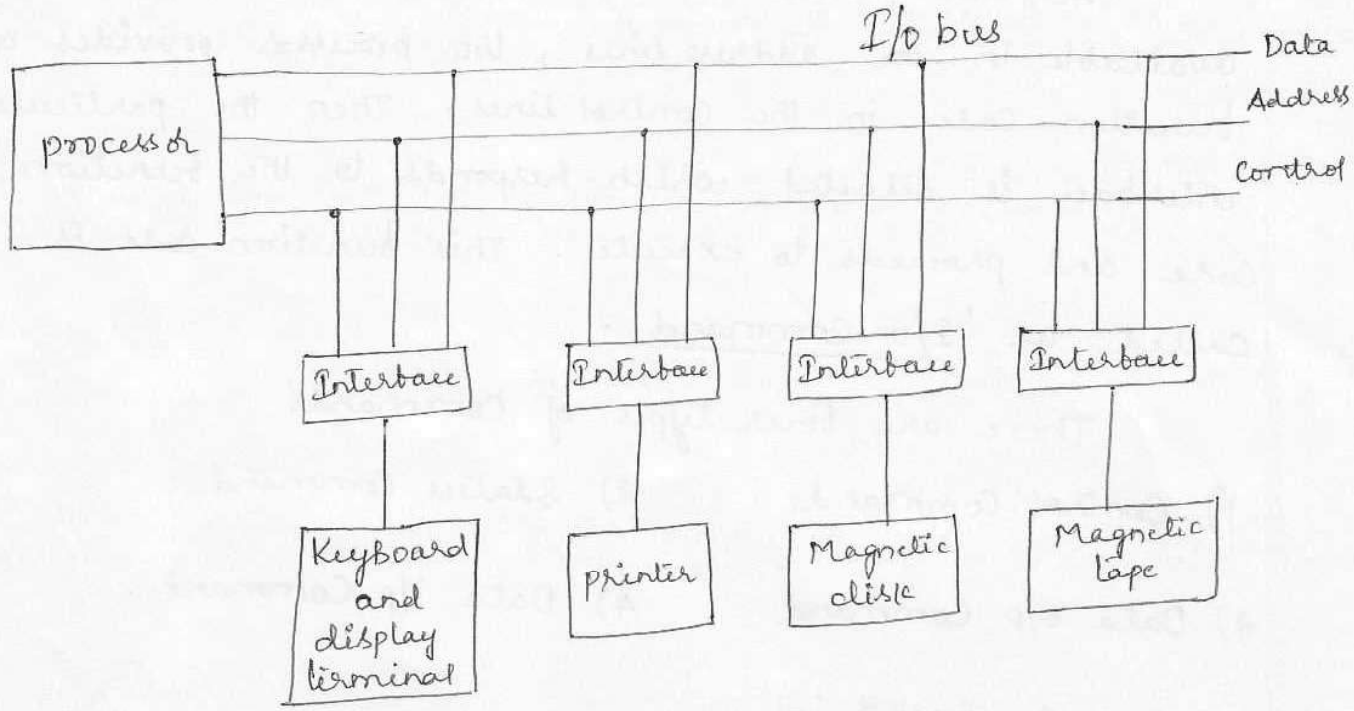
I/O Bus and Interface Modules :-

The communication link between the processor and several peripherals is shown in fig.

The I/O bus consists of data lines, address lines and control lines. The magnetic disk, printer and terminal are employed in any general-purpose computer.

Each peripheral device has associated with it an interface unit. Each interface decodes the address

www.jntuworldupdates.org



and control received from the I/O bus, interprets them for the peripheral and provides signals for the peripheral controller. It also synchronizes the data flow and supervises the transfer between peripheral and processor. Each peripheral has its own controller that operates the particular electro-mechanical device.

The I/O bus from the processor is attached to all peripheral interfaces.

To communicate with a device, the processor places a device address on the address lines. Each interface is attached to the I/O bus contains an address decoder that monitors the address lines. When the interface detects its own address, it activates the path between the bus lines and the device that it controls, remaining others are disabled by their interface.

www.jntuworldupdates.org

At that particular instance, the address is made available in the address lines, the processor provides a function code in the control lines, Then the particular interface is selected which responds to the function code and proceeds to execute. This function code is called as '<u>I/O Command</u>'.

There are four types of commands

1) Control Command 2) Status Command

3) Data o/p Command 4) Data i/p Command.

<u>Control Command</u> :-

It is issued to activate the peripheral and to inform it what to do.

<u>Status Command</u> :-

It is used to test various status conditions in the interface and the peripheral.

<u>Data o/p Command</u> :-

It causes the interface to respond by transferring data from the bus into one of its registers.

<u>Data i/p Command</u> :-

The interface which receives an item of data from the peripheral and places it in its buffer register. The processor checks if data are available by means of a status command and then issues a data i/p command. Then the interface places the data on the data lines, where they are accepted by the processor.

www.jntuworldupdates.org

I/O Versus Memory Bus :-

The memory bus contains data lines, address lines and read/write control lines.

There are three ways that computer buses can be used to communicate with memory and I/O.

1) use two separate buses, one for memory and other for I/O.
2) use one common bus for memory and I/O with common control lines.
3) use one common bus for both memory and I/O but have separate control lines.

In method 1, the computer has independent sets of data, address and control buses i.e., one for accessing memory and the other for I/O. This method can be implemented, where we have separate I/O processor in addition to CPU.

The memory communicates with both the CPU and the IOP through a memory bus. The IOP also communicates with the input and the output devices through a separate I/O bus with its own address, data and control lines.

The purpose of the IOP is to provide an independent pathway for transfer of information between external devices and internal devices.

The IOP is sometimes called 'data channel'.

www.jntuworldupdates.org

Isolated Versus Memory - Mapped I/O :-

The I/O read and I/O write control lines are enabled during an I/O transfer. The memory read and memory write control lines are enabled during an memory transfer. This configuration isolates all I/O interface addresses from the addresses assigned to memory and is refered to as the 'isolated I/O' for assigning addresses in a common bus.

In the isolated I/O configuration, the CPU has distinct i/p and o/p instructions and each of these instructions is associated with address of an interface register.

When the CPU fetches and decodes the operation code of an i/p or o/p instruction, it places the address associated with the instruction into the common address lines. At mean time, it enables the I/O read or I/O write control line. This informs the external components that are attached to the common bus that the address in the address lines is for an interface register and not for a memory word.

When the CPU is fetching an instruction or an operand from memory, it places the memory address on the address lines and enables the memory read or memory write control line. This informs the external components that the address is for a memory word and not for an I/O interface register.

www.jntuworldupdates.org

The second alternative, has same address space for both memory and I/O. i.e., it employs only one set of read and write signals and not distinguish between memory and I/O addresses. This configuration is referred to as memory-mapped I/O.

The assigned addresses for interface registers cannot be used for memory words, which reduce the memory address range available.

In this organization, there is no specific i/p or o/p instructions. The CPU can manipulate I/O data residing in interface registers with the same instructions that are used to manipulate memory words. Each interface is organized as a set of registers that respond to read and write requests in the normal address space.

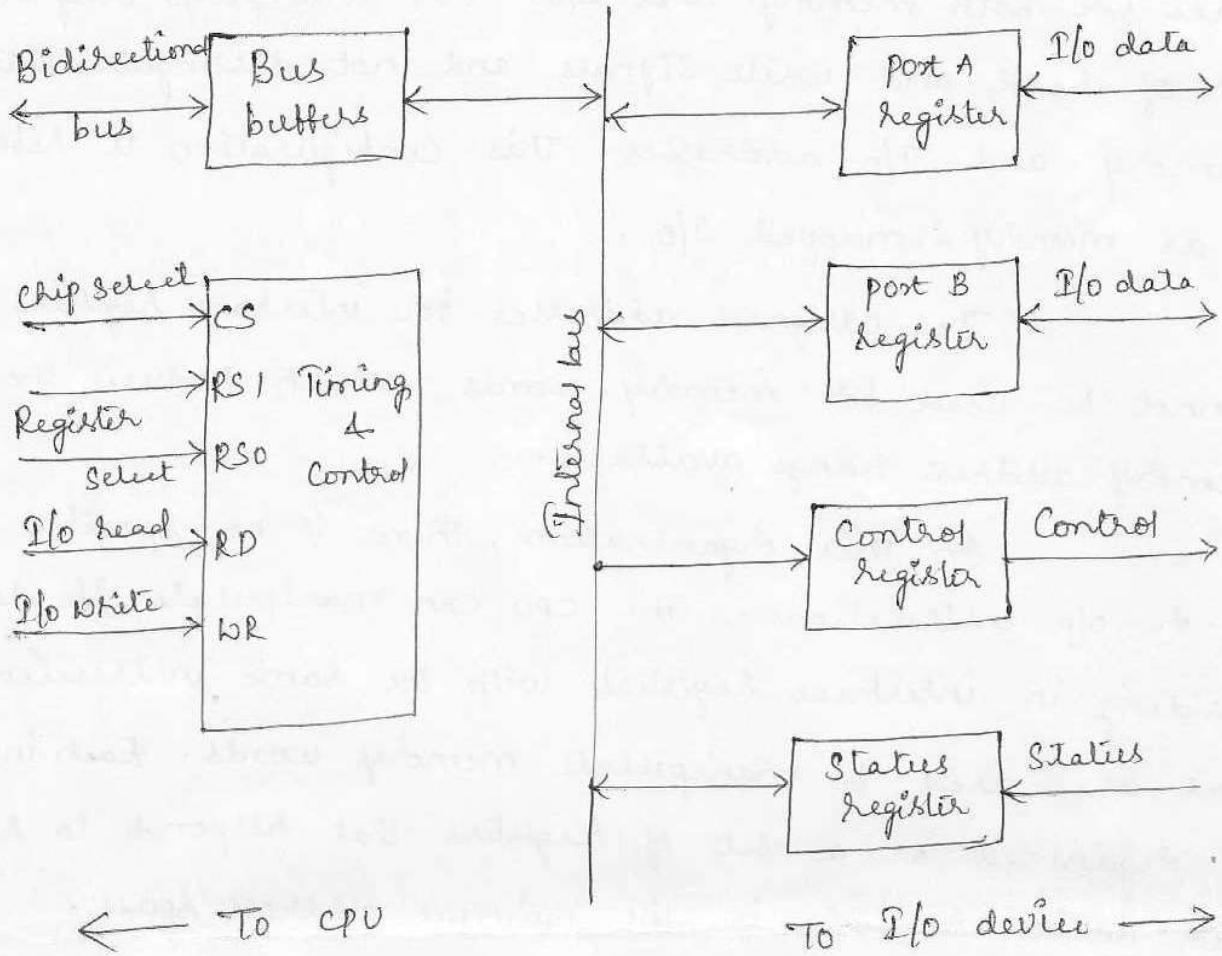
The computers who implement this organization can use memory-type instructions to access I/O data. It allows the computer to use the same instructions for either i-o transfers or for memory transfers.

The advantage is that the load and store instructions used for reading and writing from memory, can be used as input and output data from I/O registers.

<u>Example of I/O interface</u> :-

An example of an I/O interface unit is shown in the fig.

www.jntuworldupdates.org



CS	RS1	RS0	Register Selected
0	X	X	None: data bus in high impedance
1	0	0	port A register
1	0	1	port B register
1	1	0	Control register
1	1	1	Status register

www.jntuworldupdates.org

The example consists of two data registers called 'ports', a control register, a status register, bus buffers and timing and control circuits.

The interface communicates with the CPU through the data bus. The chip select and register select inputs determine the address assigned to the interface. The I/O read and write are two control lines that specify an i/p or o/p respectively.

The four registers communicate directly with the I/O device attached to the interface.

The I/O data to and from the device can be transferred into either port A or port B. The interface may operate with an o/p device or with an i/p device or with a device that requires both i/p and o/p.

If the interface is connected to a printer, it will only o/p data. But a magnetic disk unit transfers data in both directions, so the interface can use bidirectional lines.

A command is passed to the I/O device by sending a word to the appropriate interface register. In a system, the function code in the I/O bus is not needed because control is sent to the control register, status information is received from the status register and data are transferred to and from ports A and B registers. Thus the transfer of data, control and status information is always through the common data bus.

The difference between data, control or

www.jntuworldupdates.org

Status information is determined from the particular interface register with which the CPU communicates.

The control register receives control information from the CPU. By loading appropriate bits into the control register, the interface and the I/O device attached to it.

for eg, port A may be defined as i/p port and port B may be defined as o/p port. A magnetic tape unit may be instructed to rewind the tape or to start the tape moving in the forward direction.

The bits in the status register are used for status conditions and for recording errors that may occur during the data transfer.

for eg, a status bit may indicate that port A has received a new data item from the I/O device. Another bit indicates that, there is an error occured during data transfer.

The interface registers communicate with the CPU through the bidirectional data bus. The address bus selects the interface unit through the chip select and the two register select inputs.

A circuit must be provided externally, to detect the address assigned to the interface registers. This circuit enables the chip select (CS) i/p when the interface is selected by the address bus.

www.jntuworldupdates.org

Asynchronous Data Transfer :-

Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted.

There are two different methods to achieve this.

1) Strobe 2) Handshaking

Strobe :-

The strobe pulse supplied, one of the units to indicate to the other units when the transfer has to occur.

Hand Shaking :-

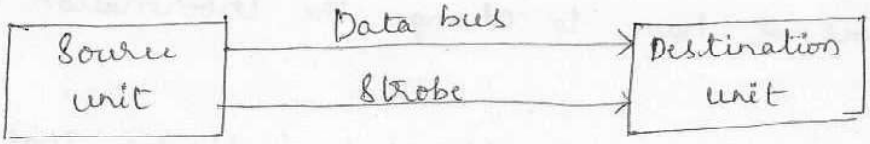
Each data item is being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units called 'Hand shaking'.

Strobe Control :-

The strobe control method of asynchronous data transfer employs a single control line to time each transfer. The strobe may be activated by either source or destination.

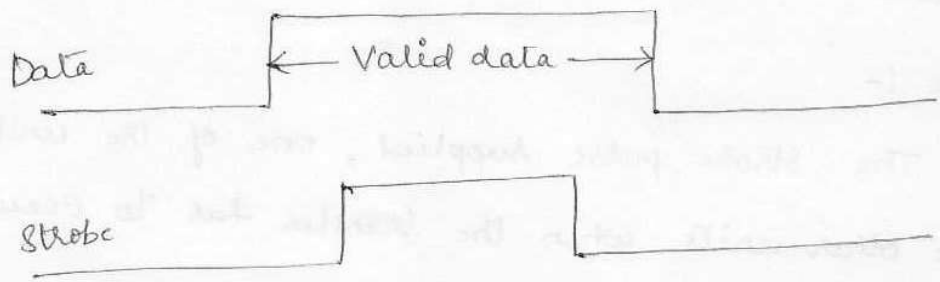
(i) Source - initiated transfer :-

The below fig. shows the source - initiated transfer.



The data bus carries the binary information from source to destination unit. The bus has multiple lines to transfer an entire byte or word. The strobe is a single line that informs the destination when a valid data word is available in the bus.

The below shows the timing diagram.



In this,

The source unit first places data on the data bus.

The data settle to a steady value, the source activates the strobe pulse.

The information on the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data item.

The destination unit uses the falling edge of the strobe pulse to transfer the contents of the data bus into one of its internal registers.

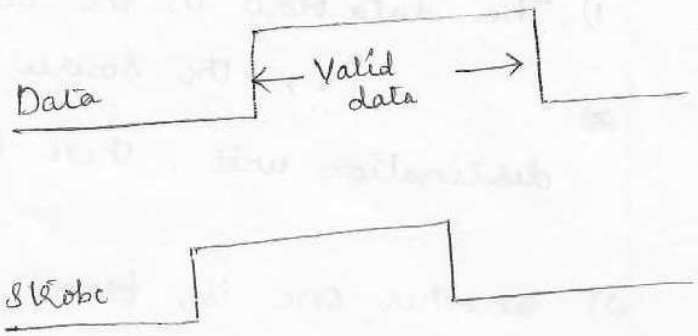
The source removes the data from the bus for a period after it disables its strobe pulse. Actually, the source does not have to change the information in the data bus.

The strobe signal is disabled indicates that the data bus

does not contain valid data.

New valid data will be available only after the strobe is enabled again.

2) Destination - initiated transfer :-



In this, the destination unit activates the strobe pulse, informing the source to provide the data.

The source unit responds by placing the requested binary information on the data bus.

The data must be valid and remain in the bus long enough for the destination unit to accept it.

The falling edge of the strobe pulse can be used again to trigger a destination register.

The destination unit then disables the strobe. The source removes the data from the bus after a predetermined time interval.

Hand shaking :-

In the strobe method, there is no reply from destination unit to source unit, whether it receives a particular data item or not.

The same happens to the destination unit, whether the source unit has actually placed the data on the bus.

www.jntuworldupdates.org

The handshake method solves this problem, by introducing a second control signal that provides a reply to the unit that initiates the transfer.

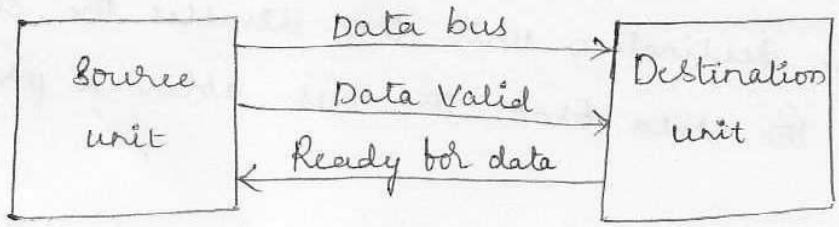
The basic principle is 'two - wire'

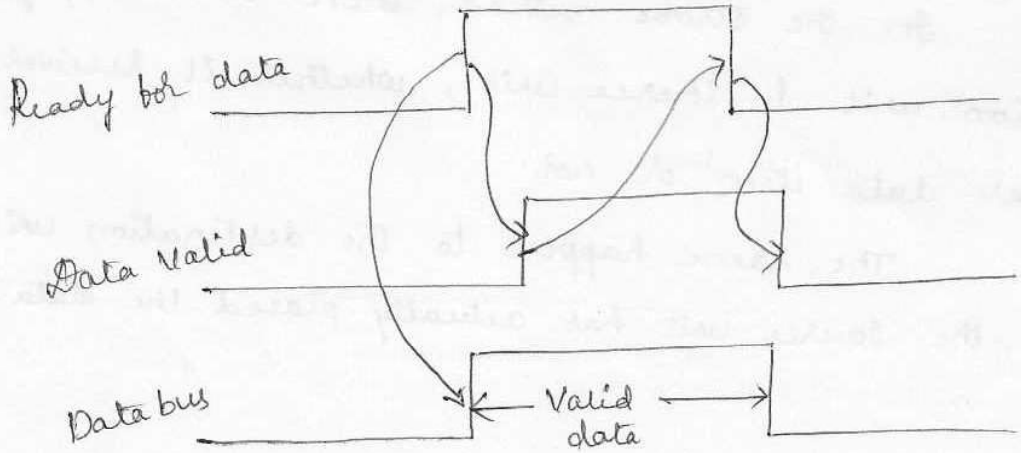
1) The data flow in the bus from source to destination. i.e., the source unit makes information to destination unit, there is a valid data on the bus.

2) another one is, from destination to source. i.e., destination unit informs that, the data is received from the source unit.

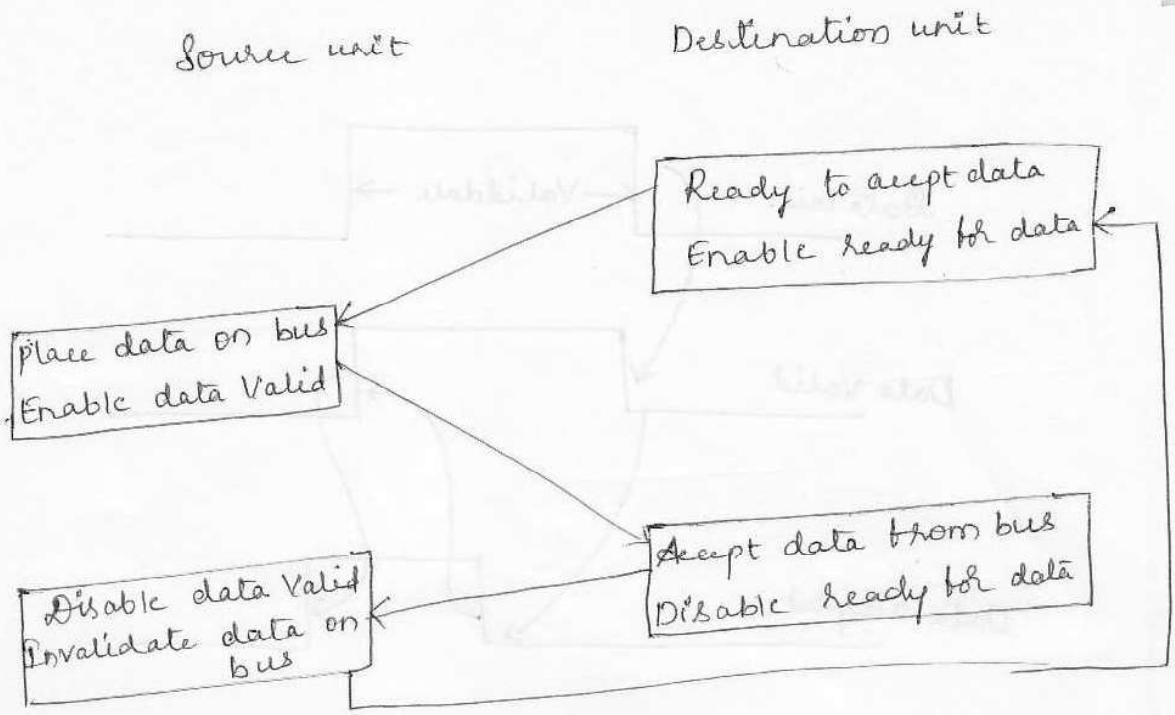
Destination ~~Source~~ - initiated transfer using handshaking :-

The below fig. shows different procedures how the data transfer exists.





www.jntuworldupdates.org



Here, we are introducing two hand-shaking lines are

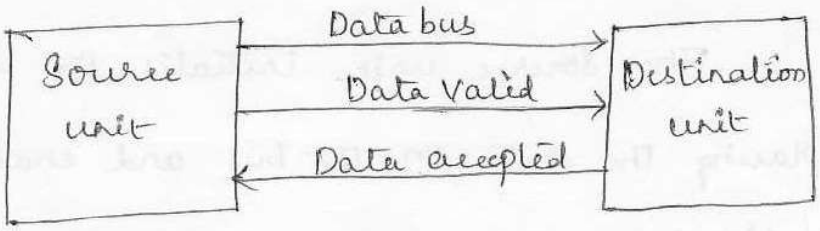
1) data valid, generated by source unit.

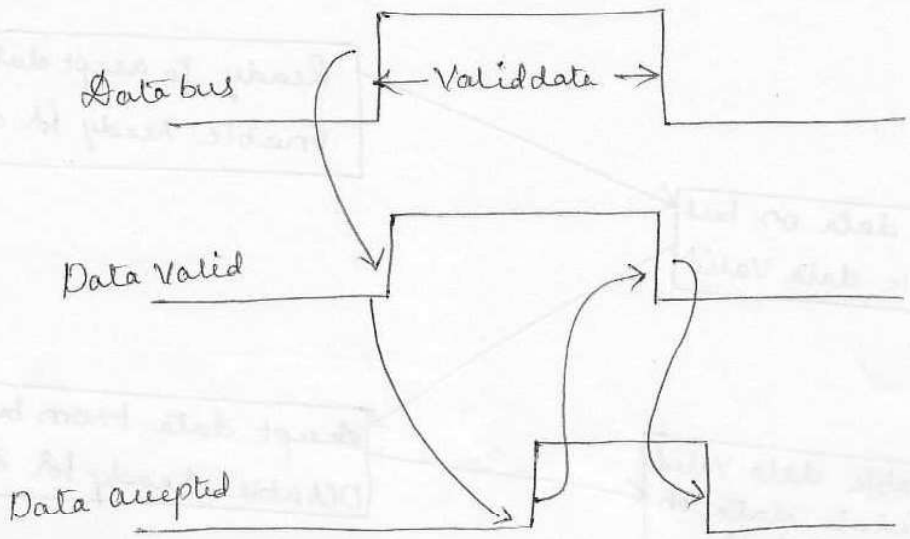
2) data accepted, generated by destination unit.

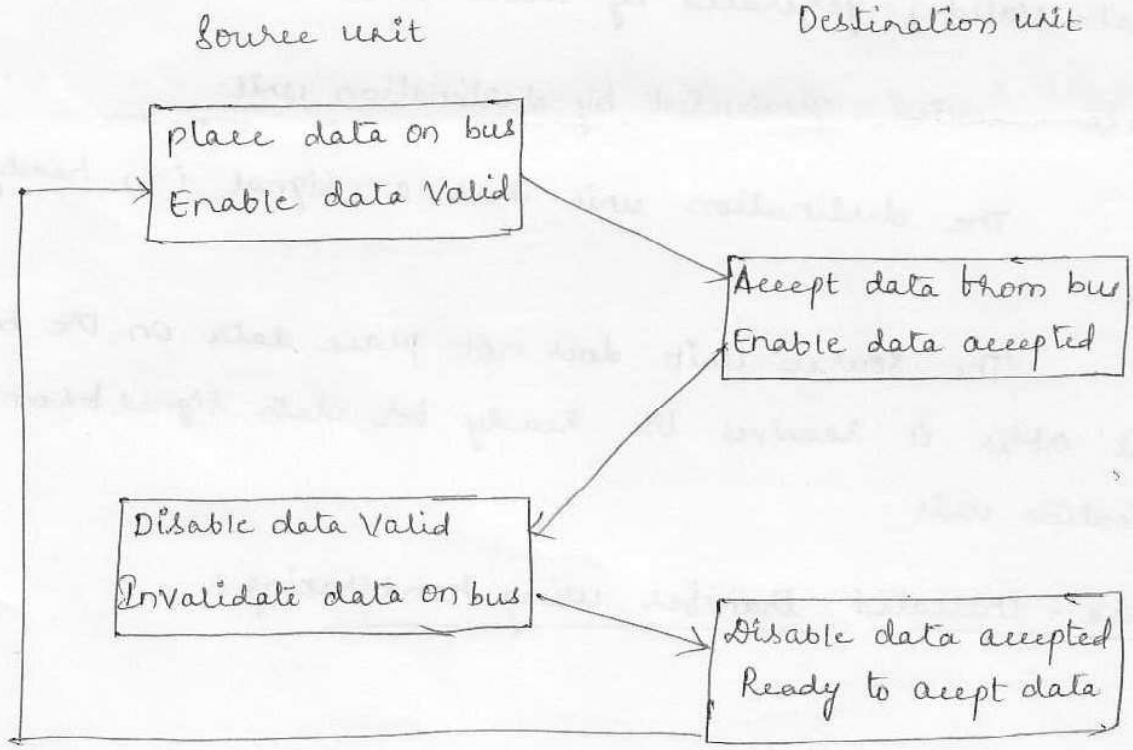
The destination unit has a signal i.e, ready for data.

The source unit does not place data on the bus until after it receives the ready for data signal from the destination unit.

<u>Source - initiated transfer using handshaking :-</u>







The source unit initiates the transfer of data by placing the data on the bus and enabling its 'data Valid' signal.

The data accepted signal is activated by

www.jntuworldupdates.org

the destination unit after it accepts the data from the bus.

Then source unit disables its data valid signal, which invalidates the data on the bus.

The destination unit then disables its data accepted signal and the system goes into its initial state.

The source does not send the next data item until after the destination unit shows its readiness to accept new data by disabling its 'data accepted signal'.

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

If one unit is faulty, the data transfer will not be completed. Such an error can be detected by means of a timeout mechanism, which produces an alarm if the data transfer is not completed with in a predetermined time. The time out is implemented by means of an internal clock that starts counting time when the unit enables one of its handshaking control signals.

If the return handshake signal does not respond with in a given time period, the unit assumes that an error has occured. The time out signal can be used to interrupt the processor and hence execute a service routine that takes appropriate error recovery action.

15

Asynchronous Serial Transfer :-

The transfer of data between two units may be done in parallel or serial.

In parallel data transmission, each bit of the message has its own path and the total message is transmitted at the same time. In this, n-bit message must be transmitted through 'n' separate conductor paths.

In serial data transmission, each bit in the message is sent in sequence one at a time.

parallel data transmission is faster than the serial data transmission.

Serial transmission can be either synchronous or asynchronous.

In synchronous, the bits are transmitted continuously at the rate of clock pulses. In this, each bit is driven a separate clock of the same frequency.

In asynchronous, binary information is sent only when it is available and the line remains idle when there is no information to be transmitted.

In a asynchronous serial transfer, a special bits are inserted at both ends of the character code. Each character consists of three parts

1) Start bit 2) character bits 3) Stop bits.

The first bit called the start bit always starts with '0' and it indicate the beginning of a character.

The last bit called the stop bit is always a '1'.

www.jntuworldupdates.org

A transmitted character can be detected by the receiver from knowledge of the transmission rules.

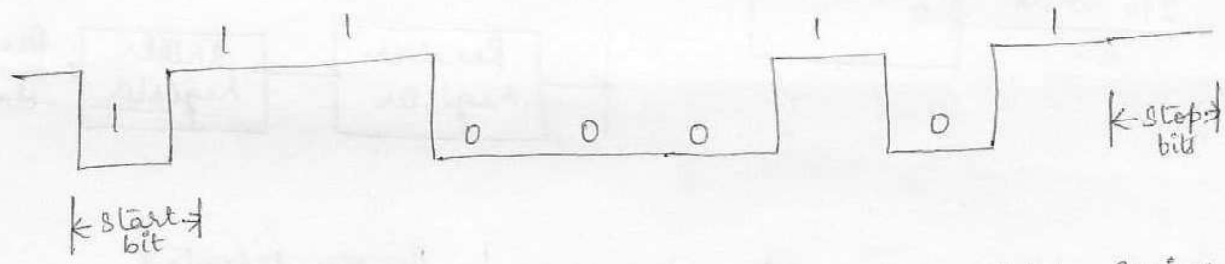
1) When a character is not being sent, the line is kept in the 1-state.

2) The initiation of a character transmission is detected from the start bit, which is always '0'.

3) The character bits always follow the start bit.

4) After the last bit of the character is transmitted, a stop bit is detected when the line returns to the 1-state for atleast one bit time.

Eg: 11000101



The 'baud rate' is defined as the rate at which serial information is transmitted and the data transfer in bits per second.

Integrated circuits are available which are specifically designed to provide the interface between computer and similar interactive terminals. Such a circuit is called an asynchronous communication interface or a universal asynchronous receiver transmitter (UART).

www.jntuworldupdates.org

This byte is transferred to a shift register for serial transmission. The receiver portion receives serial information into another shift register, and when a complete data byte is accumulated, it is transferred to the receiver register. The CPU can select the receiver register to read the byte through the data bus.

The bits in the status register are used for input and output flags and for recording certain errors that may occur during the transmission.

The chip select is used to select the interface through the address bus. The register select (RS) is associated with the read (RD) and write (WR) controls. Two registers are write-only and two are read-only.

Operation :-

The operation is initialized by the CPU by sending a byte to the control register. The initialization procedure places the interface in a specific mode of operation as it defines certain parameters such as the baud rate to use, how many bits are in each character, whether to generate and check parity and how many stop bits are appended to each character.

In status register, the two bits are used as flags. One bit is used to indicate whether the transmitter register is empty and another bit is used to indicate whether the receiver register is full.

Transmitter :-

The operation of transmitter is as follows :

- The CPU reads the status register and checks the flag to

see if the transmitter register is empty. If it is empty, the CPU transfers a character to the transmitter register and the interface clears the flag to mark the register full.

- The first bit in the transmitter shift register is set to '0' to generate a start bit. This character is transferred in parallel from the transmitter register to shift register and appropriate stop bits are appended in the shift register. The transmitter register is empty.

- The character is transmitted one bit at a time by shifting the data in the shift register at the specified baud rate.

- The CPU can transfer another character to the transmitter register after checking the flag in the status register.

- The interface is double buffered because a new character can be loaded as soon as the previous one starts transmission

Receiver :-

- The receiver data input is in the 1-state when the line is idle.

- The receiver control monitors the receive-data line for a '0' signal to detect the occurrence of a start bit. Once a start bit has been detected, the incoming bits of the character are shifted into the shift register at the prescribed baudrate.

- After receiving the data bits, the interface checks for the parity and stop bits.

- The character with out the start and stop bits is then

www.jntuworldupdates.org

transferred in parallel from the shift register to the receiver register.

- The flag in the status register is set to indicate that the receiver register is 'full'.

- The CPU reads the status register and checks the flag, and if set, it reads the data from the receiver register.

Errors :-

There are three possible errors, that the interface checks during transmission are

1) parity error 2) framing error 3) Overrun error.

A parity error occurs if the number of 1's in received data is not correct parity.

A framing error occurs if the right number of stop bits is not detected at the end of the received character.

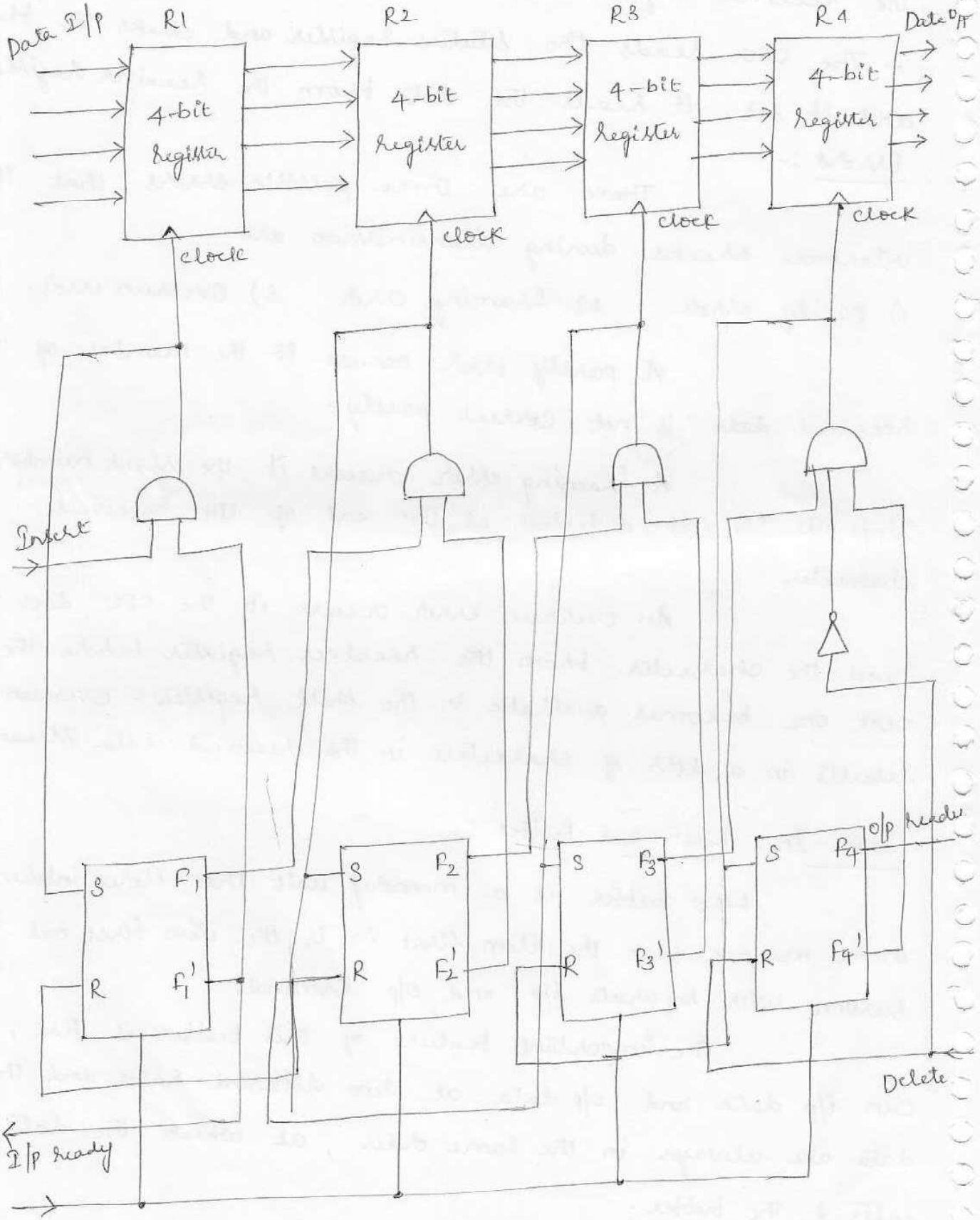
An overrun error occurs if the CPU does not read the character from the receiver register before the next one becomes available in the shift register. Overrun error results in a loss of characters in the received data stream.

First - In, First - Out Buffer :-

FIFO buffer is a memory unit that stores information in a manner that the item first in is the item first out. It survive with separate i/p and o/p terminals.

The important feature of this buffer is that, it can i/p data and o/p data at two different rates and the o/p data are always in the same order, at which the data entered the buffer.

The logic diagram of a typical 4x4 FIFO buffer is shown in fig.



www.jntuworldupdates.org

It consists of four 4-bit registers RI, $I = 1,2,3,4$ and a control register with flip-flops f_i, $i = 1,2,3,4$ one for each register.

The FIFO can store four words of four bits each. The no. of bits per word can be increased by increasing the no. of bits in each register and the no. of words can be increased by increasing the number of registers

A flip-flop f_i in the control register that is set to '1' indicates that a 4-bit data word is stored in the corresponding register RI.

A '0' in f_i indicates that the corresponding register does not contain valid data.

The control register directs the movement of data through the registers.

Whenever the f_i bit of the control register is set $f_i = 1$ and the f_{i+1} bit is reset $(f'_{i+1} = 1)$, a clock is generated causing register $R(I+1)$ to accept the data from register RI.

The clock transition sets f_{i+1} to '1' and resets f_i to '0'. This causes the control flag to move one position to the right together with the data

Data are inserted into the buffer provided that the input ready signal is enabled. This occurs when the first control flip-flop f_1 is reset, indicating that the register $R1$ is empty.

Data are loaded from the I/p lines by enabling the clock in $R1$ through the 'insert' control line. The same clock

www.jntuworldupdates.org

sets F_1, which disables the i/p ready control, indicating that the FIFO is now busy and unable to accept more data.

The ripple - through process begins provided that R2 is empty. The data in R1 are transferred into R2 and F_1 is cleared. This enables the i/p ready line, indicating that the i/p's are now available for another data word.

If the FIFO is full, F_1 remains set and the i/p ready line stays in the '0' state.

The data falling through the registers stack up at the o/p end. The o/p ready control line is enabled when the last control flip-flop F_4 is set, indicating that there are valid data in the o/p register R4.

The o/p data from R4 are accepted by a destination unit, which then enables the 'delete' control signal. This resets F_4 causing o/p ready to disable, indicating that the data on the o/p are no longer valid.

MODES OF TRANSFER

Data transfer between the central computer and I/O devices may be handled in a variety of modes. Data transfer to and from peripherals may be handled in one of three possible modes:

1) Programmed I/O
2) Interrupt-Initiated I/O
3) Direct memory Access

Programmed I/O:

Programmed I/O operations are the result of I/O instructions written in the computer program . Each data item transfer is initiated by an instruction in the program . The transfer is to and from a CPU register and peripheral.Other instructions are needed to transfer the data to and from CPU and memory . Transferring data under program control requires constant monitoring of the peripheral by the CPU.

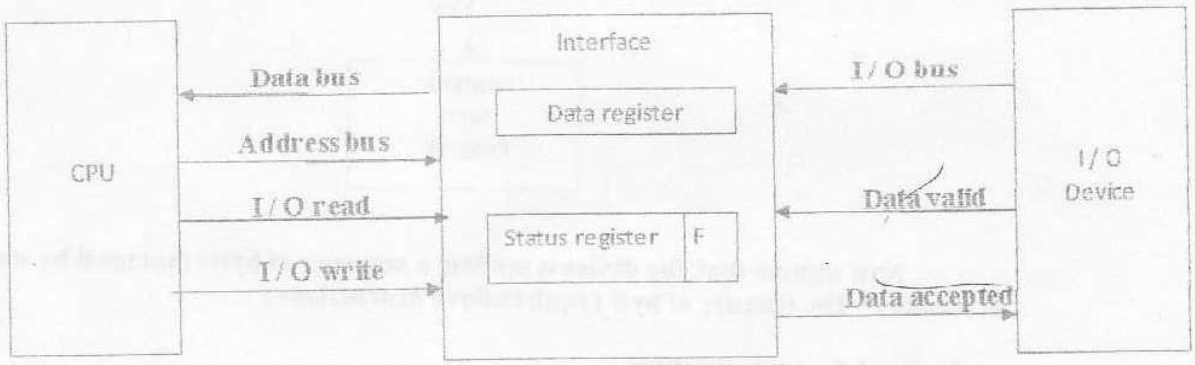
In the Programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer.

This is a time-consuming process since it keeps the processor busy needlessly

Example:

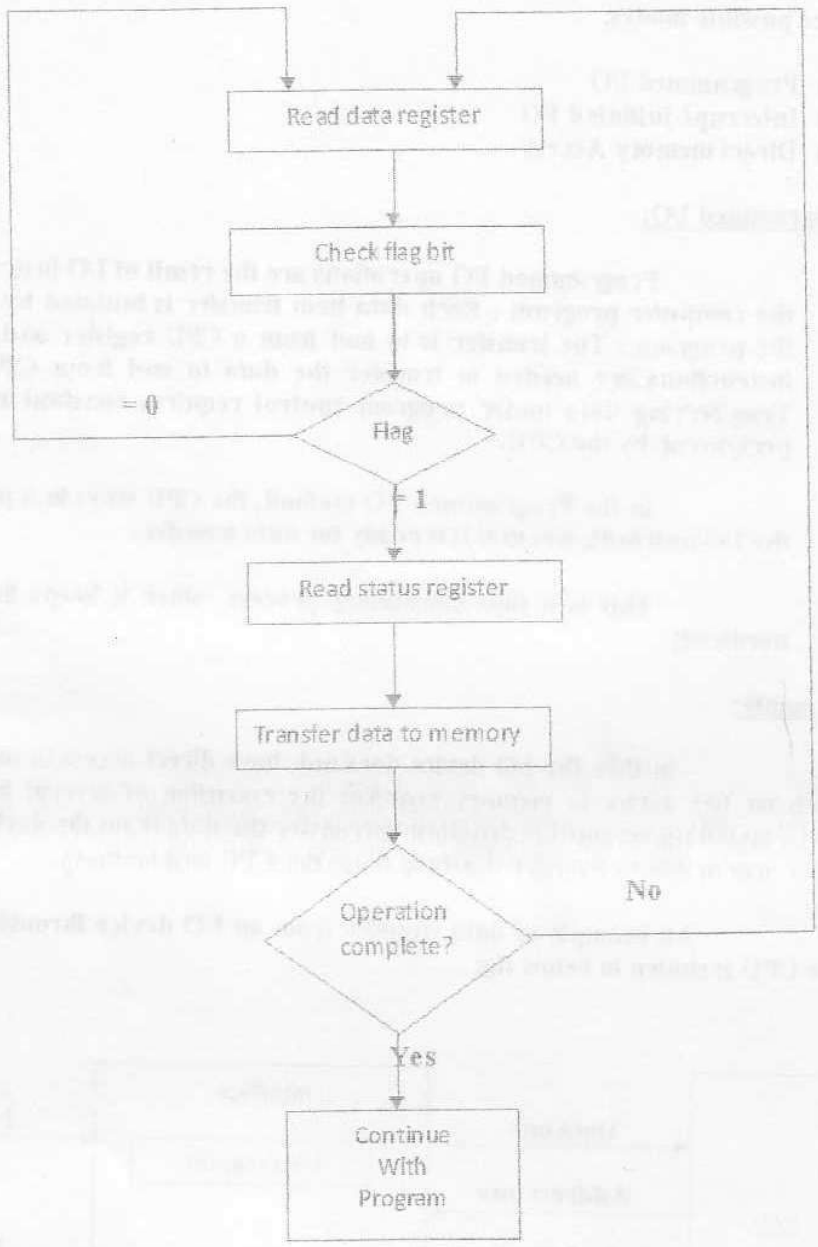
In this, the I/O device does not have direct access to memory. A transfer from an I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from the device to the CPU and store instruction to transfer the data from the CPU and memory.

An example of data transfer from an I/O device through an interface into the CPU is shown in below fig:



The device transfers bytes of information one at a time .When a byte of data is available , the device places it in the I/O bus and enables the data valid line. The interface accepts the byte into its data register and enables the data accepted line . The interface sets a bit in the status register as F bit. The device can disable the data valid line ,but it will not transfer another byte until the data accepted line is disabled by the interface .

The flow chart of the program that must be written for the CPU is shown below:



Now assume that the device is sending a sequence of bytes that must be stored in memory . The transfer of byte requires three instructions :

1) Read the Status register .
2) Check the status of the flag bit and branch to step 1 if not set or to step3 if set.
3) Read the data register.

Each byte is read into a CPU register and then transferred to memory with a store instruction .A common I/O programming task is to transfer a block of words from an I/O device and store them in a memory buffer .

This method is used in small low-speed computers.
Here the CPU is wasting time while checking the flag instead of doing some other useful processing task.

Interrupt-Initiated I/O :

Programmed I/O is an time-consuming process ,so in order to keep CPU in a busy state introduce an interrupt and special commands to inform the interface to issue an interrupt request signal when data are available from the device. In the mean time ,the CPU can proceed to execute another program

When the device is ready for data transfer , it generates an interrupt request to the computer.

When there is an external interrupt signal is generated ,the CPU will stops the execution of the original program and branches to service program to process the I/O transfer and then returns back to the original program.

The CPU responds to the interrupt signal by storing the return addresses from program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer.

The way the processor chooses the branch address of the service routine varies from one unit to another. There are two methods for using this:

1) Vectored interrupt
2) Non Vectored interrupt

In a nonvectored interrupt , the branch address is assigned to fixed location in memory.

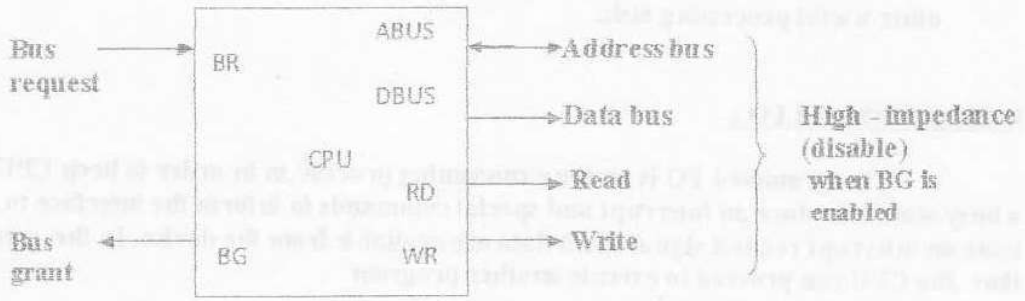
In vectored interrupt, the source that interrupts supplies the branch information to the computer. This information is called the interrupt vector.

Direct Memory Access (DMA) :

The transfer of data between a fast storage device such as magnetic disk is often limited by the speed of the CPU. Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called DMA. During DMA transfer,the cpu is idle and has no control of the memory buses . A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.

The CPU may be in idle state in many ways. One method is used in microprocessors is to disable the buses through special control signals.

The below fig. shows the control signals of the CPU to facilitate the DMA transfer.



The bus request (BR) input is used by the DMA controller to request the CPU to get control of buses.When this is input is active ,the CPU terminates the execution of the current instruction and places the address bus, data bus and read & write lines into the high-impedance state.

The CPU activates the bus grant(BG) output to inform the external DMA that the buses are in a high-impedance state. Then the DMA will take over the control of the buses with out any processor intervention. When the DMA terminates the transfer,it disables the bus request line.the CPU disables the bus grant,takes control of the buses and returns to its normal operation.

When the DMA takes control of the bus system,it communicates directly with the memory. The transfer can be done in different ways:

1) Burst Transfer: In this , a block sequence consisting of a number of memory words is transferred in a continuous burst while the DMA controller is master of all the buses. This Mode of transfer can be used by fast device such as magnetic disks etc.,

2) Cycle stealing : In this , the DMA controller to transfer one data word at a time ,after which it must return control of the buses to the CPU .The CPU delays its operation ffor one memory cycle to allow the direct memory I/O transfer to steal one memory cycle.

DMA Controller:

The below figure shows the DMA controller. The communication with the CPU via the data bus and control lines.

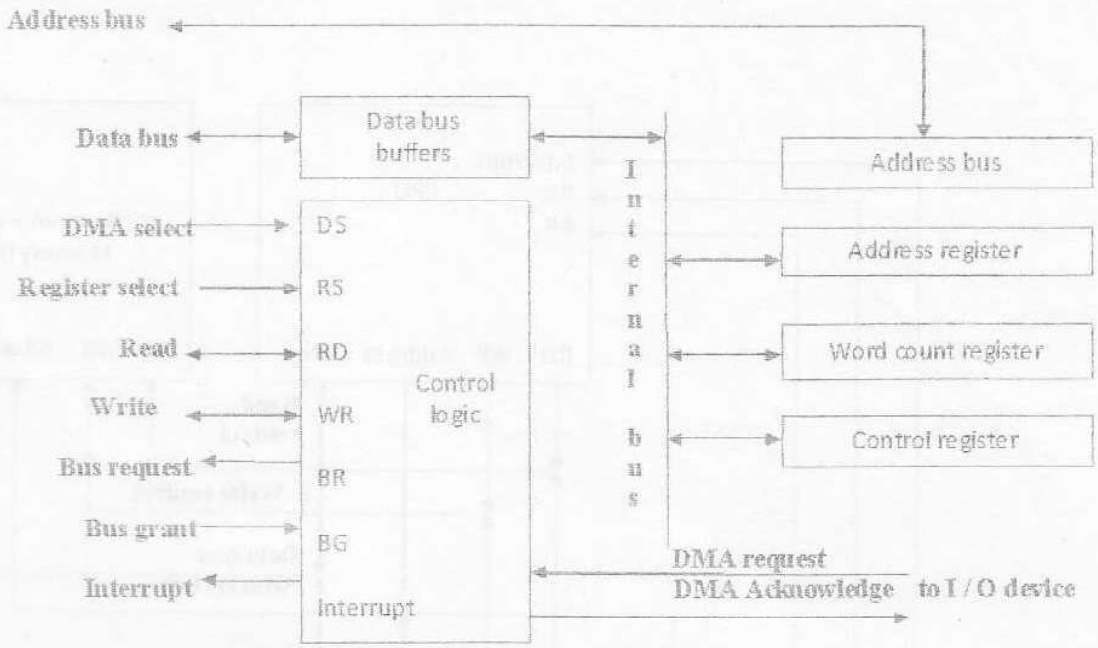
The registers in the DMA are selected by the CPU through the address bus by enabling the DS(DMA select) and RS (register select) inputs . The read and write inputs are bidirectional . When the BG(bus grant) input is 0 , the CPU communicate with the DMA registers through the data bus to read from or write to the DMA registers

When BG=1 , the CPU has the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.

The DMA communicates with the external peripheral through the request and acknowledge lines.

The DMA controller has three registers:

1) address register
2) word count register
3) control register.



The address register contains an address to specify the desired location in memory . The address bits go through bus buffers into the address bus. The address register is incremented after each word that is transferred to memory.

The word count register holds the number of words to be transferred. This register is decremented by one after each word is transferred .

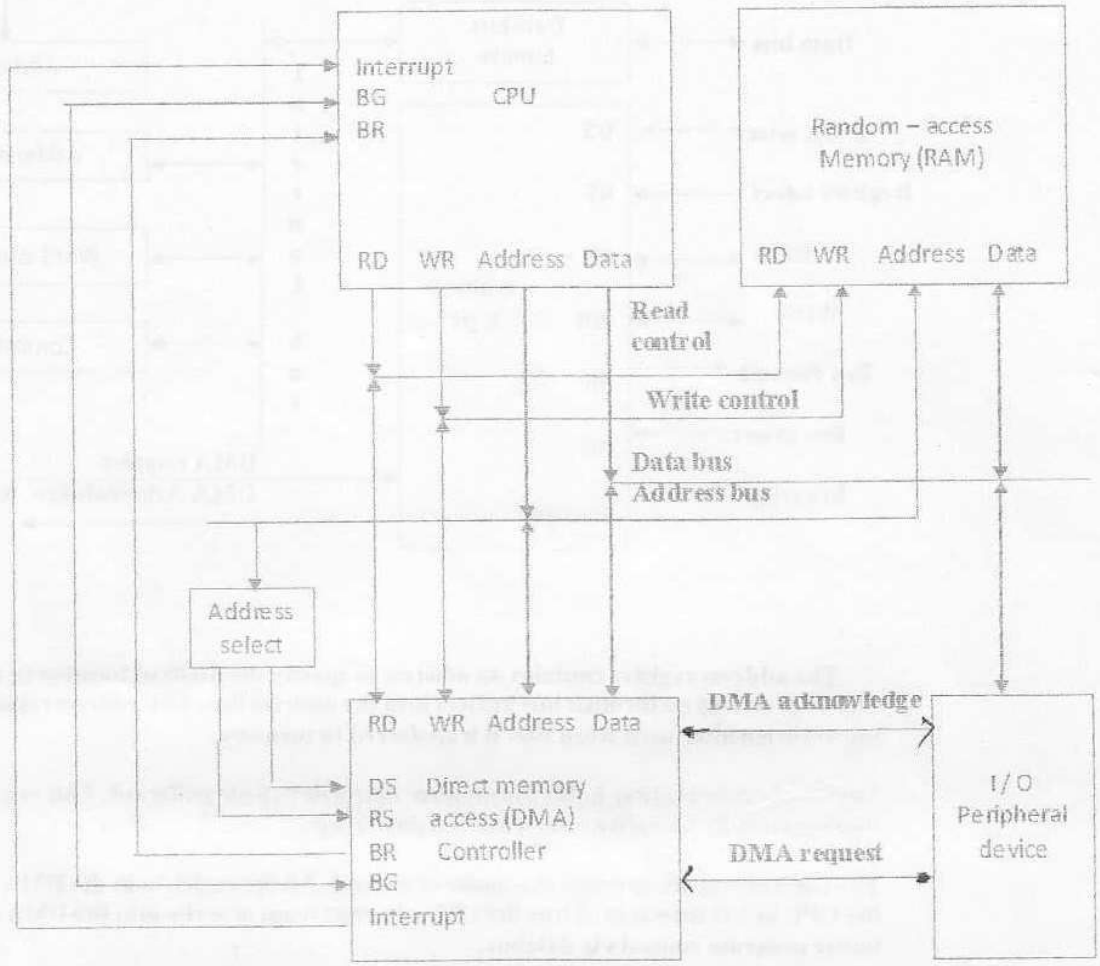
The control register specifies the mode of transfer. All the registers in the DMA appear to the CPU as I/O interface . Thus the CPU can read from or write into the DMA registers under program control via databus.

The CPU initializes the DMA by sending the following information through the databus:

1) The starting address of the memory block where data are available(for read) or where data are to be stored(for write)
2) The word count , which is the number of words in the memory block.
3) Control to specify the mode of transfer such as read or write.
4) A conrol to start the DMA transfer.

DMA transfer :

The position of the DMA controller among the other components in a system is illustrated is shown in below fig::



The CPU communicates with the DMA through the data and address buses with any interface unit. The DMA has its own address, which activates the DS and RS lines. The CPU initializes the DMA through the data bus.

Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory.

When the peripheral devices sends a DMA request, the DMA controller activates the BR line, informing the CPU to relinquish the buses.

X[The CPU initializes the DMA through the data bus. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory.]X

www.jntuworldupdates.org

~~When the peripheral device sends a DMA request,~~

The CPU responds with its BG line, informing the DMA that its buses are disabled. The DMA then puts the current value of its address register into the address bus, initiates the RD & WR signal, and sends a DMA acknowledge to the peripheral device.

Both RD and WR lines in the DMA controller are bidirectional. The direction of transfer depends on the status of the BG line. When BG=0, the RD and WR are i/p lines allowing the CPU to communicate with the internal DMA registers. When BG=1, the RD and WR are o/p lines from the DMA controller to the random-access memory to specify the read or write operation for the data.

When the peripheral device receives a DMA acknowledge, it puts a word in the data bus (for write) or receives a word from the data bus (for read). Thus the DMA controls the read or write operations and supplies the address for the memory.

The peripheral unit can then communicate with memory through the data bus for direct transfer between the two units while the CPU is momentarily disabled.

For each word that is transferred, the DMA increments its address register and decrements its word-count register. If the word count doesn't reach zero, the DMA checks the request line coming from the peripheral.

For a high speed device, the line will be active as soon as the previous transfer is completed. A second transfer is initiated, and the process continues until the entire block is transferred.

If the peripheral device speed is slow, then the DMA request line will become date. In this case, the DMA disables the bus request line so that the CPU can continue to execute its program. When the peripheral requests a transfer, the DMA requests the buses again.

If the word count register reaches zero, the DMA stops the transaction and removes its bus request.

It also informs the CPU of the termination by means of an interrupt. Now, the CPU responds to the interrupt, it reads the content of the word count register. The zero of this register indicates that all the words were transferred successfully.

If the DMA controller has more than channel. Each channel has a request and acknowledge pair of control signals which are connected to separate peripheral devices.

Each channel also has its own address register and word count register with in the DMA controller. A priority among the channels may be established so that the channels with high priority are serviced before channels with lower priority.

It is used for fast transfer of information between magnetic disks and memory.

Input - Output Processor (IOP) :—

A computer may incorporate one (or) more external processors and assign them the task of communicating directly with all I/O devices, instead of having each interface communicate with the CPU.

An Input-Output Processor (IOP), is a processor with direct memory access capability that communicates with I/O devices. Each IOP takes care of input and output tasks involved in I/O transfers.

A processor that communicates with remote terminals over telephone and other communication media in a serial fashion is called a data Communication Processor.

I/O processing:—

We know that, DMA controller must be entirely set up by the CPU. where as IOP can fetch and execute its own instructions.

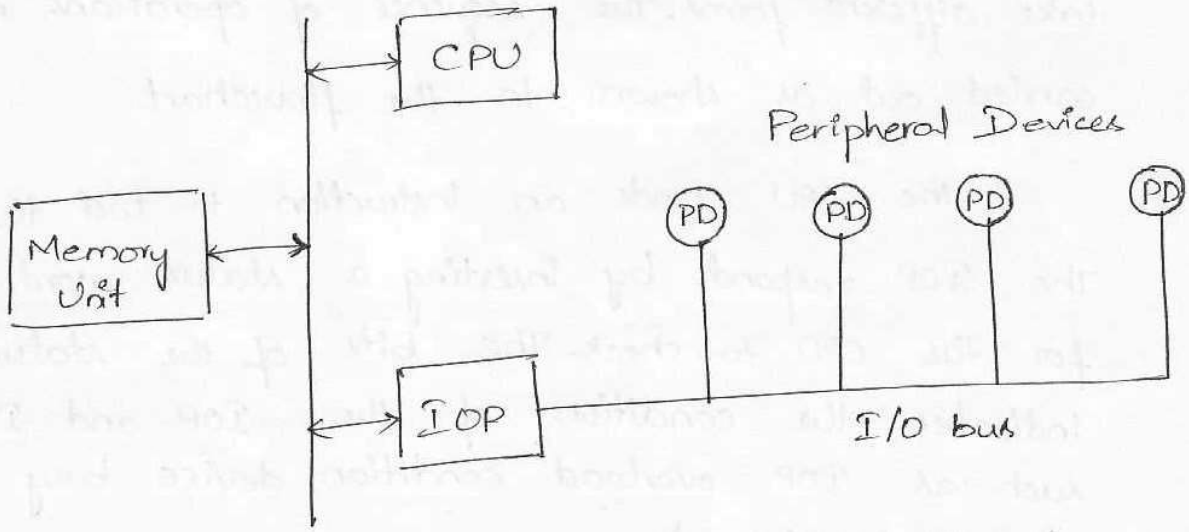


fig:- Block Diagram of a computer with I/O processor

In the block diagram, the memory unit occupies a central position and can communicate with each processor by means of direct memory access. The CPU is responsible for processing data needed in the solution of computational tasks. The IOP provides a path for transfer of data between various peripheral devices and the memory unit. The CPU is usually assigned the task of initiating the I/O program.

The data formats of peripheral devices differ from memory and CPU data formats. The IOP must structure data words from many different sources. The communication between the IOP and the devices attached to it is similar to the program control method of transfer. Communication with the CPU and IOP depends on the level of sophistication in system & communication with memory is similar to the direct access method.

CPU-IOP Communication:-

The communication between CPU and IOP may take different forms. The sequence of operations may be carried out as shown in the flowchart.

The CPU sends an instruction to test the IOP path. The IOP responds by inserting a status word in memory for the CPU to check. The bits of the status word indicates the condition of the IOP and I/O device, such as IOP overload condition, device busy with another transfer etc.,

www.jntuworldupdates.org

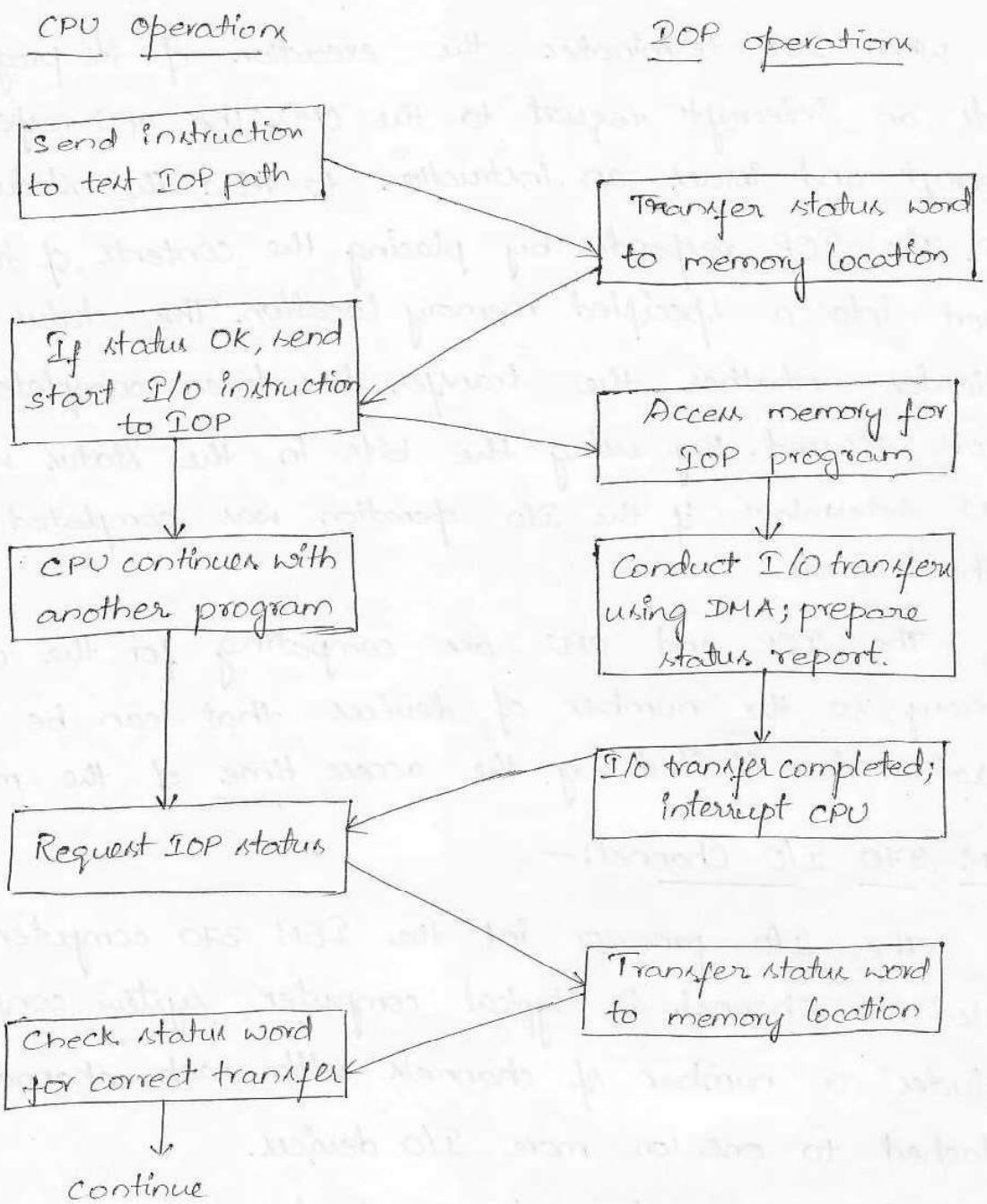


fig: CPU - IOP communication

The CPU refers to the status word in memory to decide the work to be done. Then CPU sends the instruction to start I/O transfer. The memory address received with this instruction refers to the program. & CPU can continue with another program while IOP is busy with the I/O program.

When IOP terminates the execution of its program, it sends an interrupt request to the CPU. The CPU responds to the interrupt and issues an instruction to read the status from IOP. The IOP responds by placing the contents of its status report into a specified memory location. The status word indicates whether the transfer has been completed (or) any errors occured. By using the bits in the status word, the CPU determines if the I/O operation was completed satisfactorily without errors.

The IOP and CPU are competing for the use of memory, so the number of devices that can be in operation is limited by the access time of the memory.

IBM 370 I/O Channel:—

The I/O processor in the IBM 370 computer is called a channel. A typical computer system configuration includes a number of channels with each channel attached to one (or) more I/O devices.

There are three types of channels:

i. Multiplexer

ii. Selector

iii. Block Multiplexer.

The Multiplexer channel can be connected to a number of slow and medium-speed devices. It is capable of operating with a number of I/O devices simultaneously.

www.jntuworldupdates.org

The selector channel is designed to handle one I/O operation at a time. It is normally used to control one high-speed device.

The block-Multiplexer channel combines the features of both the multiplexer and selector channels.

Operation code	Channel address	Device address

(a) I/O instruction format

Key	Address	Status	Count

(b) Channel status word format

Command code	Data Address	Flags	Count

(c) Channel command word format

fig:- IBM 370 I/O related word formats

The I/O instruction format has three fields: operation code, channel address, and device address, which contains the address of channels of computer system. Each channel may be connected to several devices and its address is specified by device address. The operation code specifies one of eight I/O instructions: start I/O, start I/O fast release, test I/O, clear I/O, halt I/O, halt device, test channel & store channel. The CPU can check the condition code in the PSW (program Status Word) to determine the result of the I/O operation

www.jntuworldupdates.org

The channel status word is always stored in location 64 in memory. The key field is a protection mechanism used to prevent un-authorized access by one user information to another user. Address field gives the address of last command word used by the channel. Count field gives the residue count when the transfer was completed successfully i.e., zero. The status field identifies the conditions in the device and the channel & also errors during transfer.

In the Channel Command Word (CCW), the data address field specifies the first address of a memory buffer. The count field gives the number of bytes involved in the transfer. The command field specifies an I/O operation and the flag bits provide additional information for the channel.

The command field corresponds to an operation code that specifies one of the six types of I/O operations:

1. Read: Transfer data from I/O device to memory
2. Write: Transfer data from memory to I/O device
3. Read backwards: Read magnetic tape with tape moving backward
4. Control: Used to initiate an operation not involving transfer of data.
5. Sense: Informs the channel to transfer its channel status word to memory location 64.
6. Transfer in channel:- Used instead of a jump instruction. Here the data address field specifies the address of next command word to be executed by the channel.

www.jntuworldupdates.org

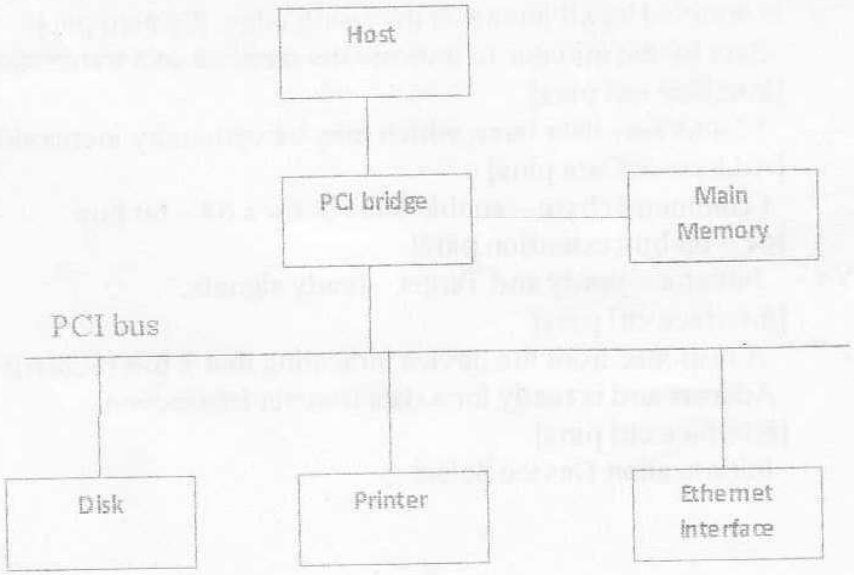
PCI: (Peripheral Component Interconnect). BUS:-

The PCI is a popular high – bandwidth, Processor – independent bus that can function as a mezzanine (or) peripheral bus. Compared with other common bus specifications, PCI delivers better system performance for high – speed I / O subsystems like graphics display adapters, network interface controllers, disk controllers and so on. PCI is specifically designed to meet economically the I / O requirements of modern systems. It requires very few chips to implement and supports other buses attached to the PCI bus.

PCI is designed to support a variety of microprocessor based configurations, including both single – and multiple – processor systems. It Provides a general – Purpose set of functions. It makes use of synchronous timing and a centralized arbitration scheme.

PCI was developed as a low – cost bus that is truly processor independent. An important feature that the PCI is a plug – and – play capability – for connecting I / O devices. To connect a new device, the user simply connects the device interface board to the bus.

Data transfer: - Most memory transfers involve a burst of data rather than one word. Data are transferred between the cache and the main memory in bursts of several words each. The words involved in such a transfer are stored at successive memory locations. When the processor specifies an address and requests a read operation from the main memory, the memory responds by sending a sequence of data words starting at that address. During a write operation the processor sends a memory address followed by a sequence of data words, to be written in successive memory locations starting at that address. The PCI is designed primarily to support this mode of operation.



www.jntuworldupdates.org

The PCI bridge provides a separate physical connection for the main memory. For electrical reasons, the bus may be further divided into segments connected via bridges. The segment device address is mapped into the processor's memory address space.

At any given time, one device is the bus master. It has the right to initiate data transfer y issuing read and write commands. A 'master' is called an "initiator" in PCI terminology. This is either a processor (or) a DMA controller. The addressed device that responds to read & write commands is called a "target".

Consider a bus transaction in which the processor reads four 32 – bit words from the memory. In this case, the initiator is the processor and the target is the memory. A complete transfer operation on the bus, involving an address and a burst of data, is called a "transaction". Individual word transfers within a transaction are called "phases". A clock signal provides the timing reference used to coordinate different phases of transaction. All signal transactions are triggered by the rising edge of the clock.

BUS Structure:- PCI may be configured as a 32 – or 64 – bit bus. PCI contains 49 mandatory signal lines. These are divided into 5 groups.

1.System pins :- Include the clock and reset pins.
2.Address & data pins
3.Interface ctrl pins
4.Arbitration pins
5.Error reporting pins.

And PCI specification defines 51 optional signal lines divided into 4 groups.

1.Interrupt pins
2.cache support pins
3.64 – bit bus extension pins
4.JTAG / boundary scan pins

PCI Commands : -

Name	Function
1.CLK -	A 33 – MHZ clock. Provides timing to all transactions and is sampled by all inputs on the rising edge. [System pin]
2.FRAME# -	Sent by the initiator to indicate the duration of a transaction. [Interface ctrl pins]
3.AD -	32 address / data lines, which may be optionally increased to 64. [Address & Data pins]
4.C/BE# -	4 command / byte – enable lines (8 for a 64 – bit bus. [64 – bit bus extension pins]
5.IRDY#, TRDY# -	Initiator – ready and Target – ready signals. [Interface ctrl pins]
6.DEVSEL# - (Device Select	A response from the device indicating that it has recognized its Address and is ready for a data transfer transaction. [Interface ctrl pins]
7.IDSWL# -	Initialization Device Select

www.jntuworldupdates.org

Serial communication standards

Serial communication is a popular means of transmitting data between a computer and a peripheral device such as a programmable instrument or even another computer. Serial communication uses a transmitter to send data, one bit at a time, over a single communication line to a receiver. We can use this method when data transfer rates are low or you must transfer data over long distances. Serial communication is popular because most computers have one or more serial ports, so no extra hardware is needed other than a cable to connect the instrument to the computer or two computers together.

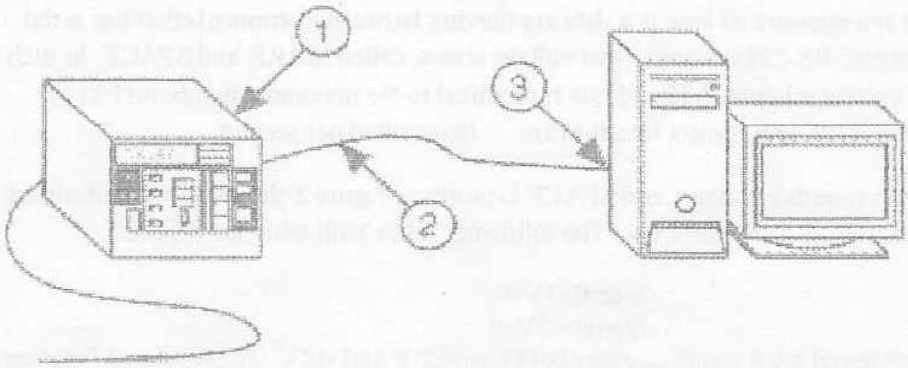


Figure 1:

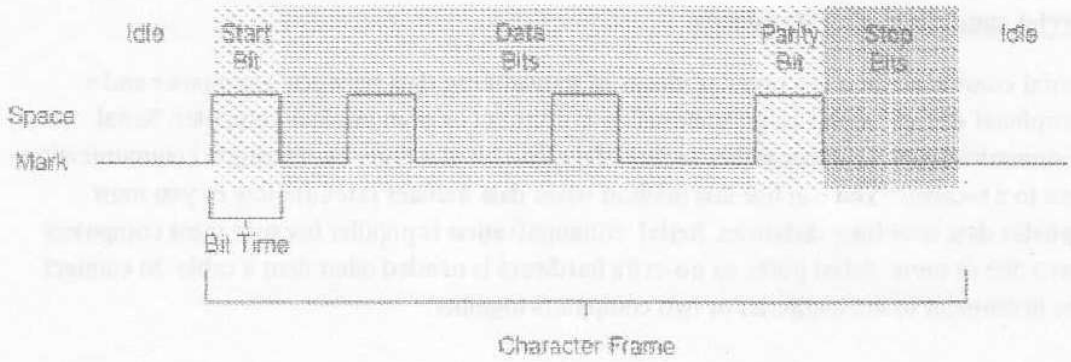
1: RS-232 Instrument 2: RS-232 Cable, 3: Serial Port

Serial communication requires that you specify the following four parameters:

1. The baud rate of the transmission
2. The number of data bits encoding a character
3. The sense of the optional parity bit
4. The number of stop bits

Each transmitted character is packaged in a character frame that consists of a single start bit followed by the data bits, the optional parity bit, and the stop bit or bits. Figure 2 shows a typical character frame encoding the letter m.

www.jntuworldupdates.org



Baud rate is a measure of how fast data are moving between instruments that use serial ommunication. RS-232 uses only two voltage states, called MARK and SPACE. In such a two-state coding scheme, the baud rate is identical to the maximum numberof bits of information, including control bits, that are transmitted per second.

MARK is a negative voltage, and SPACE is positive. Figure 2 shows how the idealized signal looks on an oscilloscope. The following is the truth table for RS-232:

Signal>3V=0
Signal>-3V=1

The output signal level usually swings between +12 V and -12V. The dead area between +3 V and -3 V is designed to absorb line noise.

A start bit signals the beginning of each character frame. It is a transition from negative (MARK) to positive (SPACE) voltage. Its duration in seconds is the reciprocal of the baud rate. If the instrument is transmitting at 9,600 baud, the duration of the start bit and each subsequent bit is about 0.104 ms. The entire character frame of eleven bits would be transmitted in about 1.146 ms.

Data bits are transmitted upside down and backwards. That is, inverted logic is used, and the order of transmission is from least significant bit (LSB) to most significant bit (MSB). To interpret the data bits in a character frame, you must read from right to left and read 1 for negative voltage and 0 for positive voltage. This yields 1101101 (binary) or 6D (hex). An ASCII conversion table shows that this is the letter m.

An optional parity bit follows the data bits in the character frame. The parity bit, if present, also follows inverted logic,1 for negative voltage and 0 for positive voltage. This bit is included as a simple means of error handling. You specify ahead of time whether the parity of the transmission is to be even or odd. If the parity is chosen to be odd, the transmitter then sets the parity bit in such a way as to make an odd number of ones among the data bits and the parity bit. This transmission uses odd parity. There are five ones among the data bits, already an odd number, so the parity bit is set to 0.

The last part of a character frame consists of 1, 1.5, or 2 stop bits. These bits are always represented by a negative voltage. If no further characters are transmitted, the line stays in the negative (MARK) condition. The transmission of the next character frame, if any, is heralded by a start bit of positive (SPACE) voltage.

Hardware Overview

There are many different recommended standards of serial port communication, including the following most common types.

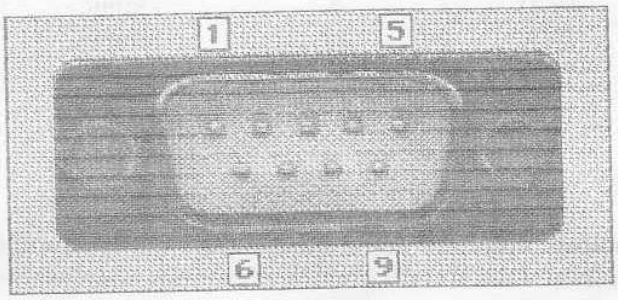
www.jntuworldupdates.org

2-12

The RS-232 is a standard developed by the Electronic Industries Association (EIA) and other interested parties, specifying the serial interface between Data Terminal Equipment (DTE) and Data Communications Equipment (DCE). The RS-232 standard includes electrical signal characteristics (voltage levels), interface mechanical characteristics (connectors), functional description of interchange circuits (the function of each electrical signal), and some recipes for common kinds of terminal-to-modem connections. The most frequently encountered revision of this standard is called RS-232C. Parts of this standard have been adopted (with various degrees of fidelity) for use in serial communications between computers and printers, modems, and other equipment. The serial ports on standard IBM-compatible personal computers follow RS-232.

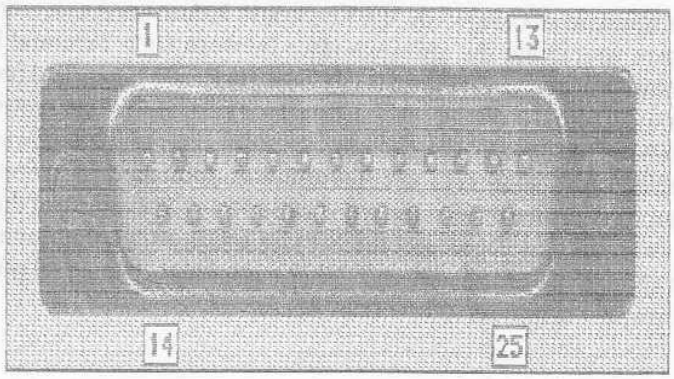
RS-232 Cabling

Devices that use serial cables for their communication are split into two categories. These are DCE and DTE. DCE are devices such as a modem, TA adapter, plotter, and so on, while DTE is a computer or terminal. RS-232 serial ports come in two sizes, the D-Type 25-pin connector and the D-Type 9-pin connector. Both of these connectors are male on the back of the PC. Thus, you require a female connector on the device. Table 1 shows the pin connections for the 9-pin and 25-pin D-Type connectors.



Function	Signal	PIN	DTE	DCE
Data	TxD	3	Output	Input
	RxD	2	Input	Output
Handshake	RTS	7	Output	Input
	CTS	8	Input	Output
	DSR	6	Input	Output
	DCD	1	Input	Output
	STR	4	Output	Input
Common	Com	5	--	--
Other	RI	9	Output	Input

The DB-9 connector is occasionally found on smaller RS-232 lab equipment. It is compact, yet has enough pins for the core set of serial pins (with one pin extra). The DB-25 connector is the standard RS-232 connector, with enough pins to cover all the signals specified in the standard. Table 2 shows only the core set of pins that are used for most RS-232 interfaces.



Function	Signal	PIN	DTE	DCE
Data	TxD	2	Output	Input
	RxD	3	Input	Output
Handshake	RTS	4	Output	Input
	CTS	5	Input	Output
	DSR	6	Input	Output
	DCD	8	Input	Output
	STR	20	Output	Input
Common	Com	7	--	--

Universal Serial Bus

In information technology, **Universal Serial Bus (USB)** is a serial bus standard to connect devices to a host computer. USB was designed to allow many peripherals to be connected using a single standardized interface socket and to improve plug and play capabilities by allowing hot swapping; that is, by allowing devices to be connected and disconnected without rebooting the computer or turning off the device. Other convenient features include providing power to low-consumption devices, eliminating the need for an external power supply; and allowing many devices to be used without requiring manufacturer-specific device drivers to be installed.

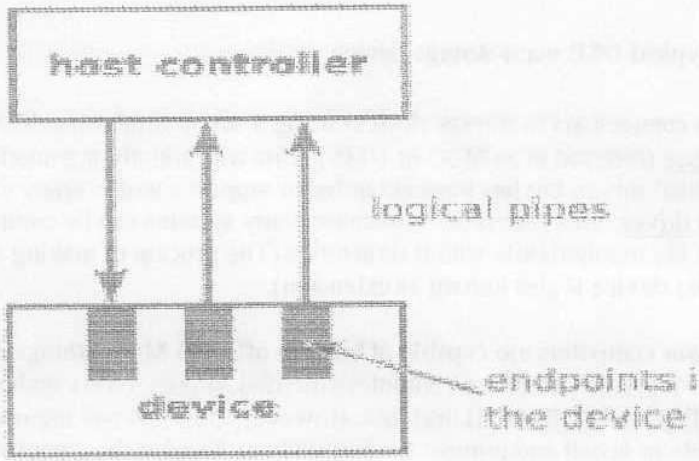
USB is intended to replace many varieties of serial and parallel ports. USB can connect computer peripherals such as mice, keyboards, PDAs, gamepads and joysticks, scanners, digital cameras, printers, personal media players, flash drives, and external hard drives. For many of those devices, USB has become the standard connection method. USB was designed for personal computers, but it has become commonplace on other devices such as PDAs and video game consoles, and as a power cord between a device and an AC adapter plugged into a wall plug for charging. As of 2008, there are about 2 billion USB devices sold per year, and about 6 billion total sold to date.[1]

The design of USB is standardized by the USB Implementers Forum (USB-IF), an industry standards body incorporating leading companies from the computer and electronics industries. Notable members have included Agere (now merged with LSI Corporation), Apple Inc., Hewlett-Packard, Intel, Microsoft and NEC.

A USB system has an asymmetric design, consisting of a host, a multitude of downstream USB ports, and multiple peripheral devices connected in a tiered-star topology. Additional USB hubs may be included in the tiers, allowing branching into a tree structure with up to five tier levels. A USB host may have multiple host controllers and each host controller may provide one or more USB ports. Up to 127 devices, including the hub devices, may be connected to a single host controller.

USB devices are linked in series through *hubs*. There always exists one hub known as the root hub, which is built into the host controller. So-called "sharing hubs", which allow multiple computers to access the same peripheral device(s), also exist and work by switching access between PCs, either automatically or manually. They are popular in small-office environments. In network terms, they converge rather than diverge branches.

A physical USB device may consist of several logical sub-devices that are referred to as *device functions*. A single device may provide several functions, for example, a webcam (video device function) with a built-in microphone (audio device function).



USB device communication is based on *pipes* (logical channels). Pipes are connections from the host controller to a logical entity on the device named an endpoint. The term *endpoint* is occasionally used to incorrectly refer to the pipe. A USB device can have up to 32 active pipes, 16 into the host controller and 16 out of the controller.

Each endpoint can transfer data in one direction only, either into or out of the device, so each pipe is uni-directional. Endpoints are grouped into *interfaces* and each interface is associated with a single device function. An exception to this is endpoint zero, which is used for device configuration and which is not associated with any interface.

When a USB device is first connected to a USB host, the USB device enumeration process is started. The enumeration starts by sending a reset signal to the USB device. The speed of the USB device is determined during the reset signaling. After reset, the USB device's information is read by the host, then the device is assigned a unique 7-bit address. If the device is supported by the host, the device drivers needed for communicating with the device are loaded and the device is set to a configured state. If the USB host is restarted, the enumeration process is repeated for all connected devices.

implementation costs and a simplified, more adaptable cabling system. The 1394 standard also defines a backplane interface, though this is not as widely used.

IEEE 1394 has been adopted as the High-Definition Audio-Video Network Alliance (HANA) standard connection interface for A/V (audio/visual) component communication and control.[1] FireWire is also available in wireless, fiber optic, and coaxial versions using the isochronous protocols.

ireWire is Apple Inc.'s name for the IEEE 1394 High Speed Serial Bus. It was initiated by Apple (in 1986[2]) and developed by the IEEE P1394 Working Group, largely driven by contributions from Apple, although major contributions were also made by engineers from Texas Instruments, Sony, Digital Equipment Corporation, IBM, and INMOS/SGS Thomson (now STMicroelectronics).

Apple intended FireWire to be a serial replacement for the parallel SCSI (Small Computer System Interface) bus while also providing connectivity for digital audio and video equipment. Apple's development began in the late 1980s, later presented to the IEEE,[3] and was completed in 1995. As of 2007, IEEE 1394 is a composite of four documents: the original IEEE Std. 1394-1995, the IEEE Std. 1394a-2000 amendment, the IEEE Std. 1394b-2002 amendment, and the IEEE Std. 1394c-2006 amendment. On June 12, 2008, all these amendments as well as errata and some technical updates were incorporated into a superseding standard IEEE Std. 1394-2008.[4]

Apple's internal code-name for FireWire was "Greyhound" as of May 11, 1992.

Sony's implementation of the system, known as "i.LINK" used a smaller connector with only the four signal circuits, omitting the two circuits which provide power to the device in favor of a separate power connector. This style was later added into the 1394a amendment.[3] This port is sometimes labeled "S100" or "S400" to indicate speed in Mbit/s.

The system is commonly used for connection of data storage devices and DV (digital video) cameras, but is also popular in industrial systems for machine vision and professional audio systems. It is preferred over the more common USB for its greater effective speed and power distribution capabilities, and because it does not need a computer host. Perhaps more important, FireWire makes full use of all SCSI capabilities and has high sustained data transfer rates, a feature especially important for audio and video editors. Benchmarks show that the sustained data transfer rates are higher for FireWire than for USB 2.0, especially on Apple Mac OS X with more varied results on Microsoft Windows.[5][6]

However, the royalty which Apple Inc. and other patent holders initially demanded from users of FireWire (US$0.25 per end-user system) and the more expensive hardware needed to implement it (US$1–$2), both of which have since been dropped, have prevented FireWire from displacing USB in low-end mass-market computer peripherals, where product cost is a major constraint.[3]

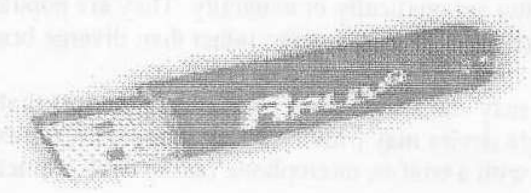
www.jntuworldupdates.org

www.jntuworldupdates.org

The host controller directs traffic flow to devices, so no USB device can transfer any data on the bus without an explicit request from the host controller. In USB 2.0, host controller polls the bus for traffic, usually in a round-robin fashion. In SuperSpeed USB, connected devices can request service from host.

USB mass-storage

Main article: USB mass storage device class



A flash drive, a typical USB mass-storage device.

USB implements connections to storage devices using a set of standards called the *USB mass storage device class* (referred to as MSC or UMS). This was initially intended for traditional magnetic and optical drives, but has been extended to support a wide variety of devices, particularly flash drives. This generality is because many systems can be controlled with the familiar idiom of file manipulation within directories (The process of making a novel device look like a familiar device is also known as extension).

Though most newer computers are capable of booting off USB Mass Storage devices, USB is not intended to be a primary bus for a computer's internal storage: buses such as ATA (IDE), Serial ATA (SATA), and SCSI fulfill that role. However, USB has one important advantage in that it is possible to install and remove devices without opening the computer case, making it useful for external drives. Originally conceived and still used today for optical storage devices (CD-RW drives, DVD drives, etc.), a number of manufacturers offer external portable USB hard drives, or empty enclosures for drives, that offer performance comparable to internal drives[citation needed]. These external drives usually contain a translating device that interfaces a drive of conventional technology (IDE, ATA, SATA, ATAPI, or even SCSI) to a USB port. Functionally, the drive appears to the user just like an internal drive. Other competing standards that allow for external connectivity are eSATA and FireWire.

Another use for USB Mass Storage devices is the portable execution of software applications without the need of installation on the host computer,[5] such as Web Browsers, VoIP clients,[6] etc.

IEEE 1394

The **IEEE 1394 interface** is a serial bus interface standard for high-speed communications and isochronous real-time data transfer, frequently used by personal computers, as well as in digital audio, digital video, automotive, and aeronautics applications. The interface is also known by the brand names of **FireWire** (Apple Inc.), **i.LINK** (Sony), and **Lynx** (Texas Instruments). IEEE 1394 replaced parallel SCSI in many applications, because of lower

www.jntuworldupdates.org

Parallel - processing :-

parallel - processing is a term used to denote a large class of technique that are used to provide simultaneous data - processing tasks.

A parallel processing system is able to perform concurrent data processing to achieve faster execution time.

For eg, while an instruction is being executed in the ALU, the next instruction can be read from memory. The system may have two or more ALUs & be able to execute two or more instructions at the same time.

The purpose of parallel processing is to speed up the computer processing capability & increase its throughput. The throughput is defined as the amount of processing that can be accomplished during a given interval of time.

The amount of h/w increases with parallel processing & the cost of the system increases.

parallel processing can be viewed from various levels of complexity. At the lowest level, we distinguish b/w parallel & serial operations by the type of registers. Shift registers operate in serial fashion one bit at a time and parallel load registers operate with the all bits of the word simultaneously.

parallel processing at higher level can be achieved by having a multiplicity of functional units that perform identical or different operations simultaneously. parallel processing is established by distributing the data among multiple functional units.

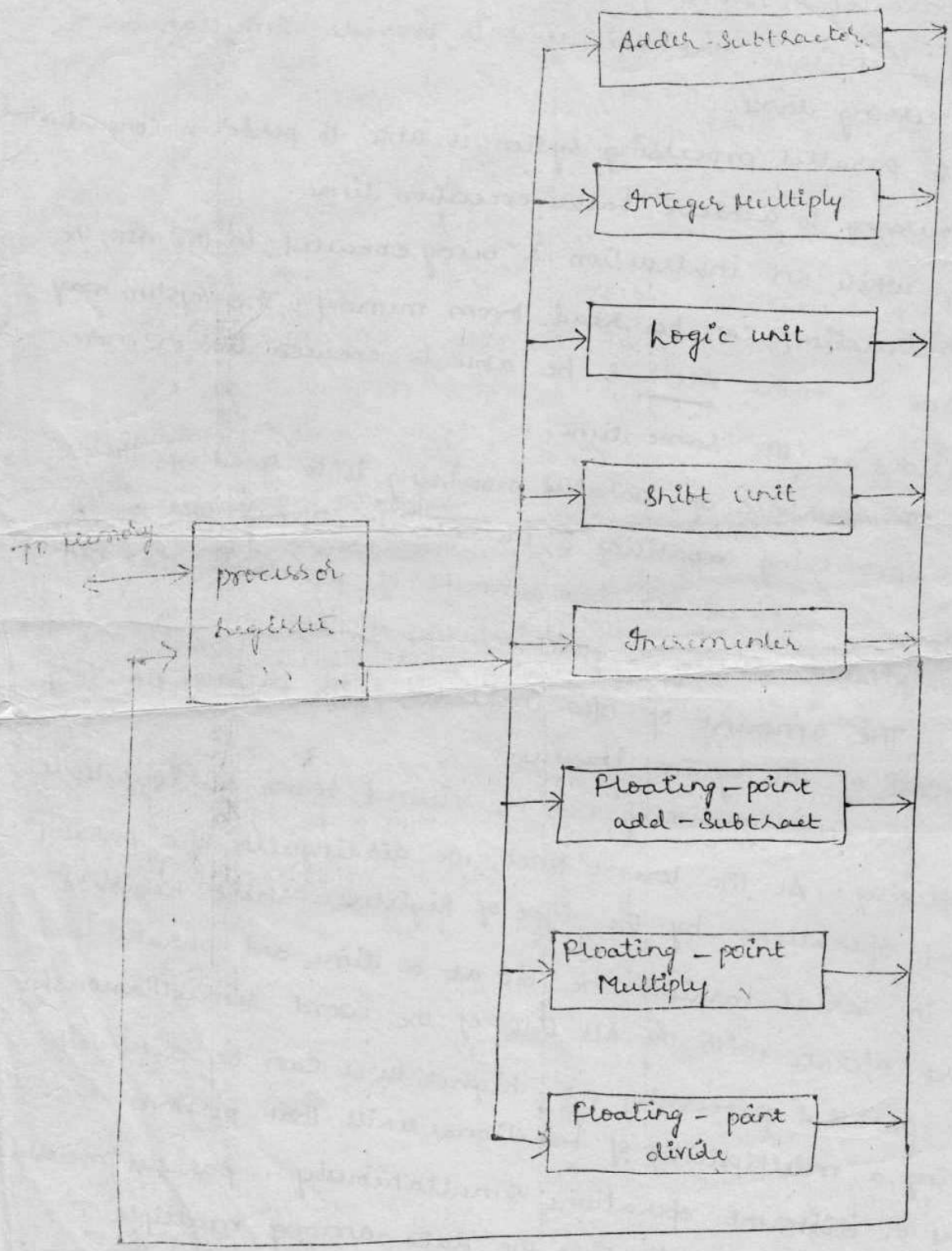
The fig. shows one possible way of separating

www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org

execution unit into eight functional units operating in



The operands in the registers are applied to one of the units depending on the operation specified by the instruction association with the operands.

www.jntuworldupdates.org

The adder & integer multiplier perform the arithmetic operations with integer numbers. The floating-point operations are separated into three circuits operating in parallel. The logic, shift & increment operations can be performed concurrently on different data. All units are independent of each other, so one number can be shifted while another number is being incremented. A multifunctional organization is usually associated with a complex control unit to coordinate all the activities among various components.

There are a variety of ways that parallel processing can be classified. It can be considered as internal organization of the processors, from the interconnection structure b/w processors or from the flow of information through the system.

One classification which introduced by M.J. Flynn, considers the organization of a computer systems by the no. of instructions & data items that are manipulated.

The sequence of instruction read from memory constitutes an instruction stream. The operations performed on the data in the processor constitutes a data stream.

Flynn's classification divides computers into four major groups as follows:

SISD (Single instruction stream, Single data stream)
SIMD (Single instruction stream, multiple data stream)
MISD (Multiple instruction stream, Single data stream)
MIMD (Multiple instruction stream, multiple data stream)

SISD represents the organization of a single computer containing a control unit, a processor unit & a memory unit

www.jntuworldupdates.org

Instructions are executed sequentially & the system may/may not have internal parallel processing

SIMD represents an organization that includes many processing units under the supervision of a common control unit. All processors receive the same instruction from the control unit but operate on different items of data. The shared memory units must contain multiple modules so that it can with all the processors simultaneously.

MISD structure is only of theoretical interest but there is no practical system.

MIMD organization refers to a computer system capable of processing several programs at the same time.

Pipelining :-

Pipelining is a technique of decomposing a sequential process into suboperations, with each subprocess being executed in a special dedicated segment that operates concurrently with all other segment.

Each segment performs partial processing in which the task is partitioned. The result obtained from the computation in each segment is transferred to the next segment in the pipeline. The final result is obtained after the data have passed through all segments.

Its characteristic of p is that several computations can be in progress in distinct segments at the same time. The overlapping is also possible, by associating a register with each segment in the pipeline.

The simplest way is to implement pipeline is to imagine that each segment consists of an i/p register followed by a combinational circuit. The register holds the data & the

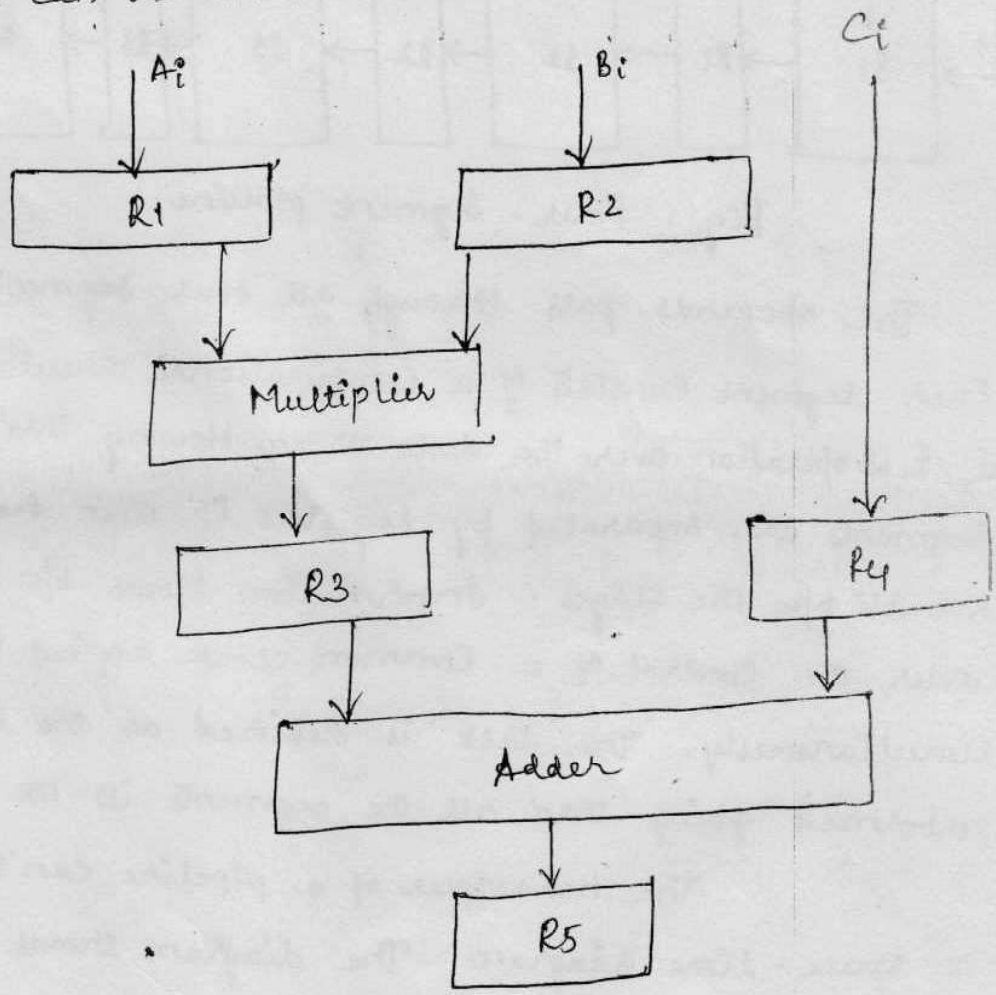
www.jntuworldupdates.org

in a given segment is applied to the i/p reg. of the next segment. A clock is applied to all registers. In this way, the information flows through the pipeline one step at a time

Eg: Suppose that want to perform the combined multiply & add operations with a stream of numbers.

$A_i * B_i + C_i$ for $i = 1$ to 7:

This can be shown below:



The registers R1 through R5 that receive new data with every clock pulse. The multiplier & adder are combinational circuits. The suboperations performed in each segment of the pipeline as follows:

$R1 \leftarrow A_i$, $R2 \leftarrow B_i$	A/p A_i & B_i
$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$	Multiply & i/p C_i
$R5 \leftarrow R3 + R4$	Add C_i to product

www.jntuworldupdates.org

~~The~~

General Considerations :-

The general structure of a four-segment pipeline is shown below.

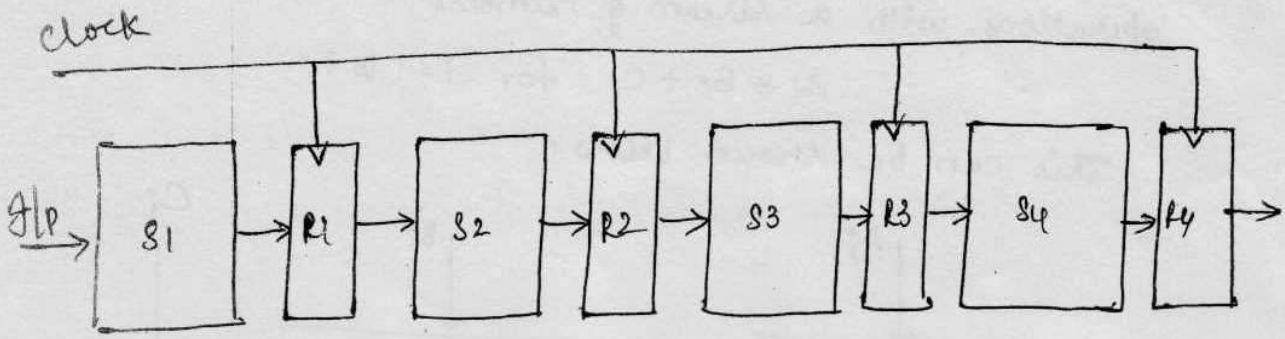


Fig : Four-segment pipeline.

The operands pass through all four segments in a fixed sequence. Each segment consists of a combinational circuit S_i that performs a suboperation over the data stream flowing thro' the pipe. The segments are separated by registers R_i that holds the intermediate results b/w the stages. Information flows b/w adjacent stages under the control of a common clock applied to all registers simultaneously. The 'task' is defined as the total operation performed going thro' all the segments in the pipeline.

The behaviour of a pipeline can be illustrated with a space-time diagram. The diagram shows the segment utilization as a function of time. This can be shown in below fig.

	1	2	3	4	5	6	7	8	9
Segment 1	T_1	T_2	T_3	T_4	T_5	T_6			
2		T_1	T_2	T_3	T_4	T_5	T_6		
3			T_1	T_2	T_3	T_4	T_5	T_6	

The diagram shows six tasks T_1 through T_6 executed in

www.jntuworldupdates.org

in four segments. Initially, task T_1 is handled by segment 1. After the first clock, segment 2 is busy with T_1, while segment 1 is busy with task T_2. Continuing this manner, the first task T_1 is completed after the fourth clock cycle. Once the pipeline is full, it takes only one clock period to obtain an o/p.

Now consider the case where a K-segment pipeline with a clock cycle time t_p is used to execute 'n' tasks. The first task T_1 requires a time equal to kt_p to complete its operation. The remaining (n-1) tasks emerge from the pipe at the rate of one task per clock cycle & they will be completed after a time equal to $(n-1)t_p$. Therefore, to complete 'n' tasks using a K-segment pipeline requires $k+(n-1)$ clock cycles.

Consider a Non pipeline unit that performs the same operation & takes a time equal to t_n to complete each task. The total time required for 'n' tasks is nt_n. The speedup of a pipeline processing over an equivalent non pipeline processing is defined by the ratio,

$$S = \frac{nt_n}{(k+n-1)t_p}$$

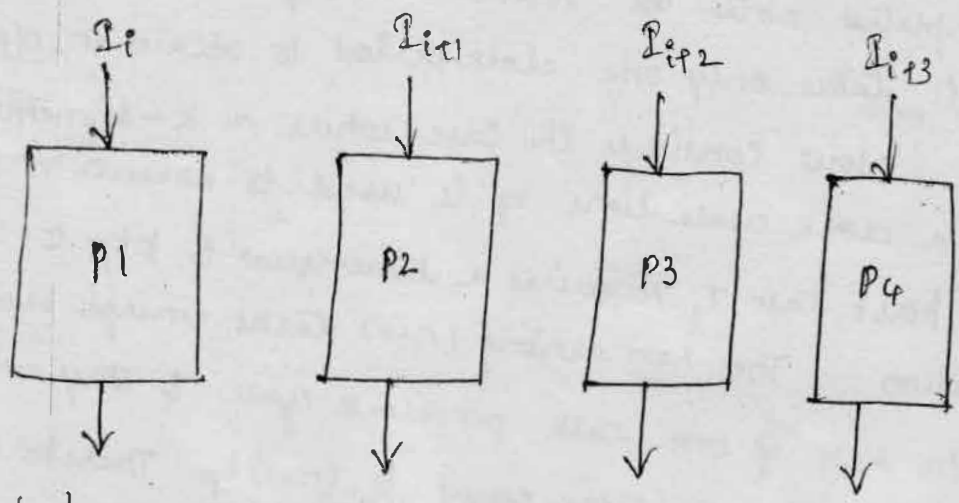
As the no. of tasks increases, n becomes much larger than K-1, & k+n-1 approaches the value of n. Under this condition, the speed up becomes

$$S = \frac{t_n}{t_p}$$

We assume that the time it takes to process a task is the same in the pipeline & non-pipeline circuits, we have $t_n = kt_p$. The speedup reduces to

$$S = \frac{kt_p}{t_p} = k.$$

The speed advantage of a pipeline process by means of multiple functional units, that will be operating in parallel. This is illustrated in below fig.



Each 'P' circuit performs the same task of an equivalent pipeline circuit. The parallel i/p's accepts the data simultaneously & perform four tasks at the same time.

The above fig. shows SIMD, in this also same instruction is used to operate on multiple data in parallel.

pipeline can't operate for many ~~reas~~ reasons. It has maximum theoretical rate. We aren't always going to be considered that the non-pipe circuit has the same time delay as that of pipeline circuit. pipeline provides a faster operation over a purely serial sequence.

~~Arithmatic pipelining:-~~

pipeline organization can be applicable in two areas 1) Arithmetic pipeline and 2) Instruction pipeline

An arithmetic pipeline divides an arithmetic operation into sub operations for execution in the pipeline segments

An instruction pipeline operates on a stream of instructions by overlapping the fetch, decode & execute phases of the instruction cycle.

www.specworld.in
www.smartzworld.com

www.jntuworldupdates.org

The pipeline arithmetic units are usually found in very high speed computers. They are used to implement floating-point operations, multiplication of fixed-point numbers.

The floating-point operations are easily decomposed into sub-operations.

The i/p's to the floating-point adder pipeline are two normalized floating-point binary numbers

$$X = A \times 2^a \quad , \quad Y = B \times 2^b.$$

A & B are two fractions that represent the mantissas & a & b are the exponents.

The floating-point addition & subtraction can be performed in four segments which is shown in fig.

The registers labeled 'R' are placed b/w the segments to store intermediate results. The suboperations that are performed in four segments are:

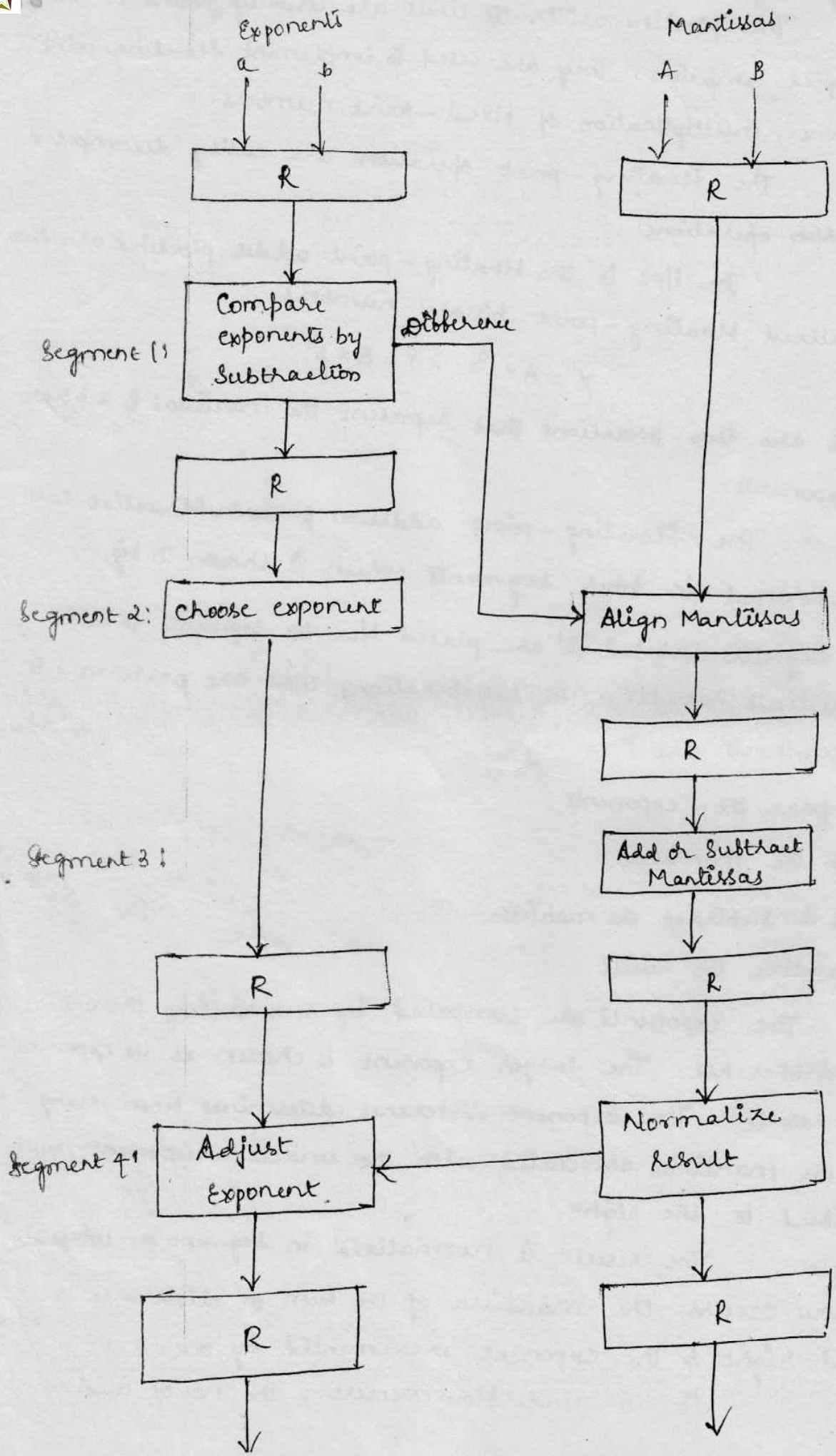
1) Compare the exponents
2) Align the mantissas
3) Add or subtract the mantissas
4) Normalize the result.

The exponents are compared by subtracting them to their difference. The larger exponent is chosen as the exponent of the result. The exponent difference determines how many times the mantissa associated with the smaller exponent must be shifted to the right.

The result is normalized in segment 4. When an overflow occurs, the mantissa of the sum or difference is shifted right & the exponent incremented by one.

If an underflow occurs, the no. of leading

www.jntuworldupdates.org



www.jntuworldupdates.org

mantissa & the number that must be subtracted from the exponent.

Eg: $X = 0.9504 \times 10^3$, $Y = 0.8200 \times 10^2$.

The two exponents are subtracted $3-2=1$. The larger exponent 3 is chosen as the exponent of the result. The next segment shifts the mantissa of Y to the right to obtain

$$X = 0.9504 \times 10^3, \quad Y = 0.0820 \times 10^3.$$

The exponents are same, then the addition of the two mantissas in segment 3 produces the sum

$$Z = 1.0324 \times 10^3.$$

The sum is adjusting by normalizing the result

$$Z = 0.10324 \times 10^4.$$

This is done by the shifting the mantissa once to the right & incrementing the exponent by one to obtain the normalized sum.

Instruction pipeline :-

An instruction pipeline reads consecutive locations from memory while previous instructions are being executed in other segments. This causes the instruction fetch & execute phases to overlap and perform simultaneous operation

Consider a computer with an instruction fetch unit & an instruction execution unit designed to provide a two segment pipeline. The instruction fetch segment can be implemented by means of a first-in, FIFO buffer. This unit uses queue. When ever the execution unit is not using memory, the control increments the pc & uses its address value to read consecutive

www.jntuworldupdates.org

instructions from memory. So, that the instructions are executed on the FIFO. Thus an instruction stream can be placed in a queue, waiting for decoding & processing by the execution segment. This means reduce the average access time to memory for reading instructions and it works in a efficient way. when ever there is space in the FIFO buffer, the control unit initiates the next instruction fetch phase. The buffer acts as a queue from which control then extracts the instruction for the execution unit.

Computers with complex instructions require other phases in addition to fetch & execute phases. The following steps are

1) Fetch the instruction from memory.
2) Decode the instruction
3) Calculate the effective address.
4) Fetch the operands from memory.
5) Execute the instruction.
6) Store the result

The instruction pipeline can prevent from difficulties at its maximum rate. Different segments may take different times to operate on the incoming information. Some segments are skipped for certain operations.

For eg, A register mode instruction doesn't need an effective address but remaining other segments may require it. The instruction must wait until another one is finished. So, Memory access conflicts are resolved by using two memory buses for accessing instructions & data in separate modules

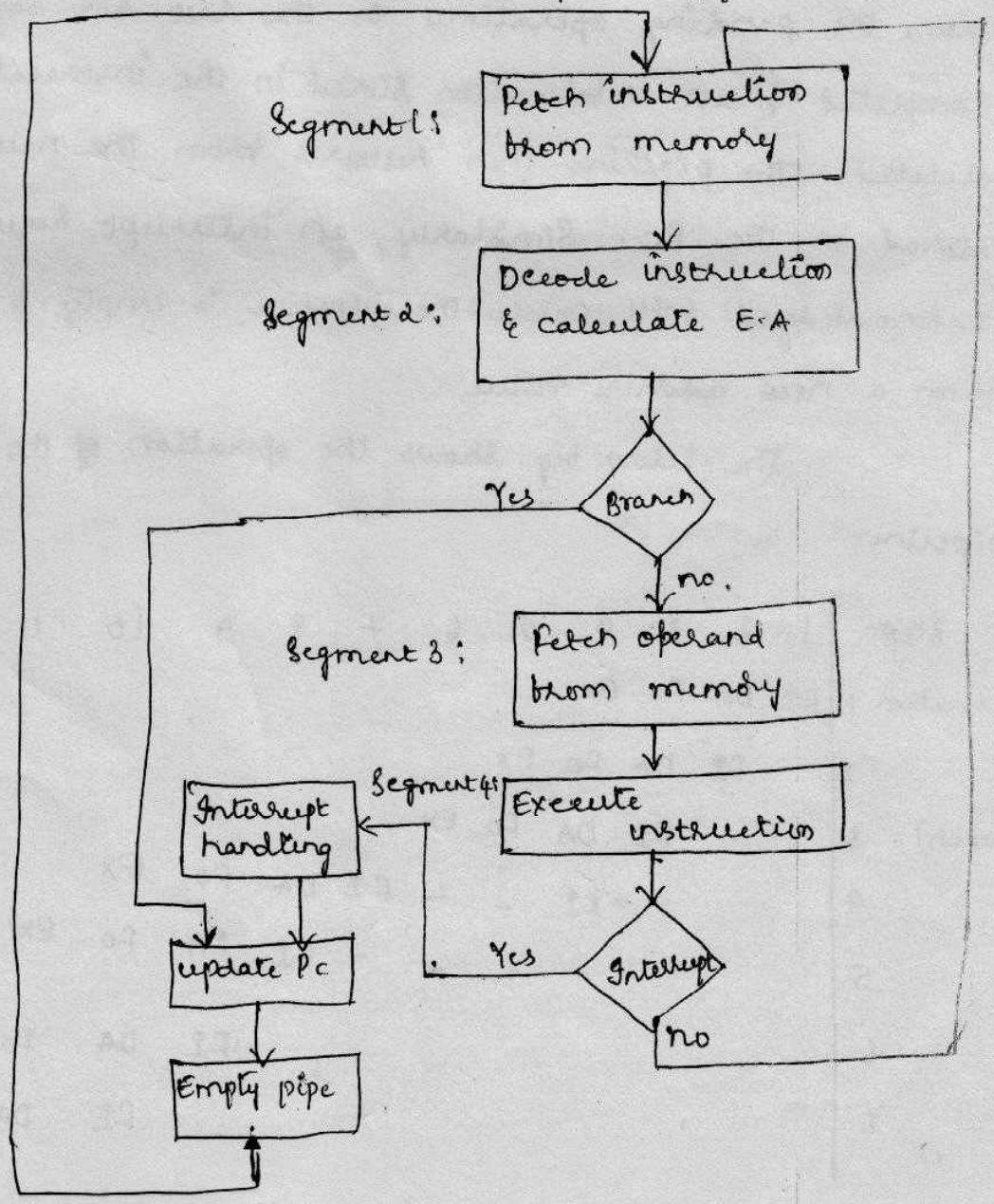
www.jntuworldupdates.org

segments of equal duration. The time that each step takes to fulfill its function depends on the instruction & the way it is executed.

Four-segment Instruction pipeline :-

Consider that the decoding of the instruction can be combined with the calculation of effective address into one segment. The instruction execution & storing of the result can be combined into one segment. This reduces the instruction pipeline into four segments.

The below fig. shows the instruction cycle in the CPU can be proceed with a four-segment pipeline.



www.smartzworld.com

www.jntuworldupdates.org

While an instruction is being executed in segment 4, the next instruction in sequence is busy fetching an operand from memory in segment 3. The effective address may be calculated in a separate arithmetic circuit for the third instruction, and whenever the memory is available, the fourth & all subsequent instructions can be fetched & placed in an instruction in FIFO. Four sub operations in the instruction cycle can overlap and upto four different instructions can be in progress of being processed at the same time.

An instruction in the sequence may be a program control type that causes a branch out of normal sequence. In that case, the pending operations in the last two segments are completed & all information stored in the instruction buffer is deleted. The pipeline then restarts from the new address stored in the PC. Similarly, an interrupt request, when acknowledged will cause the pipeline to empty & start again from a new address value.

The below fig. shows the operation of the instruction pipeline.

Step	1	2	3	4	5	6	7	8	9	10	11	12	13
Instruction 1	FI	DA	FO	EX									
2		FI	DA	FO	EX								
(Branch) 3			FI	DA	FO	EX							
4				FI	–	–	FI	DA	FO	EX			
5					–	–	–	FI	DA	FO	EX		
6									FI	DA	FO	EX	
7										FI	DA	FO	EX

www.jntuworldupdates.org

equal duration. The four segments are represented in the diagram with an abbreviated symbol.

1) FI is the segment that fetches an instruction
2) DA is the segment that decodes the instruction & calculates the effective address.
3) FO is the segment that fetches the operand
4) EX is the segment that Executes the instruction

Pipeline Conflicts :-

The three major difficulties that cause the instruction pipeline to deviate from its normal operation.

1) Resource Conflicts :-

It caused by access to memory by two segments at the same time. Most of these conflicts can resolved by using separate instruction & data memories.

2) Data dependency :-

This Conflicts arise when an instruction depends on the result of a previous instruction

3) Branch difficulties :-

This arise from branch & other instructions that change the value of PC.

Data dependency :-

This difficulty may caused a degradation of performance in an instruction pipeline is due to possible collision of data or address. A collision occurs when an instruction can't proceed because previous instructions didn't complete certain operations. A data dependency occurs when an instruction need data that not available.

Similarly, an address dependency may occur when an operand

www.jntuworldupdates.org

12 address can't be calculated

Hardware interlocks :-

This is a most straight forward Method. An interlock is a circuit that detects instructions whose source operands are destinations of instructions.

Detection of this situation causes the instruction whose source is n't available to be delayed by enough clock cycles to resolve the conflict. This approach maintains the program sequence by using h/w to insert the required delay.

operand forwarding :-

This uses a special h/w to detect a conflict and then avoid it by routing the data through special paths b/w pipeline segments.

For eg, Instead of transferring an ALU result into a destination register, the h/w checks the destination operand and if it is need as source in the next instruction, it passes the result directly into the ALU i/p, by passing the register file. This Method requires additional h/w paths through multiplexers as well as the circuit that detects the conflict.

Delayed load :-

The responsibility for solving data conflicts problems to the compiler that translates the high-level programming language into a machine language program. The compiler for such computer is designed to detect a data conflict & reorder the instructions as necessary to delay the loading of the conflicting data by inserting no-operations instructions. This Method is referred to as 'delayed load'.

www.jntuworldupdates.org

One of the major problems in an instruction pipeline is branching. A branch instruction can be conditional or unconditional. An unconditional branches always the alters the sequential program flow by loading the program counter with the target address.

In an conditional branch, the control selects the target instruction if the condition is satisfied or the next sequential instruction if the condition is n't satisfied.

- prefetch target instruction :-

This is one kind of handling conditional Branch. In this, if the branch condition is successful, the pipeline continues from the branch target instruction. An extension of this procedure is to continue fetching instruction from both places until the branch decision is made.

- Branch target ~~instruction~~ Buffer :- (BTB)

Each entry in the BTB consists of the address of a previously executed branch instruction & target instruction for that branch. It also stores the next two instructions after the branch target instruction. When the pipeline decodes a branch instruction, it searches the associative memory BTB for the address of the instruction. If it is in the BTB, the instruction is available directly & prefetch continues from the new path. If ~~the~~ it isn't in the BTB, the pipeline shifts to a new instruction stream & stores the target instruction in the BTB.

The advantage of this scheme is that branch instructions that have occured previously are readily available in the pipeline with out interruption.

www.jntuworldupdates.org

Loop buffer :-

A variation of BTB is loop buffer. When a program loop is detected in the program, it is stored in the loop buffer. The program loop can be executed directly without having to access memory until the loop mode is removed by the final branching out.

Branch prediction :-

A pipeline with branch instruction uses some additional logic to guess the outcome of a conditional branch instruction before it is executed. The pipeline then begins prefetching the instruction stream from the predicted path. A correct prediction eliminates the wasted time caused by branch penalties.

Delayed branch :-

In this procedure, the compiler detects the branch instructions & rearranges the machine language code sequence by inserting useful instructions that keep the pipeline operating without interruptions.

RISC pipeline :-

RISC has ability to use an instruction pipeline in an efficient manner. The simplicity of the instruction set can be utilized to implement an instruction pipeline using a small no. of sub operations with each being executed in one clock cycle. The instruction format is of fixed length, so, the decoding of the operation can occur at the same time as the register selection. All data manipulation instructions have register-to-register operations. All operands are in registers, there is no need for

www.specworld.in www.smartzworld.com

Calculating E.A. The instruction pipeline can be implemented with two or three segments. One segment fetches the instruction from program memory and the other segment executes the instruction in the ALU. A third segment may be used to store the result of the ALU operation in a destination register.

The data transfer instructions in RISC are limited to load & store instructions. These instructions use register indirect addressing. They usually need three or four stages in the pipeline. To prevent conflicts b/w a memory access to fetch an instruction & to load or store an operand, the RISC provide two separate buses with two memories.

1) For storing the instructions

2) For storing the data

The two memories can some time operate at the same speed as the CPU clock & are referred to as cache memories.

The Major advantages of RISC is, it has ability to execute instructions at the rate of one per clock cycle. The main goal is to achieve a single-cycle instruction execution.

The advantage of RISC over CISC is that RISC can achieve pipeline segments, requiring just one clock cycle, while uses many segments in the pipeline, with the longest segment requiring two or more clock cycles

The RISC support another characteristic, i.e., compiler that translates the HLL into Machine language program.

The following examples show how a compiler can optimize the machine language program to compensate for pipeline conflicts.

www.jntuworldupdates.org

Three segment Instruction pipeline :

The data Manipulation instructions operate on data in processor registers. The data transfer instructions are load and store instructions that use an effective address obtained from the addition of two registers or a register and a displacement constant provided in the instruction. The program control instructions use register values & a constant to evaluate the branch address, which is transferred to a register or the program counter PC.

Consider the h/w operation for such a computer. The control fetches the instruction from program memory into an instruction register. The instruction is decoded at the same time that the registers needed for the execution of the instruction are selected. The processor unit consists of a no. of registers & an ALU that performs the necessary arithmetic logic & shift operations. A data memory is used to load or store the data from a selected register in the register file. The instruction cycle can be divided into three suboperations & implemented in three segments.

I : Instruction fetch

A : ALU operation

E : Execute operation

The I segment fetches the instruction from program memory. The instruction is decoded & an ALU operation in the A segment. The ALU is used for three different functions depending on the decoded instruction. It performs an operation for a data manipulation instruction, it evaluates the effective address for a load or store instruction or it calculates the branch

www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org

address for a program control instruction. The E segment directs the output of the ALU to one of three destinations depending on the decoded instruction. It transfers the result of the ALU operation into a destination register in the register file, it transfers the effective address to a data memory for loading or storing, it transfers the branch address to the program counter.

Delayed Load :-

Consider the operations of the following four operations

1. LOAD : R1 ← M[address 1]
2. LOAD : R2 ← M[address 2]
3. ADD : R3 ← R1 + R2
4. STORE : M[address 3] ← R3.

If the three-segment pipeline proceeds with out interruptions, there will be a data conflict in instruction 3 because the operand in R2 is not available in the 'A' segment. This can be shown in the timing of the pipeline.

clock cycles	1	2	3	4	5	6
1. Load R1	I	A	E			
2. Load R2		I	A	E		
3. Add R1+R2			I	A	E	
4. Store R3				I	A	E

The E segment in clock cycle 4 is in a process of placing the memory data into R2. The A segment in clock cycle 4 is using the data from R2, but the value in R2 will not be the correct value because it hasn't been transferred

www.smartzworld.com

from memory. It is up to the compiler to make sure that the instruction following the load instruction uses the data fetched from memory. If the compiler can't find a useful instruction to put after the load, it inserts a no-operation instruction.

This is a type of instruction that is fetched from memory but has no operation, thus wasting a clock cycle. The concept of delaying the use of the data loaded from memory is referred to as delayed load.

The below shows the program with no-operation instruction inserted after the load to R2 instruction.

clock cycle	1	2	3	4	5	6	7
1. Load R1	I	A	E				
2. Load R2		I	A	E			
3. No-operation			I	A	E		
4. Add R1 + R2				I	A	E	
5. Store R3					I	A	E

The data is loaded into R2 in clock cycle 4. The add instruction uses the value of R2 in step 5. Thus the no-op instruction is used to advance one clock cycle inorder to compensate for the data conflict in the pipeline.

The advantage of delayed load approach is that the data dependency is taken care by the compiler rather than h/w. This results in a simpler h/w segment since the segment doesn't have to check if the content of the register being accessed is currently valid or not.

www.jntuworldupdates.org

The Method used in most RISC processors is to rely on the compiler to redefine the branches so that they effect at the proper time in the pipeline. This method is referred to as delayed branch.

The compiler for a processor that uses delayed branches is designed to analyse the instructions before & after the branch & rearrange the program sequence by inserting useful instructions in the delay steps.

An example of delayed branch is shown in fig. The program for this, consists of 5 instructions.

Load from memory to R1

Increment R2.

Add R3 to R4

Subtract R5 from R6.

Branch to address X.

clock cycles :	1	2	3	4	5	6	7	8	9	10
1. Load	I	A	E							
Increment		I	A	E						
Add.			I	A	E					
Subtract.				I	A	E				
Branch to X					I	A	E			
No - operation						I	A	E		
No - operation							I	A	E	
Instruction in X								I	A	E

Fig. (a) No - operation instructions.

www.smartzworld.com

www.jntuworldupdates.org

Clock cycles	1	2	3	4	5	6	7	8
Load	I	A	E					
Increment		I	A	E				
Branch to X			I	A	E			
Add				I	A	E		
Subtract					I	A	E	
Instruction in X						I	A	E

Fig.(b) Re-arranging the instructions

into

fast

www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org

MULTI PROCESSORS

Characteristics of Multiprocessors :-

A Multiprocessor system is an interconnection of two or more CPU's with memory & i/o equipment. The term "processor" in Multiprocessor can have either CPU or i/o processor.

Multiprocessors are classified as multiple instruction stream, multiple data stream (MIMD) systems.

There are some similarities b/w multiprocessor and multicomputer systems since both support concurrent operations.

Computers are interconnected with each other by means of communication lines to form a computer n/w. The n/w consists of several autonomous computers that may or may not communicate with each other.

A multiprocessor system is controlled by one O.S. that provides interaction b/w processors & all the components of the system cooperate in the solution of a problem.

The benefits derived from a multiprocessor organization is an improved system performance. The system drives its high performance from the fact that computations can proceed in parallel in one of two ways.

1) Multiple independent jobs can be made to operate in parallel.
2) A single job can be partitioned into multiple parallel tasks

Multiprocessors are classified by the way their memory is organized.

1) Tightly coupled Multiprocessor.
2) Loosely coupled Multiprocessor.

www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org

A Multiprocessor system with common shared memory is classified as a Shared Memory or tightly coupled multi processor.

In this, each provide a cache memory with each CPU. There is a global common memory that all CPU's can access. Information can be shared among the CPU's by placing it in the common global memory.

Another Alternative model of microprocessor is distributed Memory or loosely coupled system. Each PE has its own private local memory. The processors are tied together by a switching scheme designed to route information from one processor to another through a message-passing scheme.

The processors relay program & data to other processor in packets. A packet consists of an address, the data content & some error detection code. The packets are addressed to a specific processor or taken by the first available processor.

Loosely coupled systems are most efficient when the interaction b/w tasks is minimal, where as tightly coupled systems can tolerate a higher degree of interaction b/w task's

www.jntuworldupdates.org

Array processors :-

An array processor is a processor that performs computations on large arrays of data.

This array processor is divided into two types

1) Attached Array processor
2) SIMD Array processor.

An attached array processor is an auxiliary processor attached to a general-purpose computer. It is intended to improve the performance of the host computer in specific numerical computation tasks.

An SIMD array processor is a processor that has a single-instruction multiple-data organization. It manipulates vector instructions by means of multiple functional units.

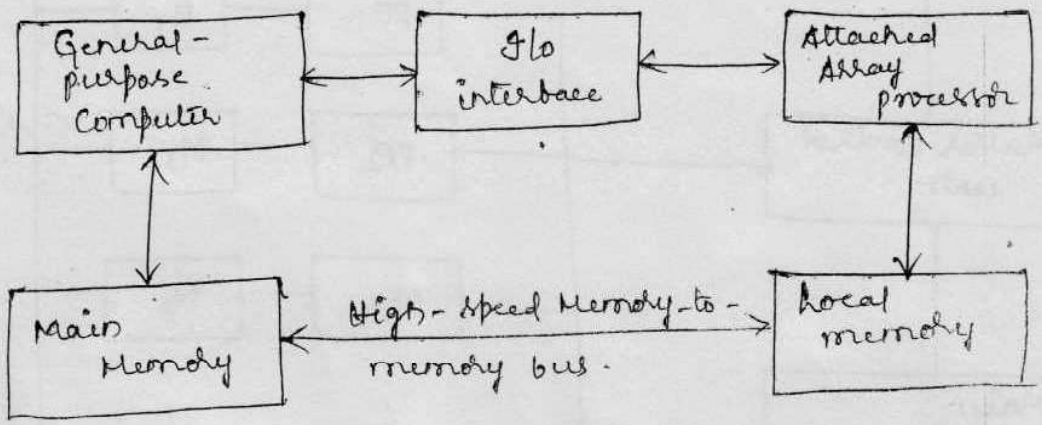
Attached Array processor :-

It is designed for a peripheral for a conventional host computer & its purpose to enhance the performance of the host computer by providing vector processing for complex scientific

www.jntuworldupdates.org

unit containing one or more pipelined floating-point adders & multipliers. The array processor can be programmed by the user to accommodate a variety of complex arithmetic problems.

The below fig. shows the interconnection of an attached array processor to a host computer.



The host computer is a general-purpose commercial computer & the attached processor is a back-end machine driven by the host computer.

The array processor is connected thro' an i/p & o/p controller of the computer & the computer treats it like an external interface. The data for the attached processor are transferred from main memory to a local memory through a high-speed bus. The general purpose computer without the attached processor serves the user that need conventional data processing. The system with the attached processor satisfies the needs for complex arithmetic applications.

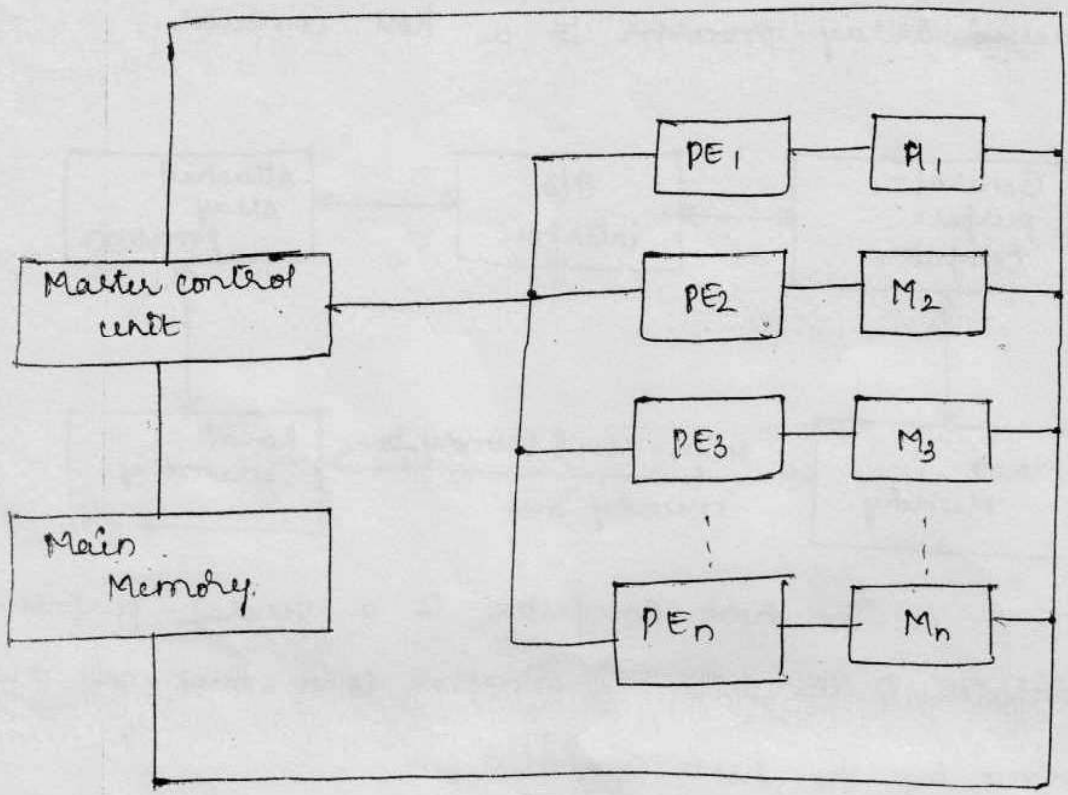
SIMD Array processor :-

An SIMD array processor is a computer with multiple processing units operating in parallel. The processing units are ~~synchronized~~ synchronized to perform the same operation under the control of

www.specworld.in www.smartzworld.com

a common control unit, thus providing a single instruction stream, multiple data stream (SIMD) organization.

A general block diagram of an array processor is shown in below fig.



The fig. consists of a set of identical processing elements (PE) each having a local Memory M. Each Processor element includes an ALU, a floating-point arithmetic unit & working registers. The master control unit controls the operations in the PE's. The main memory is used for storage of the program. The function of the master control unit ~~controls the operations in the processor elements~~ is to decode the instructions & determine how the instruction is to be executed. Scalar & program control instructions are directly executed with in the Master control unit. Vector instructions are broad cast to all PE's simultaneously. Each PE uses operands stored in its local memory. Vector operands are distributed to

www.jntuworldupdates.org

PE has a flag that is set when the PE is active & reset when the PE is inactive.

For eg, An array processor contains a set of 64 PE's. If a vector length of less than 64 data items is to be processed, the control units selects the proper number of PE's to be active. Vector of greater length than 64 must be divided into 64-word portions by the control unit.

Vector processing :-

In many scientific & engineering applications, the problems can be formulated in terms of vectors & matrices that lend themselves to vector processing.

Computers with vector processing capabilities are in demand in specialized applications.

The following are representative application areas where vector processing are used.

- Long-range weather forecasting
- petroleum explorations
- Seismic data analysis
- Medical diagnosis.
- Aerodynamics & Space flight simulations
- Artificial intelligence & expert systems
- Mapping the human genome
- Image processing.

To achieve the required level of high performance it is necessary to utilize the fastest & most reliable hw & apply innovative procedures from vector & parallel processing

www.smartzworld.com

www.jntuworldupdates.org

Vector operations :-

Many scientific problems require arithmetic operations on large arrays of numbers. These numbers are usually formulated as vectors & matrices of floating-point numbers.

A vector is an ordered set of a one-dimensional array of data items. A vector 'V' of length n is represented as a row vector by $V = [v_1 \; v_2 \; v_3 \; --- \; v_n]$. It may be represented as column vector. Consequently, operations on vectors must be broken down into single computations with subscripted variables. The element v_i of vector V is written as V(I) and the index I refers to a memory address or register where the number is stored.

Consider the following 'Do' loop:

```
DO    20 I = 1, 100
20        C[I] = B(I) + A(I).
```

This a program for adding two vectors A & B of length 100 to produce a vector C.

A possible instruction format for a vector instruction is shown below.

operation Code	Base address Source 1	Base address Source 2	Base address destination	vector length

Matrix Multiplication :-

Matrix Multiplication is one of the most computational intensive operations performed in computers with vector processors. The Multiplication of two n×n matrices consists of n^2 inner products or n^3 multiply-add operations. An n×m matrix of numbers has 'n' rows & 'm' columns and may be considered

www.jntuworldupdates.org

adder. During the next [illegible]

of the adder products entering the adder pipeline. At the end of the eighth cycle, the first products A_1B_1 through A_4B_4 are in the four adder segments, and the next four products A_5B_5 through A_8B_8 are in the multiplier segments. At the beginning of the ninth cycle, the o/p of the adder is A_1B_1 & the o/p of the multiplier is A_5B_5.

Thus the ninth cycle starts the addition $A_1B_1 + A_5B_5$ in the adder pipeline. The tenth cycle starts the addition $A_2B_2 + A_6B_6$ and so on.

This pattern is shown below.

$$
\begin{aligned}
C = {} & A_1B_1 + A_5B_5 + A_9B_9 + A_{13}B_{13} + \cdots \\
& + A_2B_2 + A_6B_6 + A_{10}B_{10} + A_{14}B_{14} + \cdots \\
& + A_3B_3 + A_7B_7 + A_{11}B_{11} + A_{15}B_{15} + \cdots \\
& + A_4B_4 + A_8B_8 + A_{12}B_{12} + A_{16}B_{16} + \cdots
\end{aligned}
$$

Memory Interleaving:-

Instead of using two memory buses for simultaneous access, the memory can be partitioned into a no. of modules connected to a common memory address & data buses. A memory module is a memory array together with its own address & data registers.

The below fig. shows a memory unit with four modules. Each memory array has its own address register AR & data register DR. The address registers receive information from a common address bus and the data registers communicate [illegible] bidirectional data bus.

www.specworld.in

www.jntuworldupdates.org

for eg, the multiplication of two 3x3 matrices A & B.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

The inner product obtain is,

$$c_{ij} = \sum_{k=1}^{3} a_{ik} \times b_{kj}$$

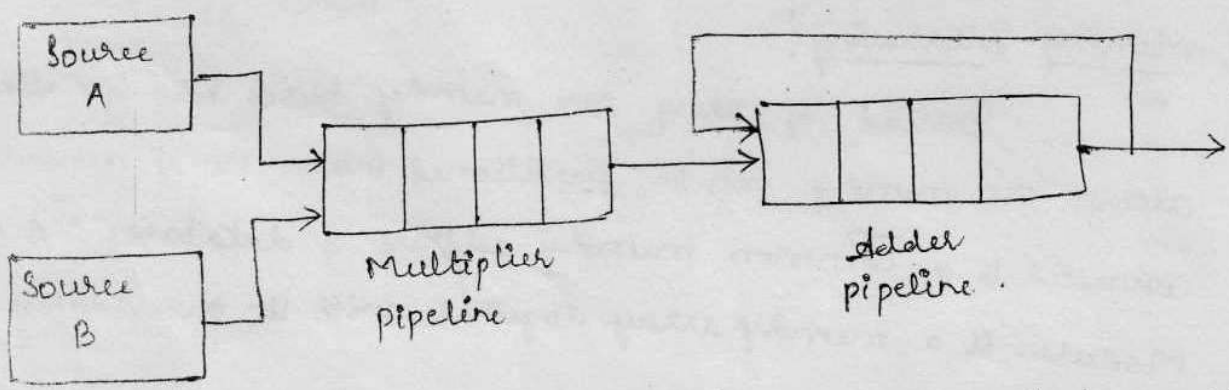
for eg, $i=1, j=1$

$$c_{11} = a_{11} b_{11} + a_{12} b_{21} + a_{13} b_{31}$$

In general, the inner product consists of the sum of 'k' product terms of the form.

$$C = A_1 B_1 + A_2 B_2 + \cdots + A_k B_k .$$

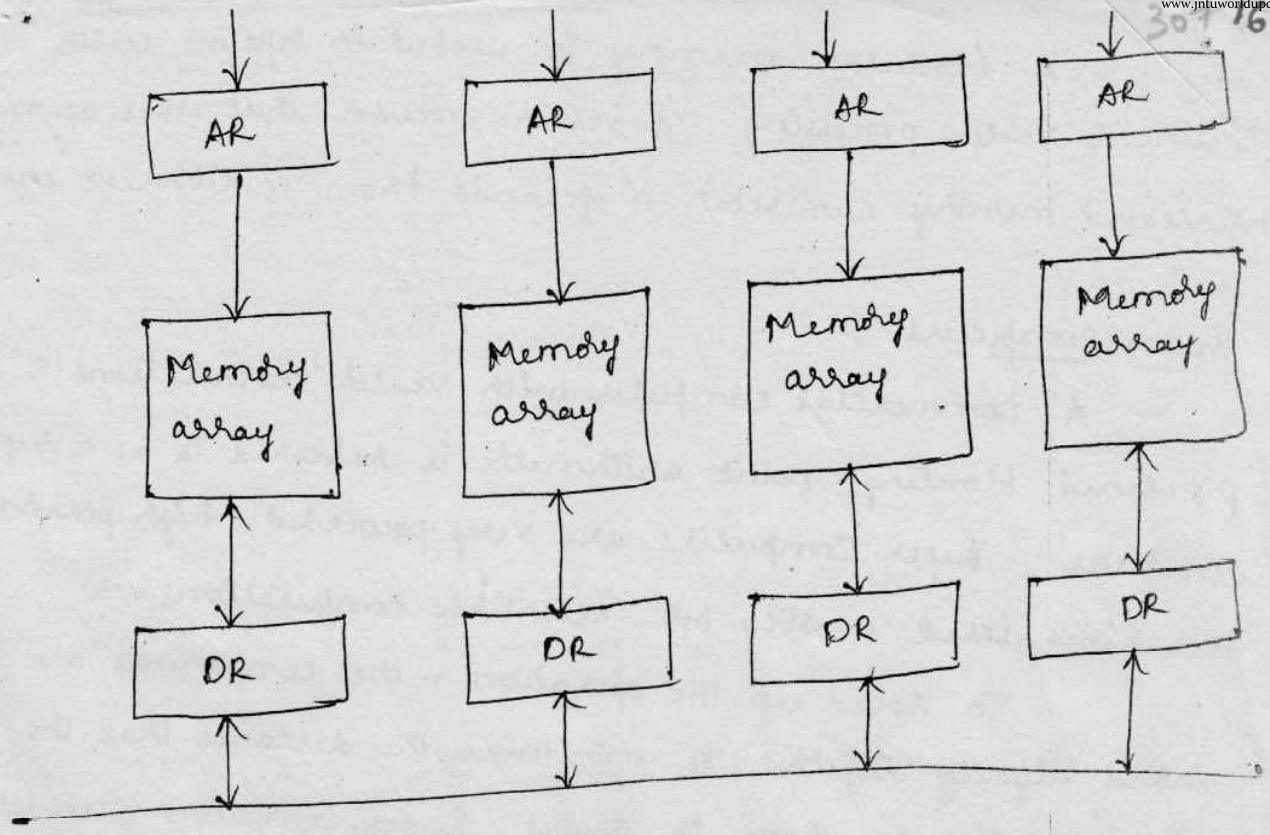
The inner product calculation on a pipeline vector processor is shown in fig.



The values of A & B are either in memory or in processor registers. The floating-point multiplier pipeline and the floating point adder pipeline are assumed to have four segments each. All segment registers in the multiplier & adder are initialized to zero. Therefore the o/p of the adder is 0 for the first eight cycle until both pipes are full. A_i & B_i pairs are brought in

... .. 10..

www.jntuworldupdates.org



The two LSB of the address can be used to distinguish between the four modules. The Modular system permits one module to initiate a memory access while other modules are in the process of reading or writing a word and each module can have a memory request independent of the state of the other modules.

The advantage of a modular memory is that it allows the use of a technique called 'interleaving'. In an interleaved memory, different sets of addresses are assigned to different Memory Modules.

For eg, In two-module memory system, the even addresses may be in one module & the odd addresses in other. When the no. of modules is of power 2, the LSB of the address select a memory module & the remaining bits designate the specific location to be accessed with in the selected Module.

www.jntuworldupdates.org

A Modular memory is useful in systems with pipeline & vector processing. A vector processor that uses an n-way interleaved memory can fetch 'n' operands from 'n' different modules.

Super Computers :-

A commercial computer with vector instructions & pipelined floating-point arithmetic is referred to as a 'super computer'. Super computers are very powerful, high performance machines used mostly for scientific computations.

To speed up the operation, the components are packed tightly together to minimize the distance that the electronic signals have to travel. Super computers also use special techniques for removing the heat from circuits to prevent them from burning up because of their closeness.

Super computers aren't suitable for normal everyday processing. They are limited in their use to a no of scientific applications etc. They have limited use & limited market because of their high price.

A measure used to evaluate computers in their ability to perform a given no. of floating-point operations per second is referred to as 'flops'. The term 'megaflops' is used to denote million flops & gigaflops to denote billion flops.

www.jntuworldupdates.org

Co

Can th

p

e model

des He or E has

p re

to from

p m n

The processors data other processor

Co an

Con s error

a specific processor or taken by the first

av or.

when

the unusual, we

C can tolerate

k

Interconnection Structures :-

There are several physical forms available for establishing an interconnection n/w.

1) Time - Shared common bus
2) Multiport Memory
3) cross bar switch
4) Multistage switching n/w.
5) Hypercube system.

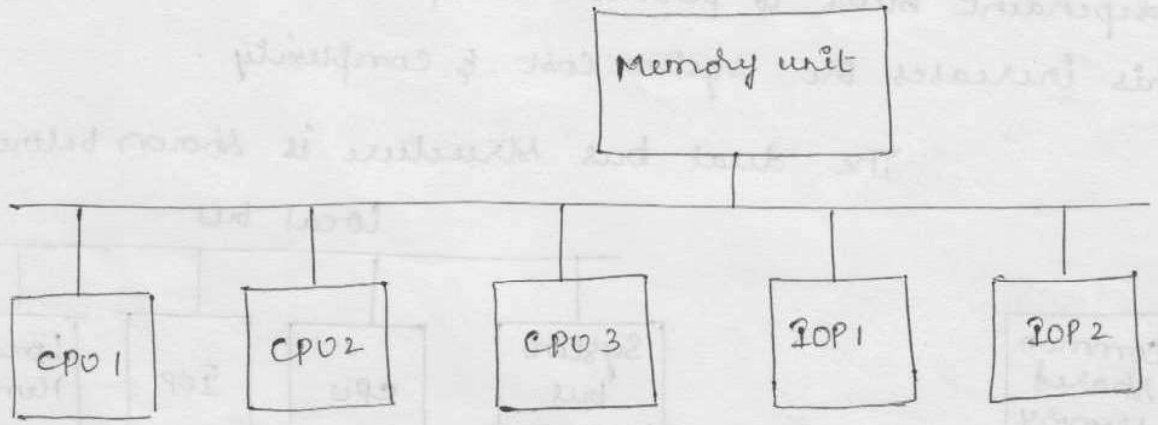
www.specworld.in

www.smartzworld.com

www.jntuworldupdates.org

Time Shared Common bus :-

A Common-bus multiprocessor system consists of a no. of processors connected through a common path to a memory unit. A time-shared common bus is shown below.



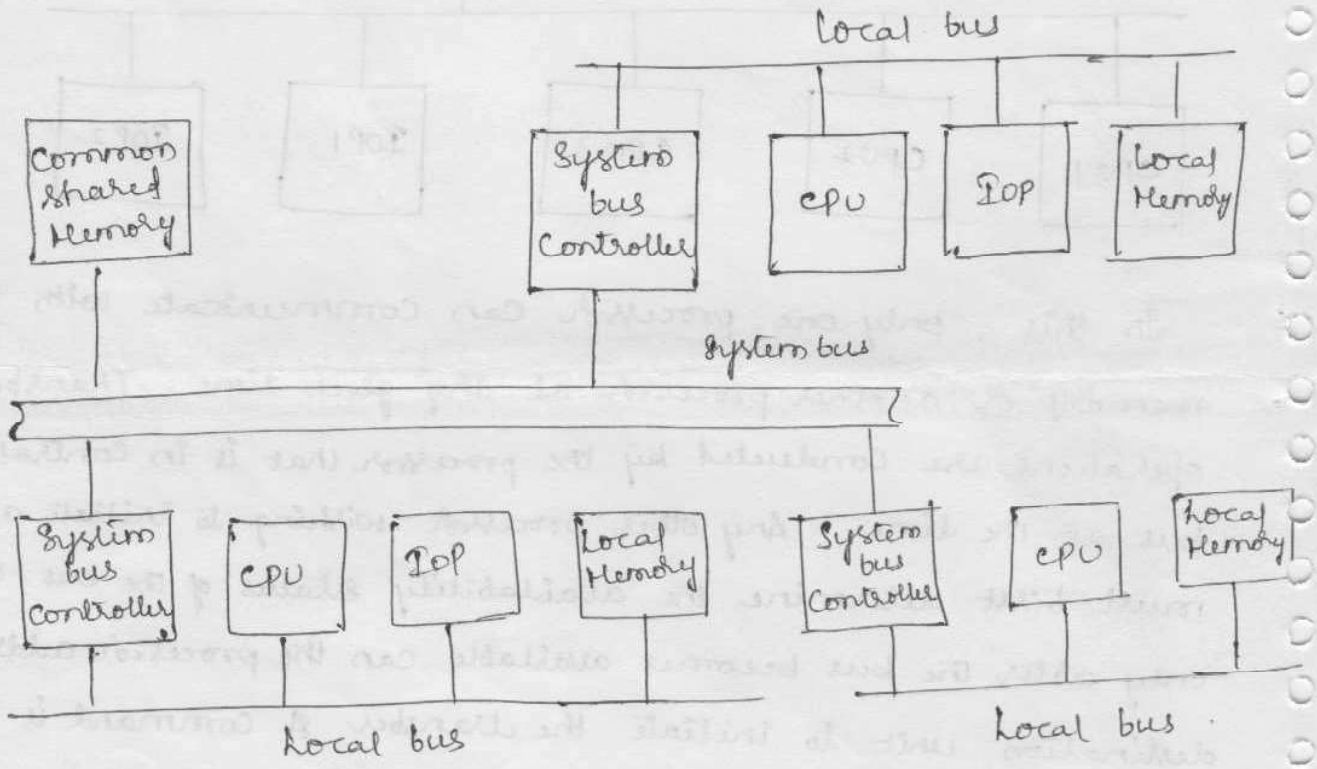
In this, only one processor can communicate with the memory or another processor at any given time. Transfer of operations are conducted by the processor that is in control of the bus at the time. Any other processor wishing to initiate a transfer must first determine the availability status of the bus and only after the bus becomes available can the processor address the destination unit to initiate the transfer. A command is issued to inform the destination unit what operation is to be performed. The receiving unit recognizes its address in the bus and responds to the control signals from the sender, after which the transfer is initiated. The system may exhibit transfer conflicts since one common bus is shared by all processors. These conflicts must be resolved by incorporating a bus controller that establishes priorities among the requesting units.

A single common-bus system is restricted to one transfer at a time. i.e., when one processor is communicating

www.jntuworldupdates.org

with the memory, all other processors are either busy with internal operations or must be idle waiting for bus. The total overall transfer rate with in the system is limited by the speed of the single path. The processors in the system can be kept busy, that the implementation of two or more independent buses to permit multiple simultaneous bus trans.. This increases the system cost & complexity.

The dual bus structure is shown below

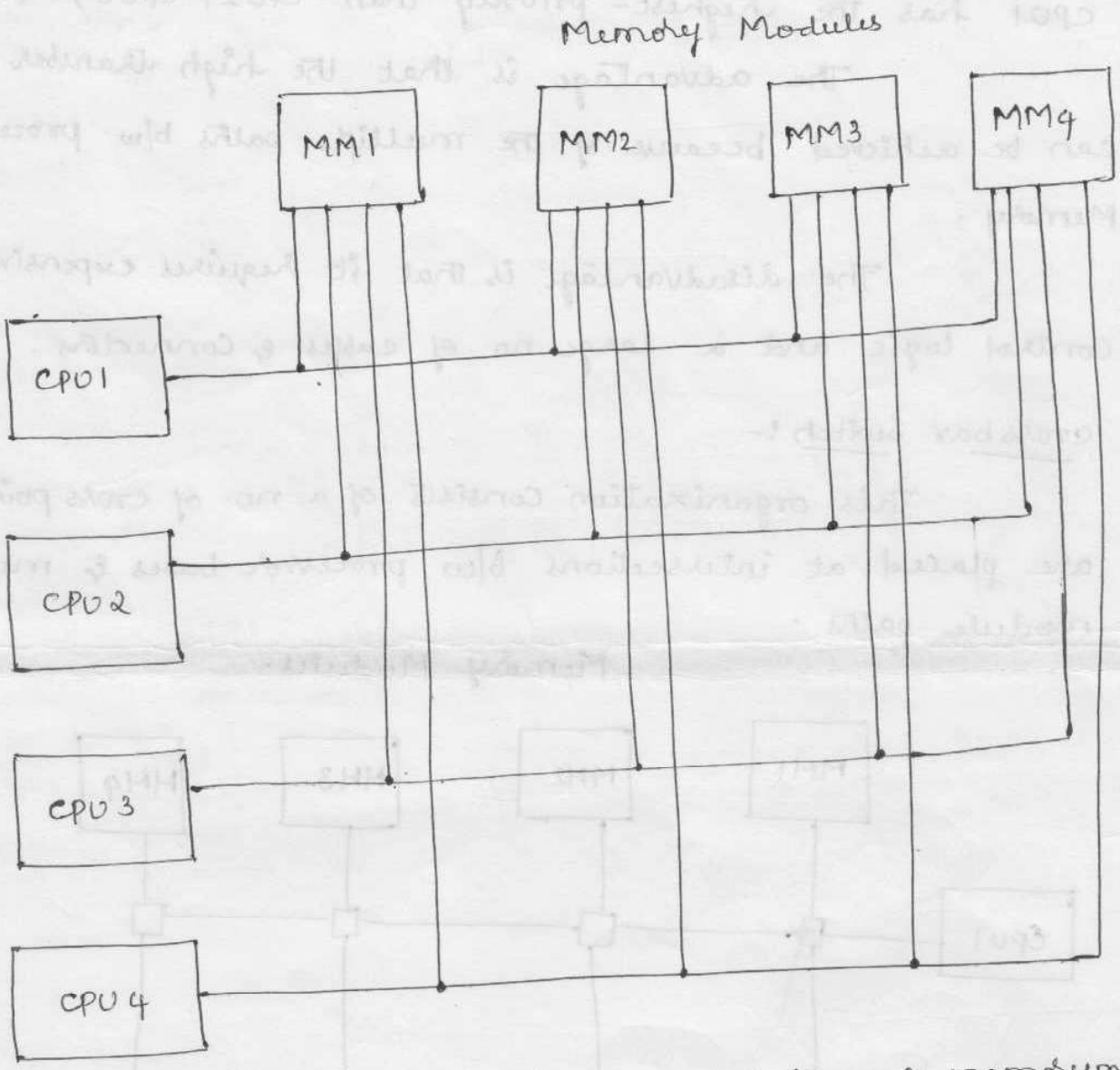


In this, their consists of local buses each connected to its own local memory & to one or more processors. Each local bus may be connected to CPU, an IOP, or any combination of processors. A system bus controller links each local bus to a common system bus. The memory connected to the common system bus is shared by all processors. part of the local memory may be designed as a cache memory attached to the CPU. In this way, the average access time of the local memory can be made to approach the cycle time of the CPU to which it is attached.

www.jntuworldupdates.org

Multiport Memory :-

A multiport memory system consists of separate buses b/w each memory module & each CPU. The below fig. consists for four CPU's & four memory modules (MM).



Each processor bus is connected to each memory module. A processor bus consists of the address, data and control lines required to communicate with memory. The MM's is said to have four ports & each port accommodates one of the buses.

The Module must have control logic to determine which port will have access to memory at any given time. Memory access conflicts are resolved by assigning fixed

www.jntuworldupdates.org

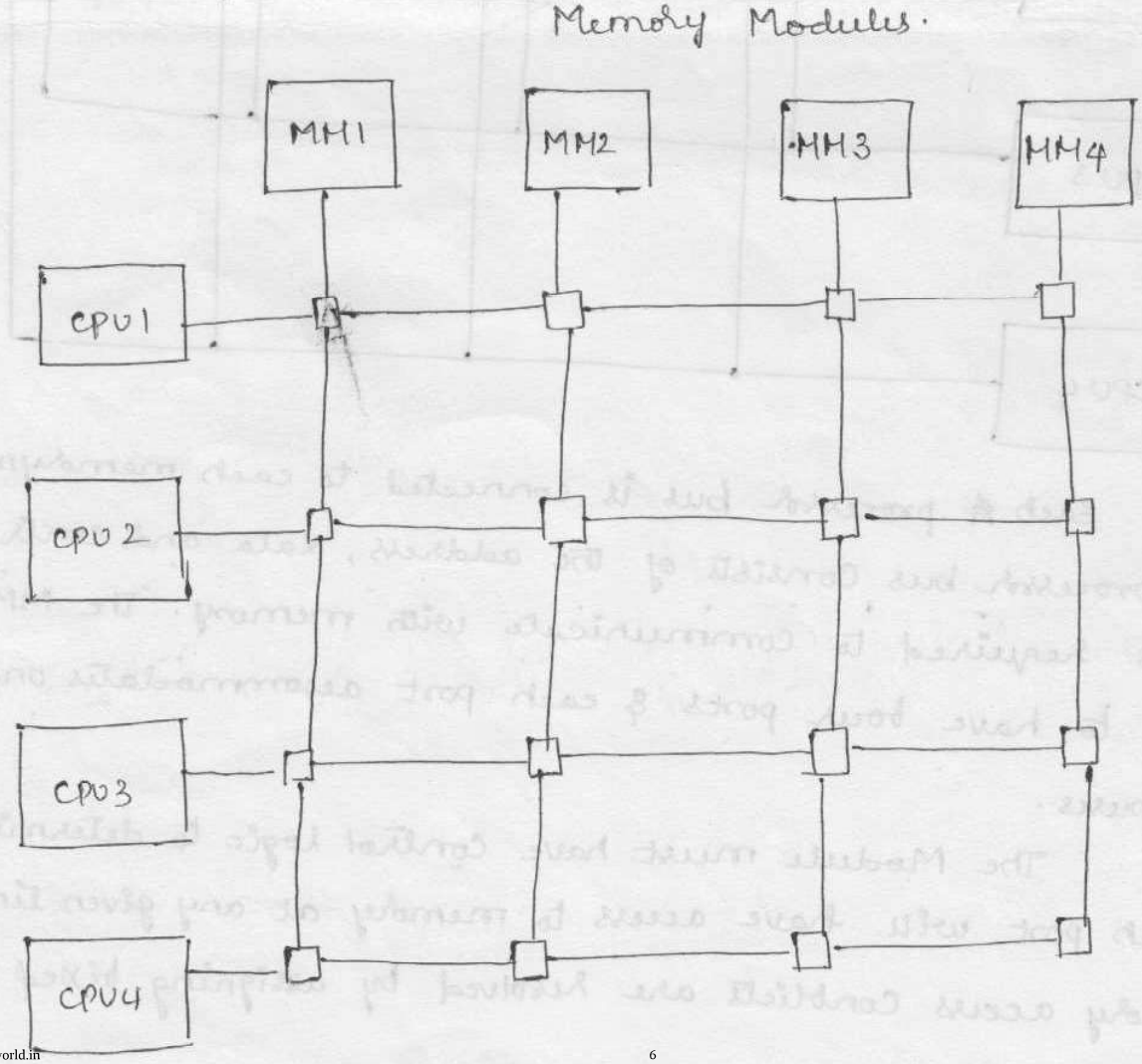
priorities to each memory port. The priority for memory access associated with each processor may be established by the physical port position that its bus occupies in each Module. The CPU1 has the highest-priority than CPU2, CPU3, CPU4.

The advantage is that the high transfer rate that can be achieved because of the multiple paths b/w processors and Memory.

The disadvantage is that it requires expensive memory control logic and a large no. of cables & Connectors.

Crossbar Switch :-

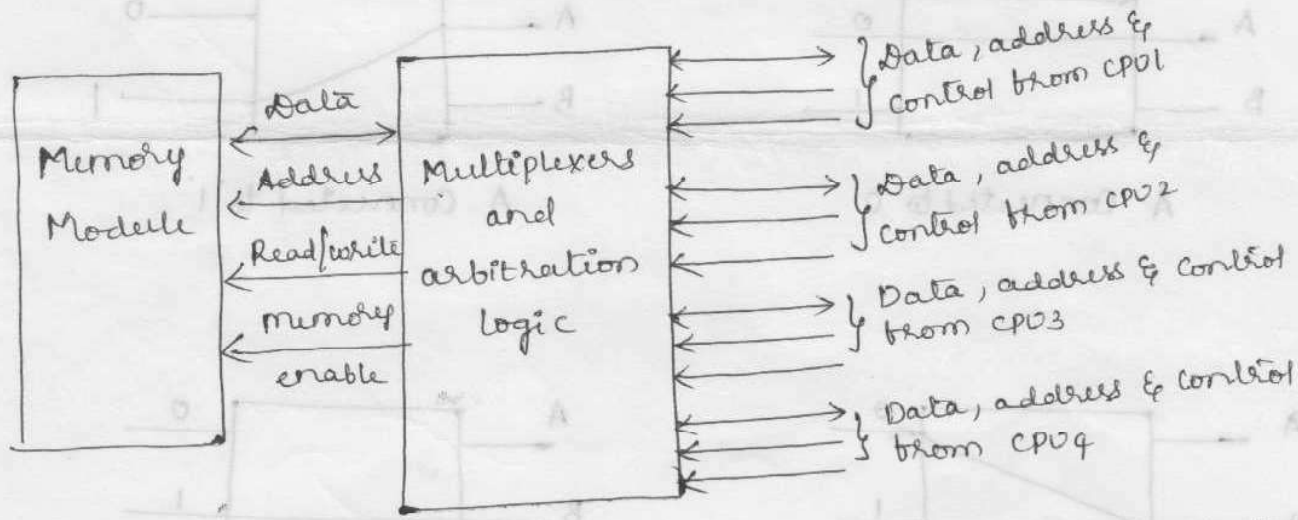
This organization consists of a no. of cross points that are placed at intersections b/w processor buses & memory module paths.



www.jntuworldupdates.org

The fig. shows a cross bar switch interconnection b/w four CPU's & four memory Modules. The small square in each cross point is a switch that determines the path from a processor to a memory module. Each point has control logic to set up the transfer path b/w a processor and memory. The address is placed in the bus to determine whether its particular module is being accessed. It also resolves multiple requests for access the same memory module on a priority basis.

The below fig. shows the functional design of a cross bar switch connected to one memory module.



The circuits consists of multiplexers that select the data, address & control from one CPU for communication with the memory module.

priority levels are established by the arbitration logic to select one CPU when two or more CPU's attempts to access the same memory. The multiplexers are controlled with the binary code that is generated by a priority encoder with in the arbitration logic.

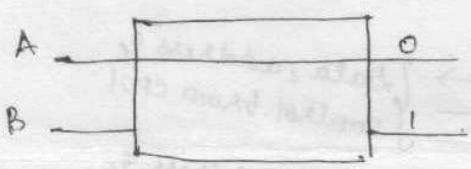
www.jntuworldupdates.org

A cross bar switch organization supports simultaneous transfers from all memory modules because there is a separate path associated with each module.

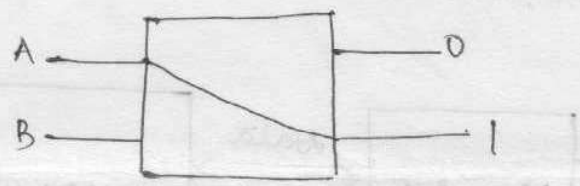
The h/w required to implement the switch can become quite large & complex.

Multistage Switching Network :-

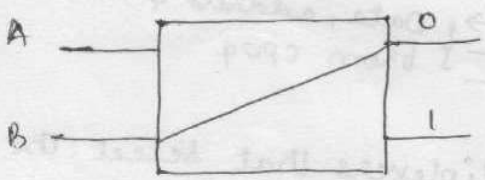
The basic component of a multistage n/w is a two-i/p, two-o/p interchange switch. The below fig. are the 2x2 switch has two-i/p's, labeled as A, B & two o/p labeled as 0 & 1. There are control signals associated with the switch that establish the interconnection b/w the i/p & o/p terminals.



A Connected to 0



A Connected to '1'



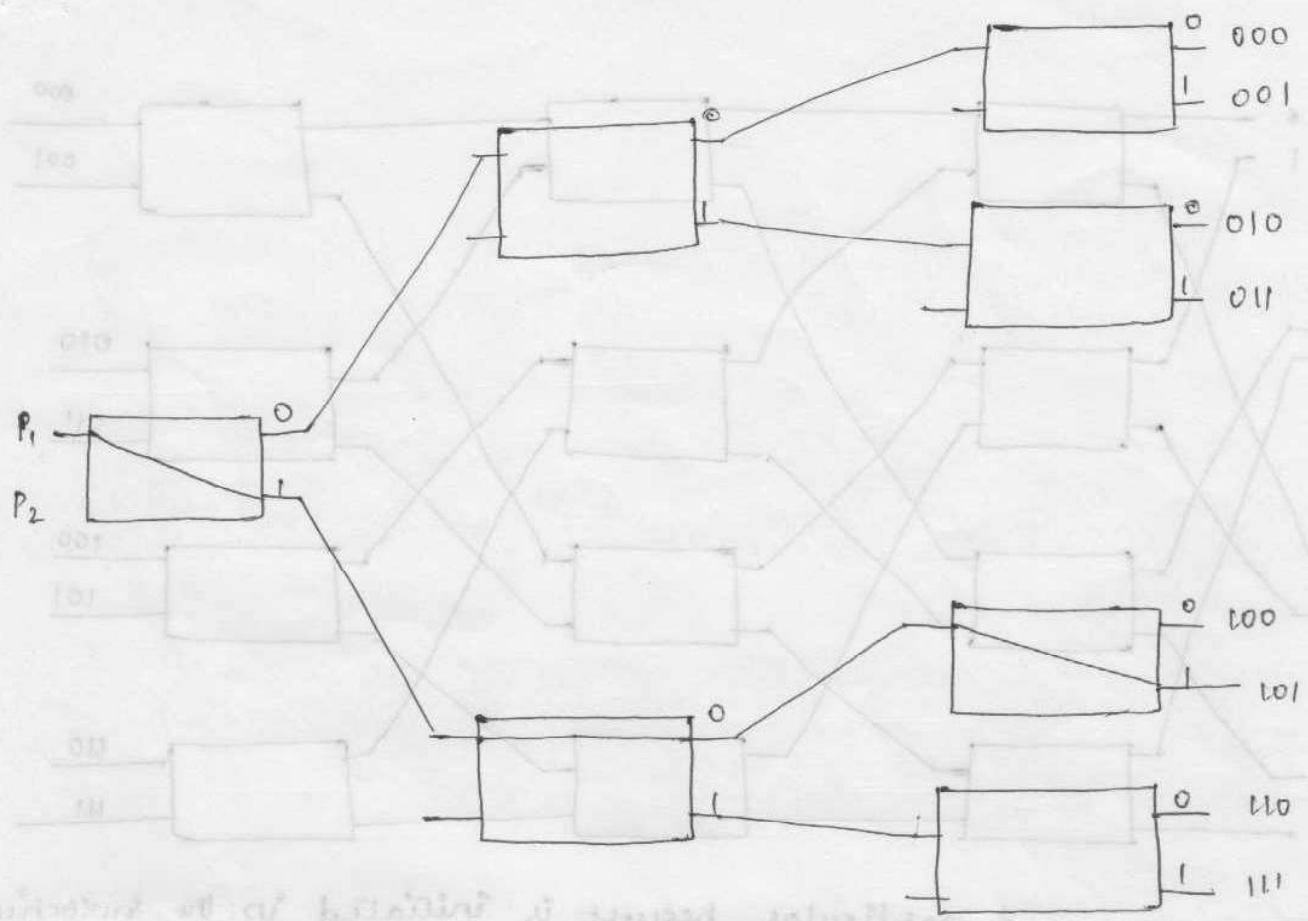
B Connected to 0

B Connected to '1'

The switch has the capability of connecting i/p's A & B to either of the o/p's. If i/p's A & B both request the same o/p terminal, only one of them will be connected, the other is blocked.

Using the 2x2 switch, we are going to build a multistage n/w to control the communication b/w a no. of sources & destinations. The construction of binary tree is shown below.

www.jntuworldupdates.org



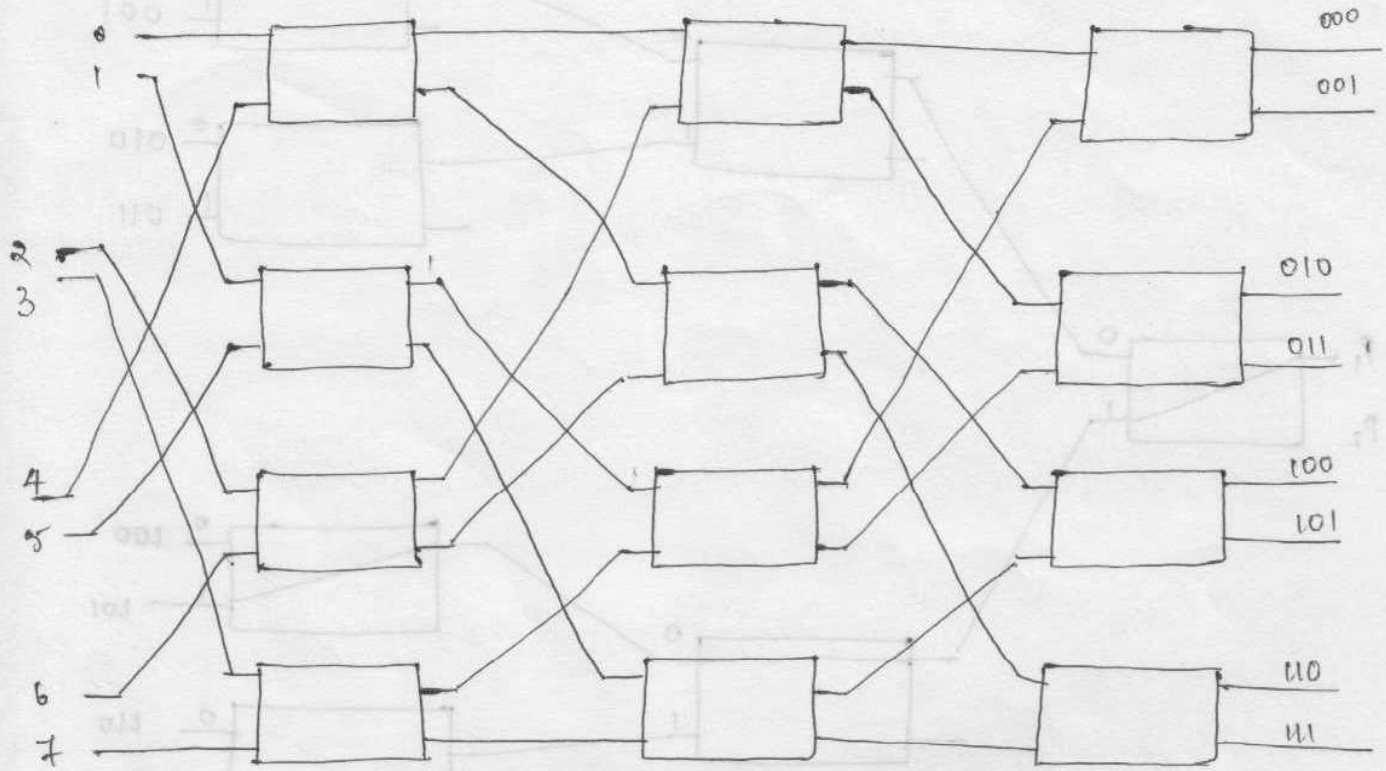
The two processors P_1 & P_2 are connected through switches to eight memory modules marked in binary from 000 through 111. The path from a source to a destination is determined from the binary bits of the destination number.

The first bit of the destination number determines the switch o/p in the first level. The second bit specifies the o/p in the second level. The third bit specifies the o/p in the third level.

Many different topologies have been proposed in the multistage switching n/w's to control processor - memory communication in a tightly - coupled multiprocessor system or to control the comm^n b/w the PE's in a loosely - coupled system. One such topology is the omega switching n/w which is shown below. In this configuration there is exactly one path from each source to any particular destination. Request patterns can't be connected simultaneously. For eg, any two sources can't be connected simultaneously to destinations 000 & 001.

www.specworld.in
www.smartzworld.com

318



A particular request is initiated in the switching n/w by the source, which sends a 3-bit pattern representing the destination number. As the binary pattern moves through the n/w, each level examines a different bit to determine the 2x2 switch setting. Level 1 inspects the MSB, Level 2 inspects the middle bit, level 3 inspects the LSB. When the request arrives on either i/p of the 2x2 switch, it is routed to the upper o/p if the specified bit is '0' or to the lower o/p if the bit is 1.

In a tightly-coupled system, the source is a processor & the destination is a memory module. The first pass through the n/w sets up the path. Succeeding passes are used to transfer the address into memory & then transfer the data in either direction, depending on whether the request is a read/write.

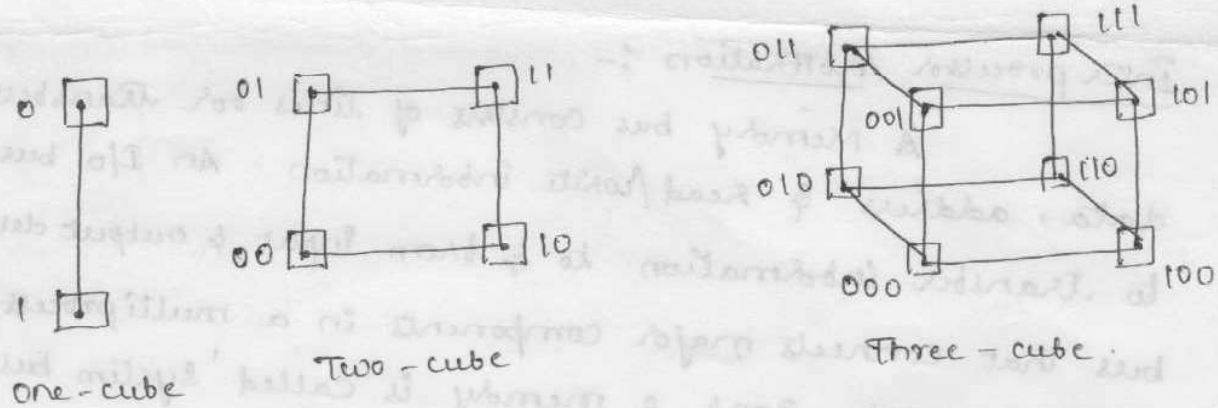
In a loosely coupled system, both the source & destination are PEs. After the path is established, the source processor transfer a message to the destination processor.

www.jntuworldupdates.org

Hyper cube Inter connection :-

The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of $N = 2^n$ processors inter-connected in an n-dimensional binary cube. Each processor forms a 'node' of the cube. Each ~~processor~~ node as processor but also local memory & I/O interface. Each processor has direct communication paths to 'n' other neighbour processors. These paths correspond to the edges of the cube. There are 2^n distinct n-bit binary addresses that can be assigned to the processors. Each processor address differs from that of each of its 'n' neighbours by exactly one bit position.

The below fig. shows the hypercube structure for n = 1, 2, 3.



One-cube Two-cube Three-cube

A one-cube structure has $n = 1$ & $2^n = 2$. It contains two processors interconnected by a single path. A two-cube structure has $n = 2$ & $2^2 = 4$. It contains four nodes interconnected as a square. A three-cube structure has eight nodes interconnected as a cube. A n-cube structure has 2^n nodes with a processor residing in each node. Each node is assigned a binary address in such a way that the addresses of two neighbours differ in exactly one bit position.

www.jntuworldupdates.org

Routing messages through an n-cube structure may take from one to 'n' links from a source node to a destination node. For eg, in a three-cube structure, node 000 can communicate directly with node 001. But it communicates with 011, it takes two links.

A routing procedure can be developed by computing the EX-OR of the source node address with the destination node address. The resulting binary value will have 1 bits corresponding to the axes on which the two nodes differ. The message is then sent along any one of the axes.

For eg, a message at 010 going to 001 produces an ex-or of the two addresses equal to 011. The message can be sent along the second axis to 000 & then through the third axis to 001.

Inter processor Arbitration :-

A Memory bus consists of lines for transferring the data, address & read/write information. An I/O bus is used to transfer information to & from input & output devices. A bus that connects major components in a multiprocessor system, such as CPU's, IOP's & memory is called 'System bus'.

The processors may request the system bus at the same time. Arbitration must then be performed to resolve this multiple contention for the shared resources. The arbitration logic would be part of the system bus controller placed b/w the local bus & the system bus.

System Bus :-

A typical system bus consists of approximately 100 signal lines. These lines are divided into three functional units i.e., data, address & control

www.jntuworldupdates.org

In addition to these there are power distribution lines that supply power to the components.

The datalines provide a path for the transfer of data b/w processors & common memory. The number of datalines is usually a multiple of 8 i.e., 16 & 32

The address lines are used to identify a memory address or any other source or destination such as i/p or o/p ports.

Data transfers over the system bus are divided into two types 1) Synchronous bus
2) Asynchronous bus.

In synchronous bus, each data item is transferred during a time slice known to both source & destination units. Synchronization is achieved by driving both units from a common clock source. An alternative procedure is to have separate clocks of approximately the same frequency in each unit. The synchronization signals are transmitted periodically.

In Asynchronous bus, each data item being transferred is accompanied by handshaking control signals to indicate when the data are transferred from the source & received by the destination.

The control lines provide signals for controlling the information b/w units. The timing signals indicate the validity of data & address information. The command signals specify operations to be performed.

Some of the control signals are memory read & write, acknowledge of a transfer, interrupt requests, bus control signals such as bus request & bus grant & signals for arbitration procedures.

www.jntuworldupdates.org

If the processor requests control of the bus & the corresponding arbiter finds its PI i/p equal to 1, it sets its PO o/p to '0'. Lower-priority arbiters receive a 0 in PI & generate a '0' in PO. Thus the processor whose arbiter has a PI=1 & PO=0 is the one that is given control of the system bus.

A processor may be in the middle of a bus operation when a higher-priority processor requests the bus. The lower-priority processor must complete its bus operation before it relinquishes control of the bus.

The bus busy line provides a mechanism for an orderly transfer of control. The busy line comes from open-collector circuits in each unit & provides a wired-OR logic connection. When an arbiter receives control of the bus it examines the busy line. If the line is inactive, it means that no other processor is using the bus. The arbiter activates the busy line & its processor takes control of the bus.

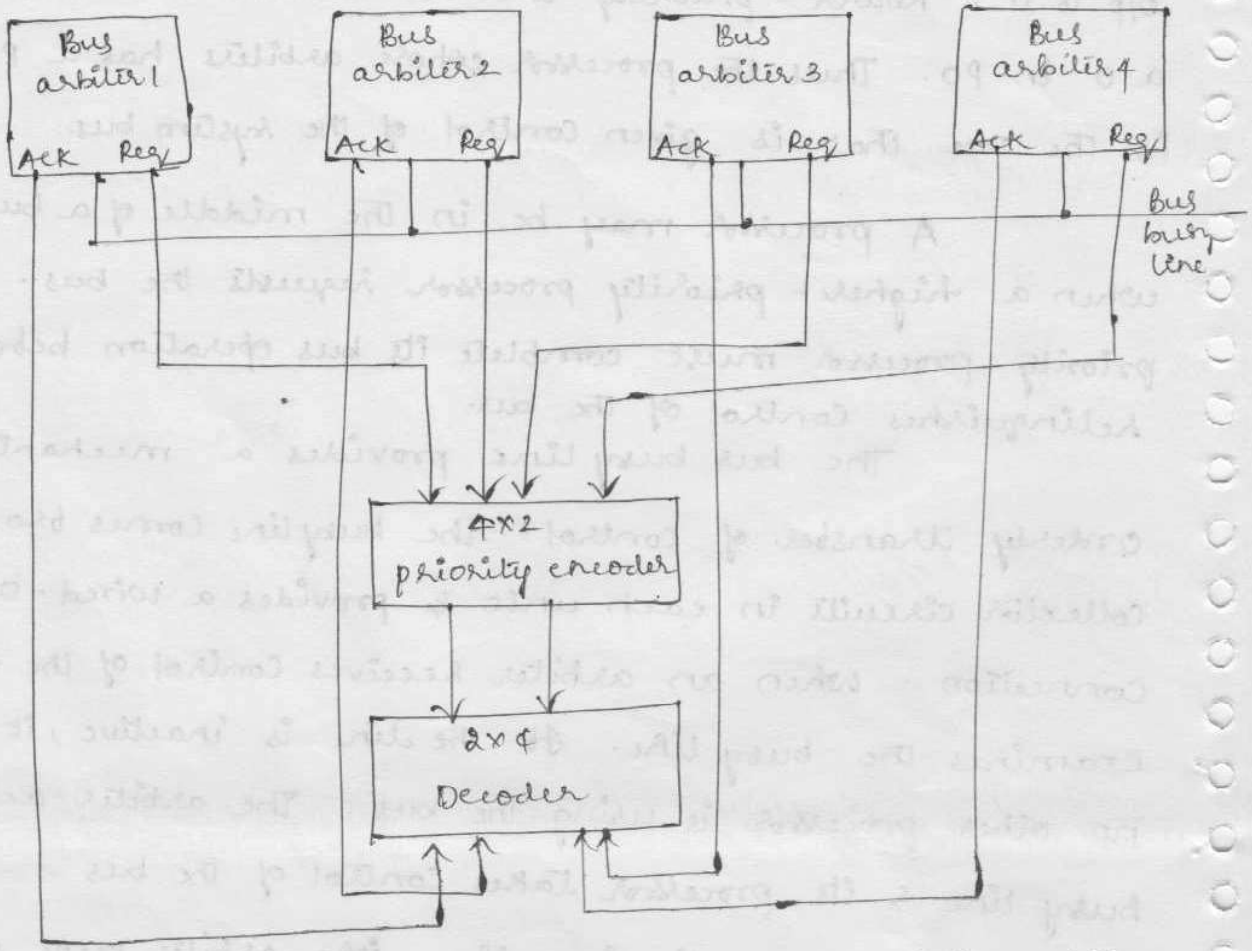
If the line is active, the arbiter keeps examining the busy line while the lower-priority processor that lost control of the bus completes its operation. When the bus busy line returns to its inactive state, the higher-priority arbiter enables the busy line & then required ~~bus~~ data transfers

<u>Parallel Arbitration logic</u> :-

This technique uses an external priority encoder & a decoder can be shown in below fig.

Each bus arbiter in the parallel scheme has a bus request o/p line & a bus acknowledge i/p line. Each arbiter enables the request line when its processor is requesting access to the system bus. The processor takes control of the bus if its acknowledge i/p line is enabled. The bus busy line provides an

orderly transfer of control.



The request lines from four arbiters going into a 4x2 priority encoder. The o/p of the encoder generates a 2-bit code which represents the highest-priority unit among those requesting bus.

The 2-bit code from the encoder o/p drives a 2x4 decoder which enables the proper acknowledge line to grant bus access to the highest-priority unit.

The bus priority-in (BPRN) & bus priority-out (BPRO) are used for daisy-chain connection.

The signals used to construct a parallel arbitration procedure are bus request (BREQ) & priority-in (BPRN). The bus clock (BCLK) is used to synchronize all bus transactions.

www.jntuworldupdates.org

The above algorithms are static priority algorithm because the priority of each device is fixed. (9)

Dynamic Arbitration Algorithms :-

This algorithm gives the system the capability for changing the priority of the devices while the system is in operation

The time slice algorithm allocates a fixed-length time slice of that is offered sequentially to each processor in round-robin fashion.

In a bus system that uses polling, the bus grant signal is replaced by a set of lines called poll lines which are connected to all units. These lines are used by the bus-controller to define an address for each device connected to the bus. The polling sequence is programmable & the selection priority can be altered.

The LRU algorithm gives the highest priority to the requesting device that hasn't used the bus for the longest interval. The priorities are dynamically changed to give every device an opportunity to access the bus.

In first-come, first-serve scheme, requests are served in the order received. To implement this algorithm, the bus controller establishes a queue arranged according to the time that the bus requests arrive. Each processor must wait for its turn to use the bus on a first-in, first-out (FIFO) basis.

The rotating daisy-chain procedure is a dynamic extension of the daisy-chain algorithm. In this scheme, there is no central-bus controller & the priority-line is connected from the priority-out of the last device back to the priority-in of the first device in a closed loop.

www.jntuworldupdates.org

Interprocessor Communication & Synchronization :

The various processors in a multiprocessor system must be provided with a facility for communicating with each other. A communication path can be established through common I/O channels. In shared memory, is to set aside a portion of memory that is accessible to all processors. The primary use of the common memory is to act as a message center similar to a mail box, where each processor can leave messages for other processors & pick up messages intended for it.

The sending processor structures a request, a message, or a procedure & places it in the memory mailbox. Status bits residing in common memory are generally used to indicate the condition of the mail box, whether there is ~~valid~~ meaningful messages or not.

The receiving processor can check the mail box, to determine if there are valid messages for it. The response time of this procedure can be time consuming.

An efficient procedure is for the sending processor to alert the receiving processor directly by means of an interrupt signal. This can be ~~also~~ used through a s/w-initiated interprocessor interrupt by means of an instruction in the program of one processor which when executed produces an external interrupt condition in a second processor.

In addition to shared Memory, a multiprocessor system may have other shared resources. For eg, a magnetic disk storage unit connected to an IOP may be available to all CPU's. This provides a facility for sharing of systems. A communication path is also established.

To prevent conflicting use of shared resources by several processors there must be providing resources to each processor

www.jntuworldupdates.org

This task is given to the O.S.

There are three different organizations, for design of O.S for multiprocessors:

1) Master - slave configuration
2) Separate operating system
3) Distributed operating system

In a Master - slave mode, one processor is treated as Master which performs all O.S. functions. The remaining processors denoted as slaves which not perform O.S. functions. If slave processor needs an O.S service, it must request it by interrupting the master & waiting until the current program can be interrupted.

In a separate O.S., each processor can execute the O.S. routines it needs. This organization ~~needs~~ is suitable for loosely coupled systems where every processor may have its own copy of the entire O.S.

In a distributed O.S, the O.S. routines are distributed among the available processors. Each particular O.S. function is assigned to only one processor at a time & the execution may be assigned to different processors at different times. This most suitable for floating operating system.

In a loosely coupled multiprocessor system the memory is distributed among the processors & there is no shared memory for passing information. The communication b/w processors is by means of message passing through I/O channels.

A message is then sent with a header & various data objects used to communicate b/w nodes. There may be a no. of possible paths available to send the message b/w any two nodes. The O.S. in each node contains routing information indicating the alternative paths that can be used to send a message to other nodes

www.specworld.in www.smartzworld.com

The communication efficiency of the inter processor n/w depends on the communication routing protocol, processor speed, data link speed & the topology of the n/w.

Inter processor Synchronization :-

The instruction set of a multiprocessor contains basic instructions that are used to implement communication & synchronization b/w co-operating processors. Communication refers to exchange of data b/w different processes. Synchronization refers to the where data used to communicate b/w processors & control information. Synchronization is needed to enforce the correct sequence of processes & to ensure mutually exclusive access to shared writable data.

Multiprocessor system usually include various mechanisms to deal with the synchronization of resource. Low-level primitives are implemented directly by the h/w. The basic mechanisms for primitives is that enforce mutual exclusion for more complex mechanisms implemented in s/w.

The h/w mechanisms for mutual exclusion is developed i.e., binary semaphore.

Mutual Exclusion with a semaphore :

The multiprocessor system provide a mechanism that the orderly access to shared memory & other shared resources. It is necessary to protect the changing data. This mechanism has been called 'mutual Exclusion'.

Mutual exclusion in a multiprocessor system to enable one processor to exclude or lock out access to a shared resource by other processors when it is in critical section. A critical section is a program sequence that must complete execution

www.jntuworldupdates.org

before another processes access the shared resource. 11

A binary variable called a semaphore is often used to indicate whether or not a processor is executing in critical section. A semaphore is a s/w controlled flag that is stored in memory.

When semaphore is equal to 1, it means that a processor is executing a critical program, so that shared memory is not available to other processors.

When semaphore is equal to 0, the shared memory is available to any processors.

Testing & setting the semaphore is itself is a critical operation & must be performed as a single indivisible operation. If it is not, two or more processes may test the semaphore simultaneously and then each set it, and enter them into CS. This results an erroneous.

A semaphore can be initialized by means of test & set instruction in conjunction with a h/w lock mechanism. A h/w lock is a processor generated signal that serves to prevent other processors from using the system bus as long as the signal is active.

The test and set instruction tests & sets a semaphore and activates the lock mechanism during the time that the instruction is being executed.

Assume that the semaphore is a bit in the LSB of a memory word whose address is symbolized by SEM. Let the mnemonic TSL designate the "test & set while locked" operation. The instruction

TSL SEM

will be executed in two memory cycles (i.e., first to read & second to write)

www.jntuworldupdates.org

This can be represented as follows:

$R \leftarrow M[SEM]$ Test semaphore

$M[SEM] \leftarrow 1$ Set semaphore.

The semaphore is tested by transferring its value to a processor register R and then it is set to 1. The value in R determines what to do next. If the processor finds that R=1, it knows that the semaphore was originally set.

If R=0, it means that the common memory is available. The semaphore is set to '1' to prevent other processors from accessing memory. The processor can now execute the critical section. The last instruction in the program must clear location SEM to zero to release the shared resource to other processors.

Cache coherence :-

We are using separate caches for each processor is to reduce the average access time in each processor. The same information may reside in a no. of copies in some caches & main memory. The ability of a system to execute memory operation correctly, the multiple copies must be kept identical. Here, there arises a 'Cache coherence' problem.

A memory scheme is 'coherent' if the value returned on a load instruction is always the value given by the latest store instruction with the same address. Without a proper solution to the cache coherence problem, caching can't be used in bus-oriented multiprocessors with two or more processors.

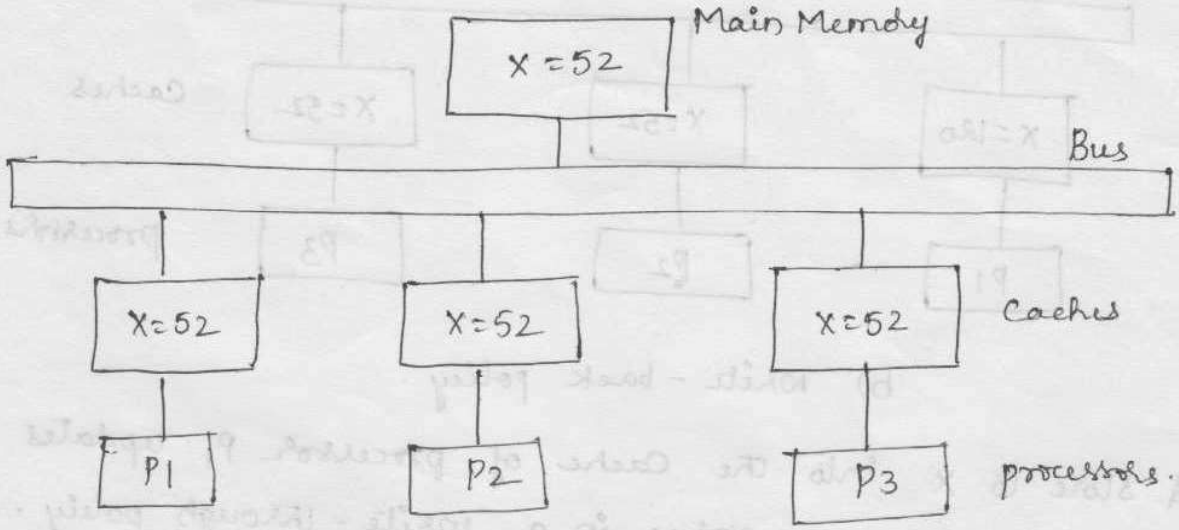
Conditions for incoherence :-

Cache coherence exists in multiprocessors with private caches because of the need to share writeable data. Read-only data can be safely placed without cache coherence mechanisms.

www.jntuworldupdates.org

To illustrate the problem,

Consider the three-processor configuration with private caches as shown below.

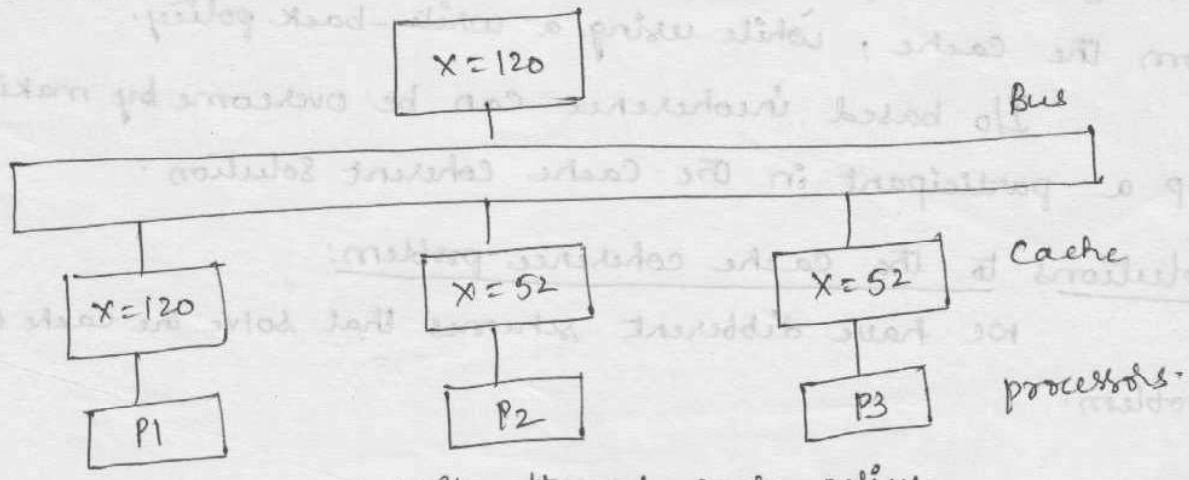


During the operation on element x from main memory is loaded into three processors P1, P2, P3. It is also copied into the private caches of the three processors.

We assume that x = 52, is load on x to three processors results in consistent copies in the caches & main memory.

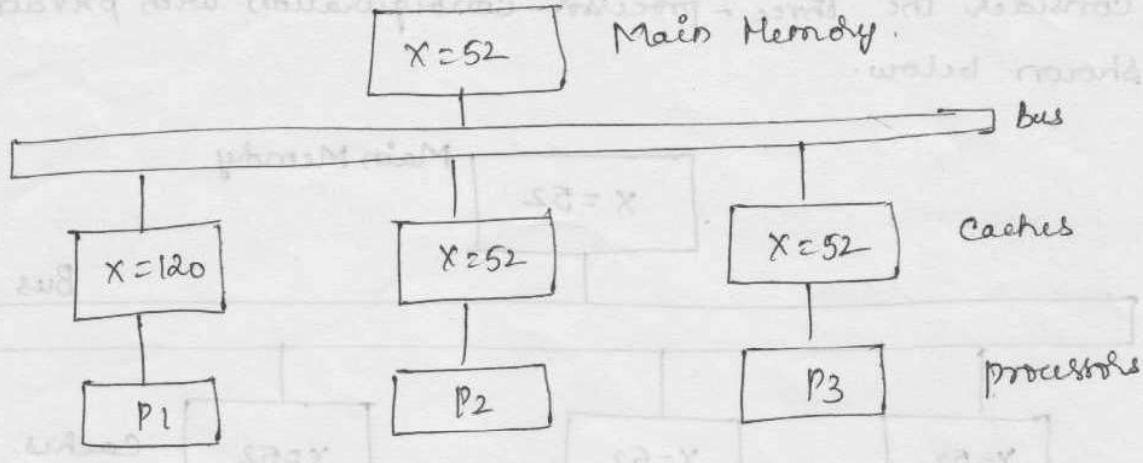
If one of the processors performs a store to x, the copies of x in the caches become inconsistent. A load by other processors will not return the latest value.

Depending on the memory update policy in the cache, the main memory may also be inconsistent with respect to the cache. This shown below.



a) Write-through cache policy.

www.jntuworldupdates.org



b) Write - back policy.

In fig ⓐ A store to X into the Cache of processor P_1 updates the memory to the new value in a write - through policy. A write - through policy maintains consistency b/w memory & the originating cache, but the other two caches are inconsistent since they hold the old value.

In fig ⓑ Main Memory is not updated at the time of the store. The copies in other two caches & main memory are inconsistent.

Another configuration that may cause consistency problems is a DMA. During DMA i/p, the locations in main memory that also reside in cache without updating the cache. During DMA O/p, memory locations may be read before they are updated from the cache, while using a write - back policy.

I/o based incoherence can be overcome by making the IOP a participant in the cache coherent solution.

<u>Solutions to the cache coherence problem:</u>

We have different schemes that solve the cache coherence problem.

www.jntuworldupdates.org

→ A scheme that each processor & have a shared cache memory associated with main memory. Every data access made shared to cache. This increases the memory access time.

→ For performance considerations, it is desirable to attach a private cache to each processor. One scheme that has been used allows only non shared & read-only data to be stored in caches. Such items are called cacheable. Shared writeable data are non cachable. The system h/w makes that only cachable data are stored in caches & the non cacheable data remains in main memory. This methods restricts the type of data stored in caches & introduces an extra s/w.

→ A scheme that allows writable data to exist in at least one cache is a called centralized global table. The status of memory blocks is stored in the central global table. Each block is identified as Read-only (RO) or read & write (RW). All caches can have copies of blocks identified as RO. only one cache can have a copy of an RW block.

. The cache coherence problem can be solved by means of a combination of s/w & h/w or by means of h/w-only schemes.

H/w-only solutions are handled by the h/w automatically & have the advantage of higher speed.

In h/w solution, the cache controller is specially designed to monitor all bus requests from CPU's & IOP's.

Depending on the Method used, they must either update or invalidate their own cache copies when a match is detected. The bus controller that monitors this action is referred as 'snoopy cache controller'. This is a h/w unit designed to

www.jntuworldupdates.org

334

maintain a bus-watching mechanisms over all the cache attached to the bus.

All the snoopy controllers watch the bus for memory store operations. The local snoopy controllers in all other caches check their memory to determine if they have a copy that has been overwritten. If a copy exists in a remote cache, the location is marked invalid. Because all caches snoop on all bus writes, when ever a word is written, the net effect is to update it in the original cache & main memory & remove it from all other caches.

(4)