Name- Mukesh Singh
Class - B.Ed ICT Sec- 3rd Semester
Roll. No- 7
Subject - Microprocessor and Computer Organization
Subject Teacher-
College- Sukuna Multiple Campus

Microprocessor and Computer Organization

Notes
ICT 3rd semester
by

Mukesh Singh
ICT 5th batch 2073

Sukuna Multiple Campus
Sunclarharaincha, Morang

Mukesh Singh SMC

Unit-1

Introduction

Page No. 1
Date

1.1 Introduction and History of Microprocessors

A microprocessor is a computer processor which incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC) or at most a few integrated circuits.

History of Microprocessors:

→ Microprocessor revolution began in 1969 by Ted Hoff (Intel Engineer).

→ The first microprocessor was introduced in 1971 by Intel Corporation. It was named Intel 4004 as it was 4 bit processor.

→ It was a processor on a single chip and can perform simple arithmetic and logic operations such as addition, subtraction, ~~multiplication and~~ Boulean AND and Boulean OR.

→ The first microprocessor was quite a success in industry. Soon other microprocessors were also introduced.

→ The first 8 bit microprocessor which could perform arithmetic and logical operations on 8 bit words was introduced in 1973 again by Intel.

Nikesh Singh SMC

→ This was intel 8008 and was later followed by an improved version, Intel 8080. Some other 8 bit processors are Zilog-80 and Motorola M6800.

→ The 8-bit microprocessors were followed by 16 bit processors. They are intel 8086 and 80286.

→ The 32 bit microprocessors were introduced by several companies but the most popular one is Intel 80386.

Microprocessors	Introduction	Data Bus	Address Bus
4004	1971	4	8
8008	1972	8	8
8080	1974	8	16
8085	1977	8	16
8086	1978	16	20
80186	1982	16	20
80286	1983	16	24
80386	1986	32	32
80486	1989	32	32
Pentium	1993 onwards	32	
Core Solo	2006	32	
Dual Core	2006	32	
Core to Duo	2006	32	
Core to Quad	2008	32	
i3, i5, i7	2010	64	

Mukesh Singh SVC

1.2 Basic Block Diagram of a Computer

The basic block diagram of a Computer is given below:

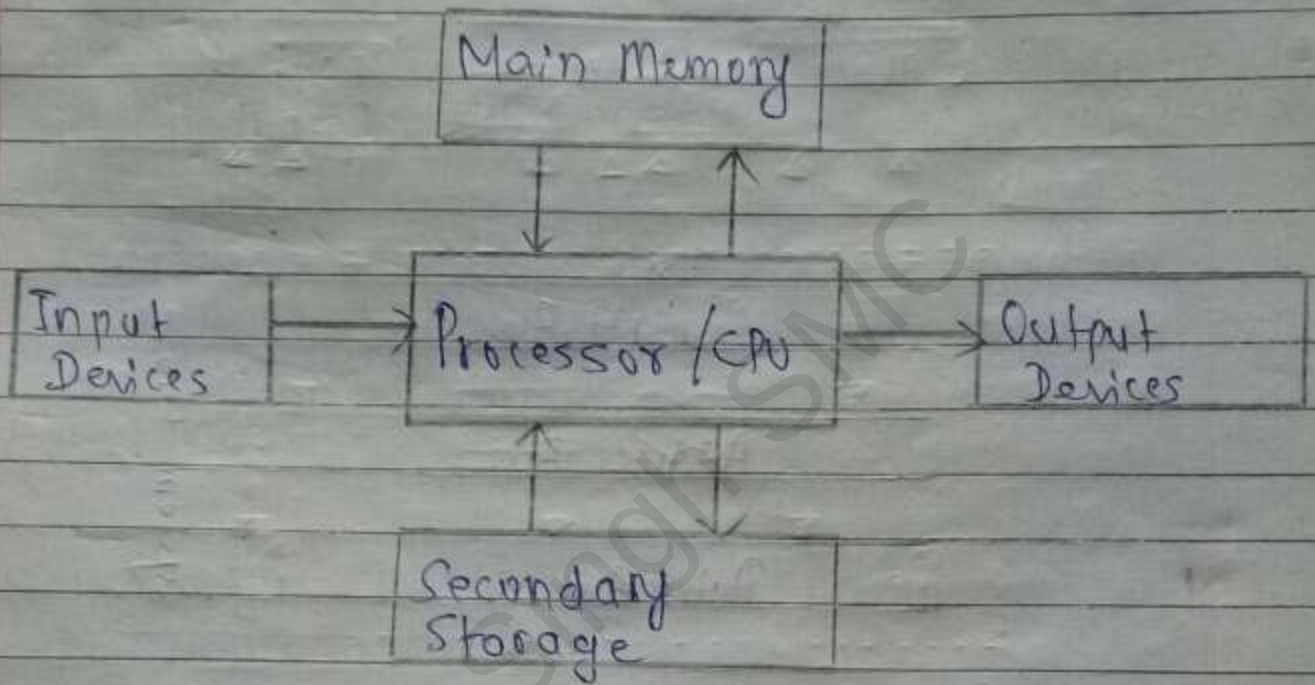
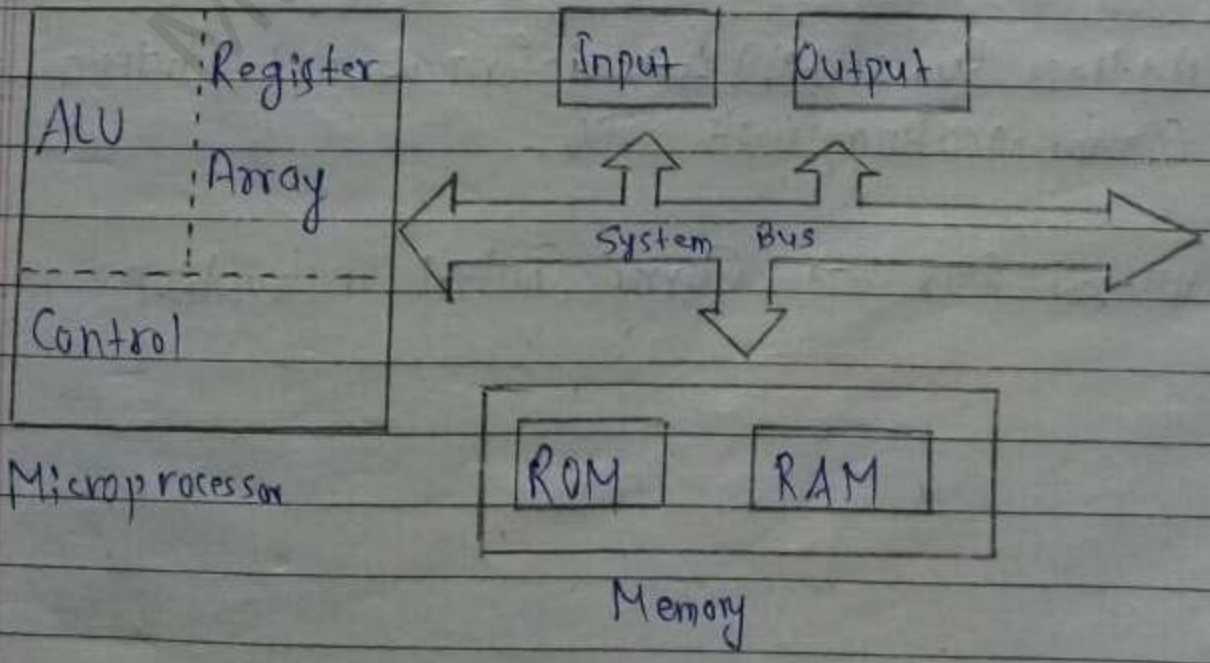


Fig: Basic block diagram of a Computer

1.3 Organization of Microprocessor Based System



Mukesh Singh SMC

1.4 Bus Organization

The diagram of Bus Organization is shown below:

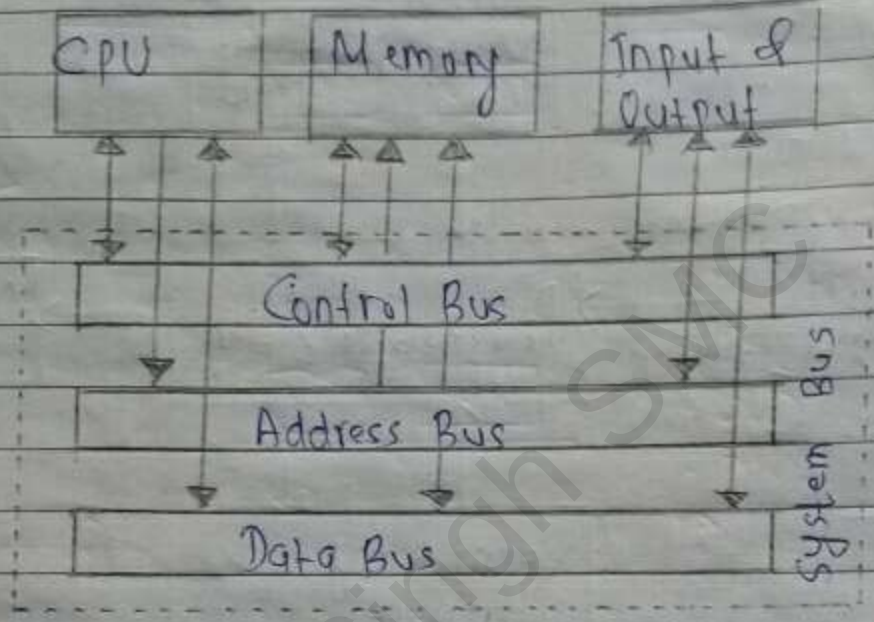


Fig: Bus System Organization

* Data Bus (16 bit) :→ responsible for transmission of data i.e. carry data.
* Address Bus (20 bit) :→ information regarding devices communicating with CPU.
* Control Bus :→ Carry Control instruction

Mukesh Singh SMC

1.5 Stored Program Concept and Von Neumann Machine

Stored program technology was discovered in 1945 AD by John Von Neumann. This became the fundamental program technology for the modern digital computer. This technology made computer programming and computing faster, more flexible and more efficient. Program can be electrically stored in binary number format in a memory device so that instructions could be modified by the computer as determined by intermediate computational results.

The basic structure proposed in the draft became known as the "Von Neumann machine". It has following parts:

→ a memory, containing instructions and data

→ a processing unit, for performing arithmetic and logical operations

→ a control unit, for interpreting instructions

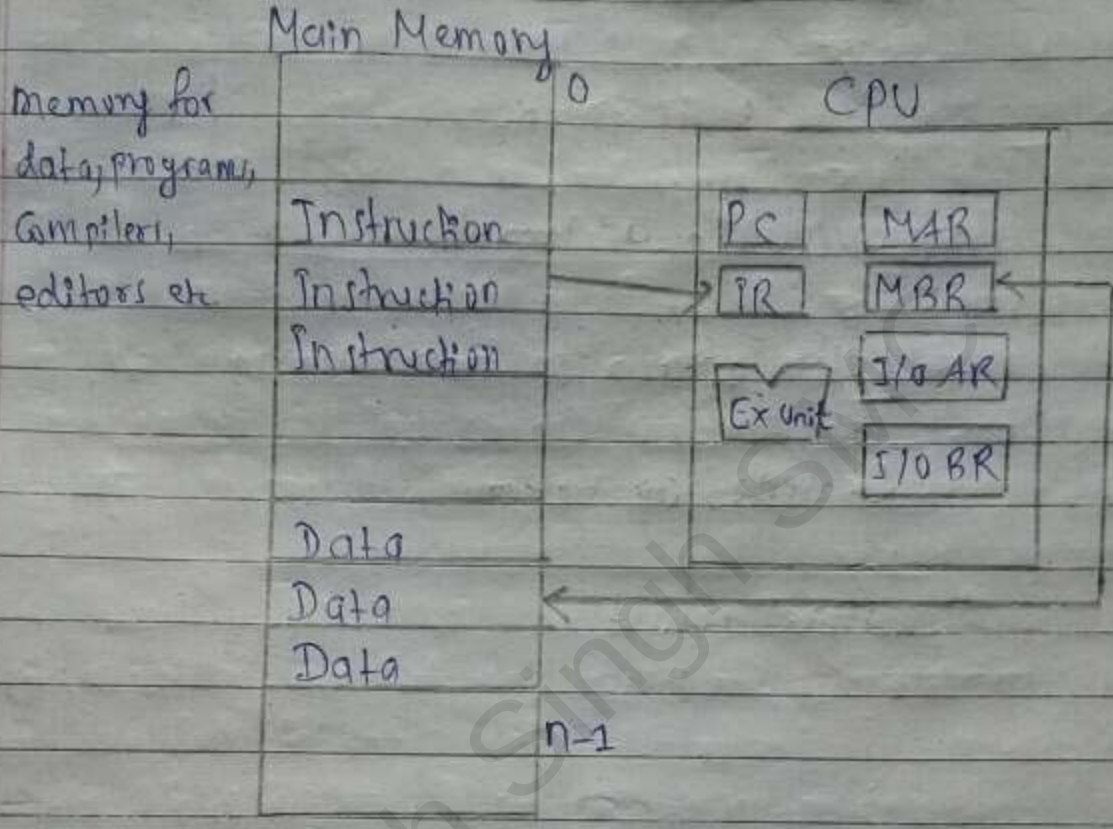
Mukesh Singh SMC

MAR → Memory Address Register
MDR → Memory Data Register

Fig: Abstraction of Von Neumann

Mukesh Singh Sir

1.6. Processing Cycle of a Stored Program Computer

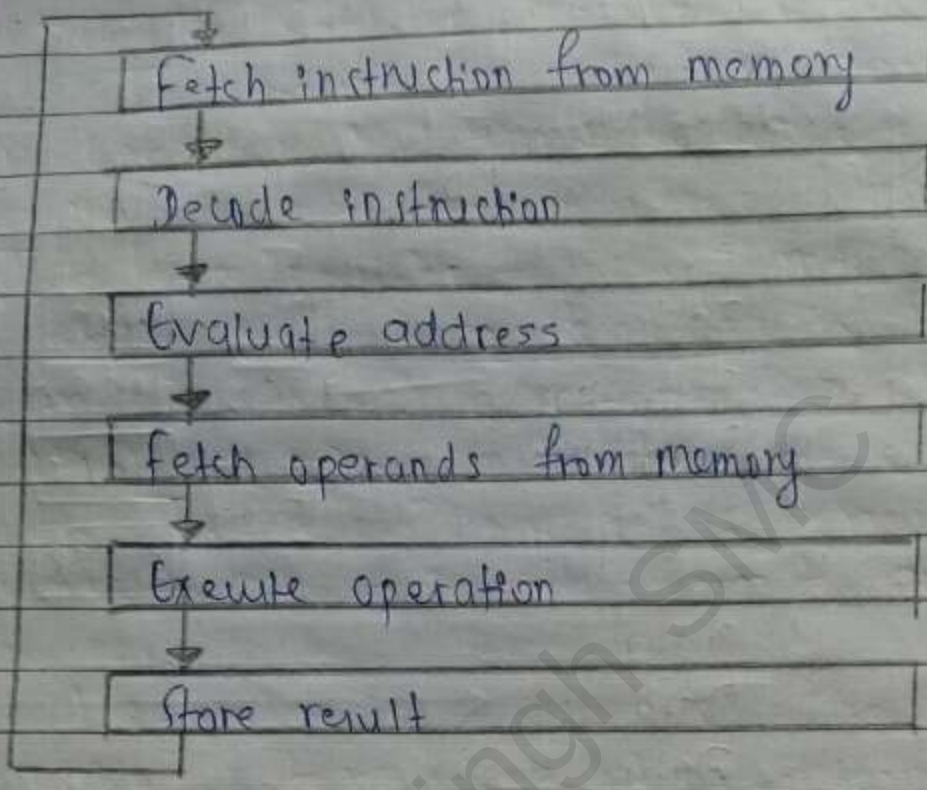


- Instructions are collection of bits
- Programs are stored in memory.

Steps

1. fetch an instruction from memory to the instruction register
2. increment the program counter register
3. decode the instruction
4. fetch operands, if any, usually from registers
5. perform the operation; this may modify the PC register
6. Store the results, usually to register

Mukesh Singh Shah



1.7 SSI, MSI, LSI circuits:

An integrated circuit (IC) is a set of electronic circuits on one small flat piece (or chip) of semiconductor material, normally silicon.

→ The first integrated circuits contained only a few (numbering in tens) transistors and so were called "Small-Scale integration (SSI).
E.g. gates, flip-flops etc

Mukesh Singh SVC

→ The increased in number of transistors to hundreds on a chip and so were called Medium-Scale Integration (MSI). Eg: 4 bit microprocessors.

→ The Large Scale integration (LSI) includes tens of thousands of transistors integrated on a single chip. Eg: 8-bit Microprocessors, RAM, ROM etc

1.8 VLSI Technology:

VLSI is a technology by which 10000 - 1 Million transistors can be fabricated on a single chip.
Eg: 16-32 bit microprocessors.

Advantages of VLSI:

1. Reduces the size of circuits
2. Reduces the effective cost of the devices
3. Increases the operating speed of circuits.
4. Requires less power than discrete components.
5. Higher Reliability.
6. Occupies a relatively smaller area.

Nikesh Singh SMC

1.9 Introduction to Register Transfer Language:-

The symbolic notation describing Micro-Operation (MU) transfers among registers is called a register transfer language.

The term "register transfer" means the availability of hardware logic circuits that can perform a stated micro-operation and transfer the result of operation to the same or another register. The word "language" is borrowed from programmers who apply this term to programming languages.

A register is a group of flip-flops which is capable of storing one bit of information. Registers are designated by capital letters followed by optional number. Eg: PC, IR, R1, R2 etc.

Basic Symbols for Register Transfers		
Symbol	Description	Examples
Letters and numerals	Denotes a register	MAR, R2
Parenthesis ()	Denotes a part of a register	R2(0-7), R2(L)
Arrow ←	Denotes transfer of information	R2 ← R1
Comma ,	Separates two micro-operations	R2 ← R1, R1 ← R2

Mukesh Singh SMC

Examples

① • Read : DR ← M[AR]
→ Read memory word M from address AR to data register DR

② Write : M[AR] ← R1
→ Writes data from R1 to a memory word M at address AR.

Arithmetic micro-operations (Mano 1993)

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

Mukesh Singh SMC

Unit-2

Programming with 8086 Microprocessor

Page No. 13
Date

2.1 Internal Architecture & features of 8086 Microprocessor:

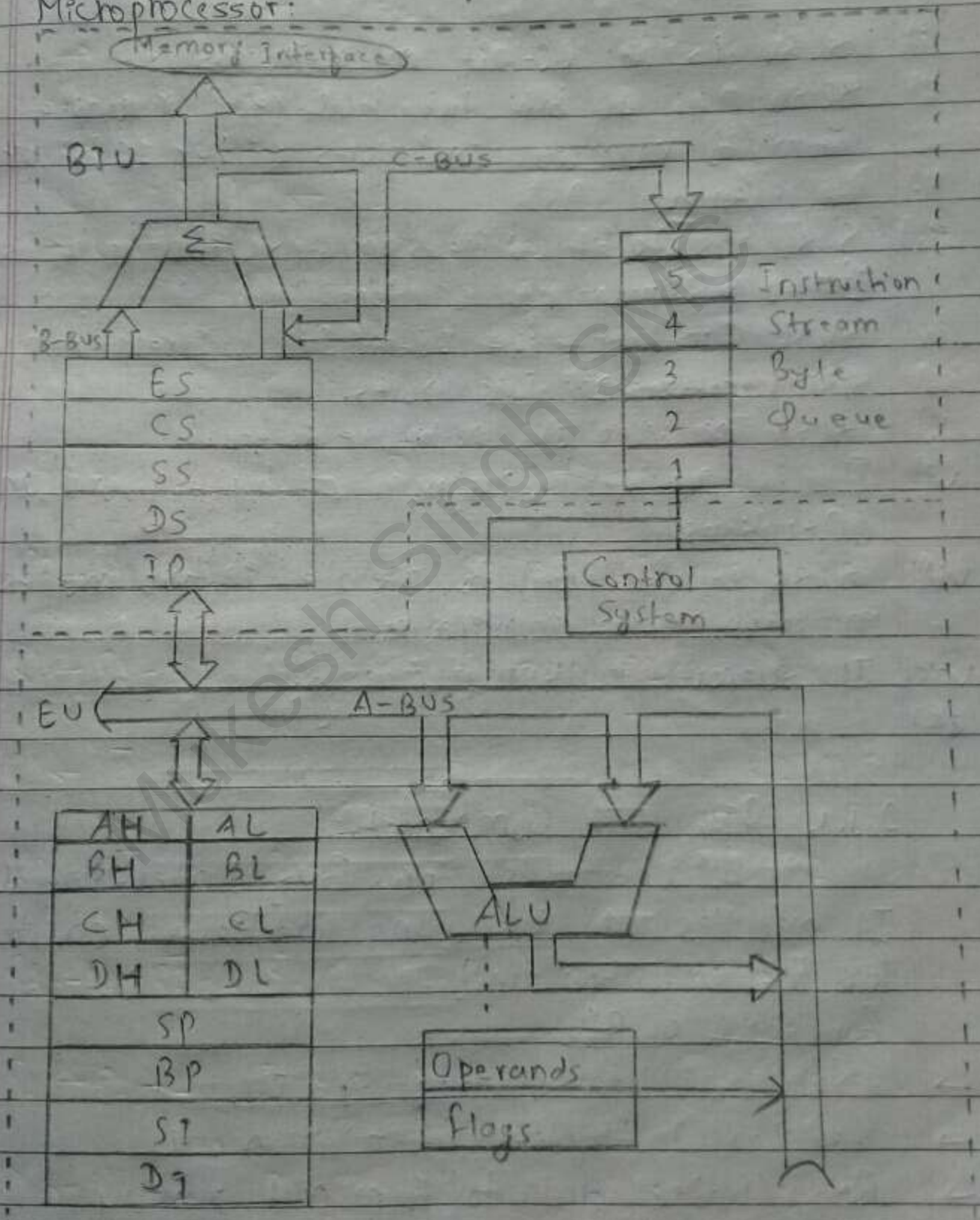


Fig: Internal Architecture of 8086 microprocessor

Mukesh Singh SMC

Features of 8086 Microprocessor:

i) 8086 is a 16-bit microprocessor.
ii) 8086 has 16-bit data bus, read or write data from/to memory and ports either 16 bit or 8 bits at a time.
iii) 8086 has 20 bit address bus, so it can access 2^{20} (1 MB) memory locations.
iv) 8086 can generate 16-bit I/O address, hence it can access 2^{16} I/O ports.
v) 8086 provides 14 16-bit registers.
vi) It is possible to perform bit, byte, word and block operations.
vii) 8086 is designed to operate in two modes; minimum and maximum.
viii) It supports multiprogramming.

2.1.1. BIU and Components:

BIU stands for Bus Interface Unit. It handles all data and addresses on the buses for the execution unit

Functions of BIU

→ Sends address of memory / IO.
→ Fetches instructions from memory.
→ Read/write data to/from port/memory.
→ Supports instruction queue and address relocation facility.

Mukesh Singh SMC

by Mukesh Singh

Page No. 15
Date:

Components or parts of BIU:
1. Instruction Queue
2. Segment Registers
3. Instruction Pointer (IP)

1. Instruction Que:
To increase the execution speed, BIU fetches as many as 6 instruction bytes ahead of time from memory. The pre-fetched instruction bytes are held for the EU in a fast in first out group of registers called a instruction queue. When the EU is ready for its next instruction, it simply reads the instruction from this instruction queue. Fetching the next instruction while the current instruction executes is called pipelining.

2. Segment Registers:
The BIU contains four 16-bit registers. They are: Extra segment (ES) registers, Code segment (CS) registers, data segment (DS) registers, and Stack segment (SS) registers. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations.
The registers that specify the location of segments are called segment registers.

Mukesh Singh SVC

i) Code Segment (CS):
The CS register is used for addressing a memory location in the Code Segment of the memory, where the executable program is stored

ii) Data Segment (DS):
It stores the starting address of data segment.

iii) Stack Segment (SS):
It handles memory to store data and addresses during execution.

iv) Extra Segment (ES):
ES is additional data segment used by some of the string to hold the extra destination data.

Fig: 8086 Memory Segment

3. Instruction Pointer (IP):

In the BIU, the next register, below the segment register is instruction pointer. It is a 16-bit register used to hold the address of the next instruction to be executed.

2.1.2 EU and Components:

EU stands for Execution Unit. It gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions.

Its function is to control operation on data using the instruction decoder & ALU. EU has no direct connection with system buses, it performs operations over data through BIU.

Components or parts of EU:

1. Control Circuitry or System
2. Instruction Decoder
3. ALU
4. Flag Registers
5. General Purpose Registers
6. Stack Pointer Registers
7. Pointer and Index Registers

1. Control Circuitry:

It performs various internal operations.

Mukesh Singh SMC

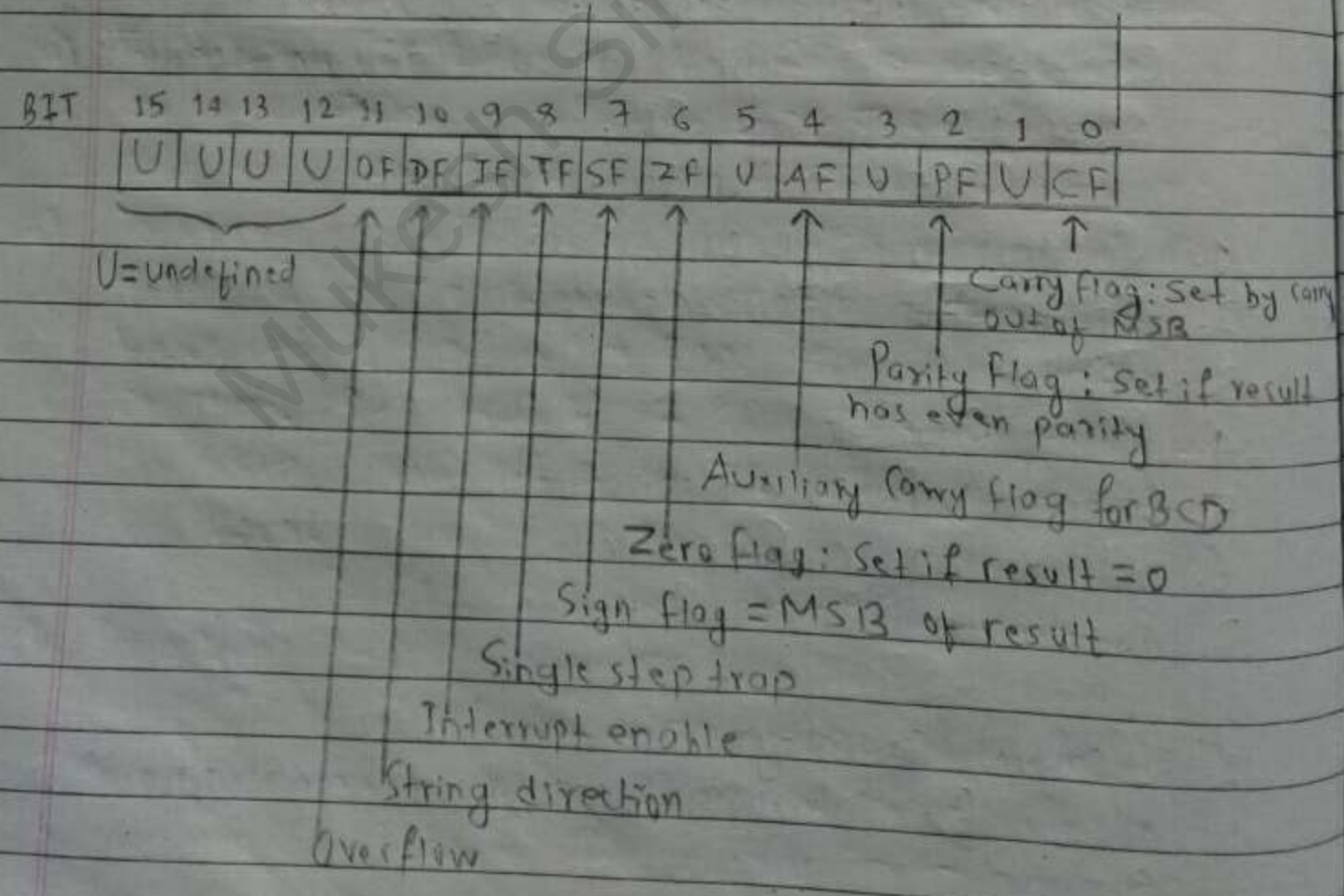
2. Instruction Decoder:

It translates instructions fetched from memory to generate different internal or external control signals that required performing the operation.

3. ALU:

The EV has a 16-bit ALV, which performs basic Arithmetic and logical operations such as: addition (+), subtraction (−), multiplication (*), division (/), OR, AND, NOT etc

4. Flag Registers:



Mukesh singh SMC

A 16-bit flag register is a flip-flop which indicates some condition produced by the execution of an instruction or controls certain operation of the EU. It has 9 flags and they are divided into two categories:

1. Conditional flags
2. Control flags

1. Conditional flags:

Conditional flags represent result of last arithmetic or logical instructions.

(i) Carry flag (CF):

This flag will set to 1 if ~~there~~ the arithmetic operation produces the carry in MSB position.

(ii) Parity flag (PF):

It is set to 1 if result of ALU is even.

(iii) Auxiliary flag (AF):

It is set to 1 if carry from lower ~~to~~ nibble to upper in addition, and borrow from upper to lower nibble in subtraction. Not available to programmers used for BCD operations.

(iv) Zero flag (ZF):

It is set to 1 if result of ALU is zero otherwise it is reset.

(v) Sign flag (SF):
It is set ~~to 1~~ if MSB bit is 1, indicates negative result.

(vi) Overflow flag (OF):
This flag represents the result when the system capacity is exceeded.

2. Control flags:
Control flags control the operation of the execution unit (EU). Following is the list of control flags:-

(i) Trap flag (TF):
It is used for debugging.

(ii) Interrupt flag (IF):
It is an interrupt enable/~~flag~~ disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.

(iii) Direction flag:
It is used in string operation. As the name suggests, when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-versa.

Mukesh Singh SMC

5. General Purpose Registers:

The EU has eight general purpose registers namela: AH, AL, BH, BL, CH, CL, DH and DL. These registers can be used individually for temporary storage of 8-bit data. The AL register is also called the accumulator. Certain pairs of these general purpose registers can be used together to store 16-bit data. The valid register pairs are: AH & AL, BH & BL, CH & CL and DH & DL. These register pairs is referred to the AX, BX, CX and DX respectively.

(i) AX Register:

For 16-bit operations, AX is called the accumulator register that stores operands for arithmetic operations.

(ii) BX register:

This register is mainly used as a base register. It holds the starting base location of a memory region within a data segment.

(iii) CX Register:

It is defined as a counter. It is primarily used in loop instruction to store loop counter.

Mukesh Singh SVC

(iv) DX Register:
This register is used to hold I/O port address for I/O instruction.

6. Stack Pointer Register:
It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack. This most recently stored memory location is called the top of stack.

7. Pointer and Index Registers:
The EU also contains a 16-bit source index (SI) register, base pointer (BP) register, and Destination Index (DI) registers. These three registers can be ~~primarily~~ mainly used for temporary storage of 16-bit data just like a general purpose registers.

Mukesh Singh SMC

2.1.3 EU and BIU Operations :

The BIU provides hardware functions. Including generation of the memory and I/O addresses for the transfer of data between itself and the outside world.

The Operations of BIU are :

→ It provides a full 16 bit bidirectional data bus and 20 bit address bus.
→ It sends address of memory or I/O.
→ It fetches instruction from memory.
→ It reads data from port/memory.
→ It makes 8086's interface to the outside world.

The EU receives program instruction codes and data from the BIU, executes these instructions, and stores the results in the general registers.

The Operations of EU are:

→ The EU is responsible for decoding and executing all instructions.
→ The EU extends instructions from the top of the queue in the BIU.
→ If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to top of the queue.
→ It tells from where to fetch instructions or data, decodes instructions & execute instructions.

Mukesh Singh

2.1.4. Segment and Offset Address:

Addresses in 8086 are 20 bit long. Addresses are computed from a segment start and an offset, both are 16 bit. The 16-bit segment start is shifted left by 4, then offset is added. These two operations are performed in a 20 bit register, so the result is a 20 bit address.

Each logical address consists of two parts: a 16-bit segment and a 16-bit offset. Physical addresses are generated by adding the segment register (not shifted) and outputting the 20-bit result. Thus, segment 0000 [all numbers in hex] offsets 0000 to FFFF represent physical addresses 00000 to 0FFFF. Adding those offsets to various other segments will yield other address ranges:

Segment	Physical addressing		Range
0000	00000	–	0FFFF
0001	00010	–	1000F
0002	00020	–	1001F
0FFF	0FFF0	–	1FFEF
1000	10000	–	1FFFF
FFFF	FFFF0	–	0FFEF (or 10FFEF if higher physical addressing bit exists)

Page No. 25
Date 01/11

2.2 Addressing modes of 8086:

Addressing modes indicates the way of locating data as operands. The different ways in which a source operand is denoted in an instruction is known as addressing modes. There are 8 different addressing mode in 8086 microprocessor:

1. Immediate addressing mode:
   The addressing mode in which data operand is apart of the instruction itself is known as immediate addressing mode.
   Example: MOV Ax, MOV AL etc

2. Register addressing mode:
   It means that the register is the source of an operand for an instruction
   Example:- Mov Cx, Ax etc

3. Direct addressing mode:
   The addressing mode in which the effective address of the memory location is written directly in the instruction is direct addressing mode.
   Example: MOV Ax, [5000H], MOV AL, [5300H]

Mukesh Singh SMC

4. Register indirect addressing mode:
This addressing mode allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI & SI.
Example: MOV AX, [BX]

5. Based addressing mode:
In this addressing mode, the offset address of the operands is given by the sum of contents of the BX/BP registers and 8-bit/16 bit displacement.
Example: MOV CX, [BX + 04].

6. Index-addressing mode:
In this addressing mode, the operand offset address is found by adding the content of SI or DI register and 8-bit/16-bit displacements.
Example: MOV BX, [SI + 16], ADD AL, [DI + 16]

7. Based-index addressing mode:
In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an index register.
Example: ADD CX, [AX + SI], MOV AX, [AX + DI]

8. Based-index with displacement mode:
In this addressing mode, the operands offset is computed by adding the base register contents. An index register contents and 8 or 16-bit displacement.

Example: - MOV AX, [BX + DI + 08],
ADD CX, [BX + SI + 16]

2.3. Assembly Language Programming:

Assembly language is a special type of low level language which consists set of alphanumeric instructions, called mnemonics. It was developed by Bob Nevelon, Jack Powell, Clement and Michael in IBM laboratory. Programming could easily be done using alphanumeric symbols instead of 0s and 1s.

Advantages

→ It is easier to understand & more standard form.
→ It is easier to find errors because of translator program.

Disadvantages

→ They are machine dependent.
→ Program execution is slower and less efficient than machine language.

Mahesh Singh SMC

2.4. High level versus Low Level Programming:

High level Programming	Low level Programming
1. High level programming languages are platform independent.	1. Low level Programming languages are platform dependent.
2. High level language programs are slower than low level language as they need to convert.	2. Low level language programs are faster than high level language programs as they don't need to convert.
3. High level languages are easy to learn.	3. Low level languages are difficult to learn.
4. High level languages are near to human languages.	4. Low-level languages are far from human languages.
5. Programs in high-level languages are easy to modify.	5. Program in low-level languages are difficult to modify.
6. Knowledge of hardware is not required to write programs.	6. Deep knowledge of hardware is required to write programs.

Mukesh Singh

7.	There languages are normally used to write application programs.	7 There languages are normally used to write hardware programs.
8.	It is divided into 3 types :- 3GL, 4G and 5GL.	8. It is divided into two types :- Machine language and assembly language.
9.	Examples :- C, C++, Java, Cobol, FORTRAN, PASCAL, Perl, BASIC, PHP etc.	9. Examples :- C, C++, machine language, assembly language etc.

2.5. Assembly Language Syntax:

The above topic describes the basic lexical elements of assembly language programming, and explains how those elements combine to form complete assembly language expressions.

It also explain how sequences of expressions are put together to form the statements that make up an assembly language program.

The syntax of an assembly language program

~~2.5.1 Comments:~~

2.5.1. Comments:

Assembly language comments begins with a semicolon (;). It may contain any printable character including blank. It can appear on a line by itself, like

```
; This program displays a message on screen
```

Or, on the same line along with an instruction, like-

```
add eax, ebx ; adds ebx to eax
```

2.5.2 Reserved words:

Reserved word is a word that cannot be used as an identifier, such as the name of a variable, function, or label - it is "reserved from use". Reserved words are not case sensitive except for predefined symbols. The assembler generates an error if we use a reserved words as a variable.

Examples of reserved words are:

Operands - $, ?, @B, BASIC, DWORD, SYCALL, PASCAL, FORTRAN.

Predefined Symbols :- @CatStr*, @Code, @Line, @FileName, @Stack* etc

Registers :- AH, AL, AX, BH, BL, BX, CH, CL, CX, etc

Mukesh Singh SMC

2.5.3. Identifiers:

An identifier is a name that we invent and attach to a definition. Identifiers can be symbols representing variables, constants, procedure names, code labels, and user defined data types such as structure, unions, records and types defined with TYPEDEF. Identifiers longer than 247 characters generate an error.

→ The first character of the identifier can be an alphabetic character (A-Z) or any of these four characters: @, -, $, ?

→ The other characters in the identifier can be any of the characters listed above or a decimal digit (0-9).

2.5.4. Statements:

Statements are the line-by-line components of source files. Each statement specifies an instruction or directive for the assembler. Statements have up to four fields,: [[name:]] [[operation]] [[operands]] [[; comment]]

2.5.5. Directives:

Directive is optional and specifies the title of the program. Like a comment, it has no effect on the program. It is just

~~used to make the program easier to understand.~~

2.5.5 Directives:

Unlike instructions being compiled and written to chip program memory, directives are commands of assembly language itself and have no influence on the operation of the microcontroller. Some of them are obligatory part of every program while some are used only to facilitate or speed up the operation. Directives are written in the column reserved for instructions. There is a rule following only one directive per program line.

2.5.6 Operators:

The assembler supports the following operator for use in expressions. Operators have no assigned precedence. Expression can be grouped in square brackets ([]) to establish precedence.

Addition (+), Subtraction (-), multiplication (*), division (/), Bitwise logical AND (&), Bitwise logical OR (|), shift right (>>), shift left (<<) etc.

Note:→ The asterik (*), slash (/), and percent sign (%) characters are overloaded. When used as operators in an expression, these characters must

be preceded by the backslash (\).

2.5.7. Instructions:

There exist around 150 instructions for the 8086 processor. There instructions may be classified into different classes. The following table summarizes the different classes of instructions, together with examples of each class.

Instruction Type	Definition	Examples
Data transfer instructions	Transfer information between registers & memory locations or I/O parts.	MOV, LEA, POP
Arithmetic instruction	Perform arithmetic operations on binary or binary-coded-decimal (BCD) numbers.	ADD, SUB, INC, DEC
Bit manipulation instructions	Perform shift, rotate & logical operations on memory locations & registers.	SHL, SHR, SAR, ROL
Control transfer instructions	Control sequence of program execution. Include jumps & procedure transfers.	JL, JE, JNE, JGE
String handling instructions	Move, compare & scan strings of data.	OVSB, CMPS, MOVSW
Processor control instructions	Set and clear status flags, and change the processor execution state.	~~INT, INTO, IRET~~ STC, STD, STI

Mukesh Singh SMC

Miscellaneous instructions		NOP, WAIT
Interrupt instructions	Interrupt processor to service a specific condition	INT, INTO, IRET

2.6. EXE and COM Programs:

.EXE files

→ .EXE is executable file which is a main entry point for execution of a program.
→ .EXE Code files use RAM more efficiently.
→ Data and Code occupy separate segments.
→ The programmer is reponsible for setting up the data and code segments properly.

.COM files:

→ The .COM files format is a historic file format since the first version of MS-Dos
→ All code, data and stack occupy one 64K segment.
→ The COM format is the original binary executable format used in Control Program for Microcomputers and MS-Dos.
→ It is very simple.
→ It has no header (with the exception of CP/M 3 files), and contains only code and data.

Mukesh Singh SMC

2.7. Assembling, Linking and Executing:

1. Assembling:

→ Assembling converts source program into object program if syntactically correct and generates an intermediate .obj file or module.

→ It calculates the offset address for every data item in data segment and every instruction in code segment.

→ A header is created which contains the incomplete address in front of the generated obj module during the assembling.

→ Assembler complains about the syntax error if any and does not generate the object module.

→ Assembler creates .obj, .lst and .crf files and last two are optional files that can be created at run time.

→ For short programs, assembling can be done manually where the programmer translates each mnemonic into machine language using lookup table.

→ Assembler reads each assembly instruction of

Mukesh Singh SMC

a program as ASCII character and translates them into respective machine code.

Assembler Types:

a) One pass assembler
It scans the assembly language program once and converts to object code at the same time.

b) Two pass assembler:
It scans the assembly language twice.

2. Linking:
→ This involves the converting of .OBJ. module into .EXE (executable) module i.e. executable machine code.

→ It completes the address left by the assembler.
→ It combines separately assembled object files.
→ Linking creates .EXE, .LIB & .MAP files among which last two are optional files.

3. Executing:
→ It loads the program in memory for execution.
→ It resolves remaining address.
→ This process creates the program segment prefix (PSP) before loading.

Mukesh Singh SMC

→ It executes to generate the result.

Sample program —assembling→ Object program —linking→ executable program

2.8 One pass and two pass assemblers:

1 One pass Assemblers

→ This assembler scans the assembly language program once and converts to object code at the same time.

→ This assembler has the program of defining forward references only.

→ This jump instruction uses an address that appears later in the program during scan. for that case, the programmer defines such addresses after the program is assembled.

2. Two pass assembler:

→ This type of assembler scans the assembly language twice.

→ First pass generates symbols table of names and labels used in the program and circulates their relative address.

→ The table can be seen at the end of the list file and here user need not define anything.

→ Second pass uses the table constructed

in the first pass and completes the object code creation.
→ The assembler is more efficient and easier than earlier.

2.9 Keyboard and Video Services:

The Intel CPU uses two types of interrupt: hardware and software interrupt. The common interrupts are: 10H, is used for video services and 21H, is used for DOS.

INT 21H:

It is called the DOS function call for keyboard operation. The different services under it are:

i) 00H :- It terminates the current program. However, it is generally not used instead 4CH is used.

ii) 01H :- It reads the character with echo. It waits for the character if the buffer is empty. The character that is read is returned in AL in ASCII value.

Eg:

```
.data
  char DB ?
```

Mukesh Singh SMC

```
.code
MOV AH, 01H
INT 21H
MOV char AL
```

iii) 02H : It is used to display the single character. The character is sent to DL for display.

Eg:

```
MOV AH, 02H
MOV DL, 'B'; Returns the ASCII value of B To DL
                                    i.e 61
INT 21H
```

iv) 03H and 04H :- used as auxiliary input output. However, INT 14H is preferred instead.

v) 05H :- Also called printer service as it sends the character in DL to printer.

Eg:

```
:data
.code
MOV AH, 05H; select printer output
MOV DL, 65; character to be printed
```

vi) 0CH: Clears the keyboard buffer and invoke input function such as 01, 06, 07, 08 or 0A. AL will contain the input function.

Mukesh Singh SMC

INT 10H:

It is called video display control. It controls the screen format. It controls the screen format color, text style, making window scrolling etc. The control functions listed below are obtained in AH register.

i) 00H :- sets the video mode
ii) 01H :- Set cursor lines
iii) 02H :- Set cursor position
iv) 03H :- Get cursor position and size
v) 06H :- Scroll window up
vi) 07H :- Scroll window down
vii) 08H :- Read character and attribute
viii) 09H :- Write character and attribute
ix) 0AH :- Write character
x) 10H (AL=03H) :- Toggle blinking/intensity bit
xi) 0FH :- Get video mode

2.11. Pin Configuration of 8086 Microprocessor:

Muneshwar Singh SMC

2.

1. Minimum Mode:

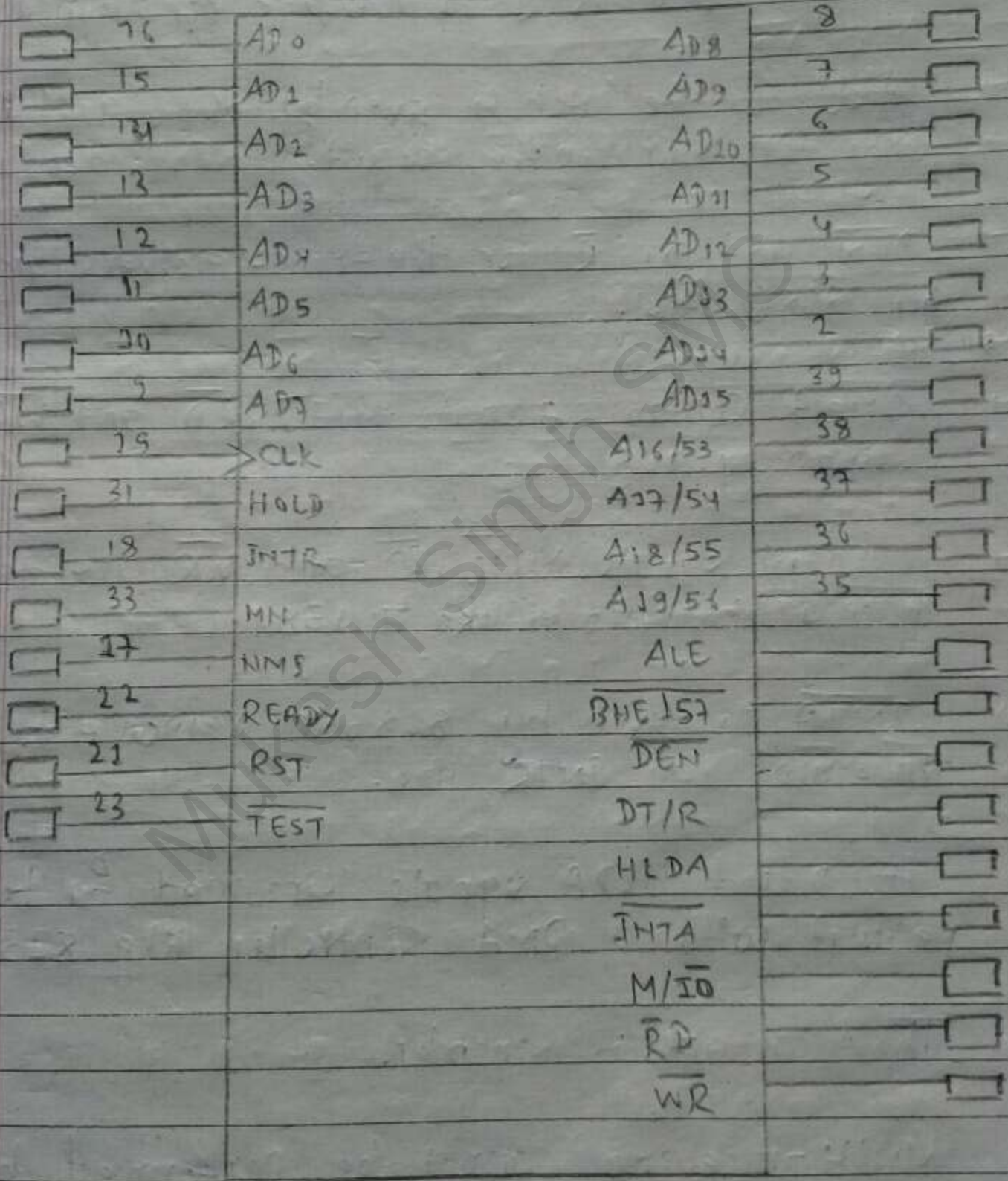


Fig (a) 8086 minimum mode

Mukesh Singh SVC

→ In minimum mode, there can be only one processor i.e. 8086.

→ MN/$\overline{MX}$ is 1 to indicate minimum mode.

→ ALE for the latch is given by 8086 as it is the only processor in the circuit.

→ $\overline{DEN}$ and DT/$\overline{R}$ for the trans-receivers are given by 8086 itself.

→

→ Direct control signals M/$\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ are given by 8086.

→ Control signals M/$\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ are decoded by a 3:8 decoder like 74138.

→ $\overline{INTA}$ is given by 8086 in response to an interrupt on INTR line.

→ HOLD and HLDA signals are used for bus request with a DMA controller like 8237.

→ The circuit is simpler.

→ Multiprocessing cannot be performed, hence performance is lower.

2. Maximum Mode:

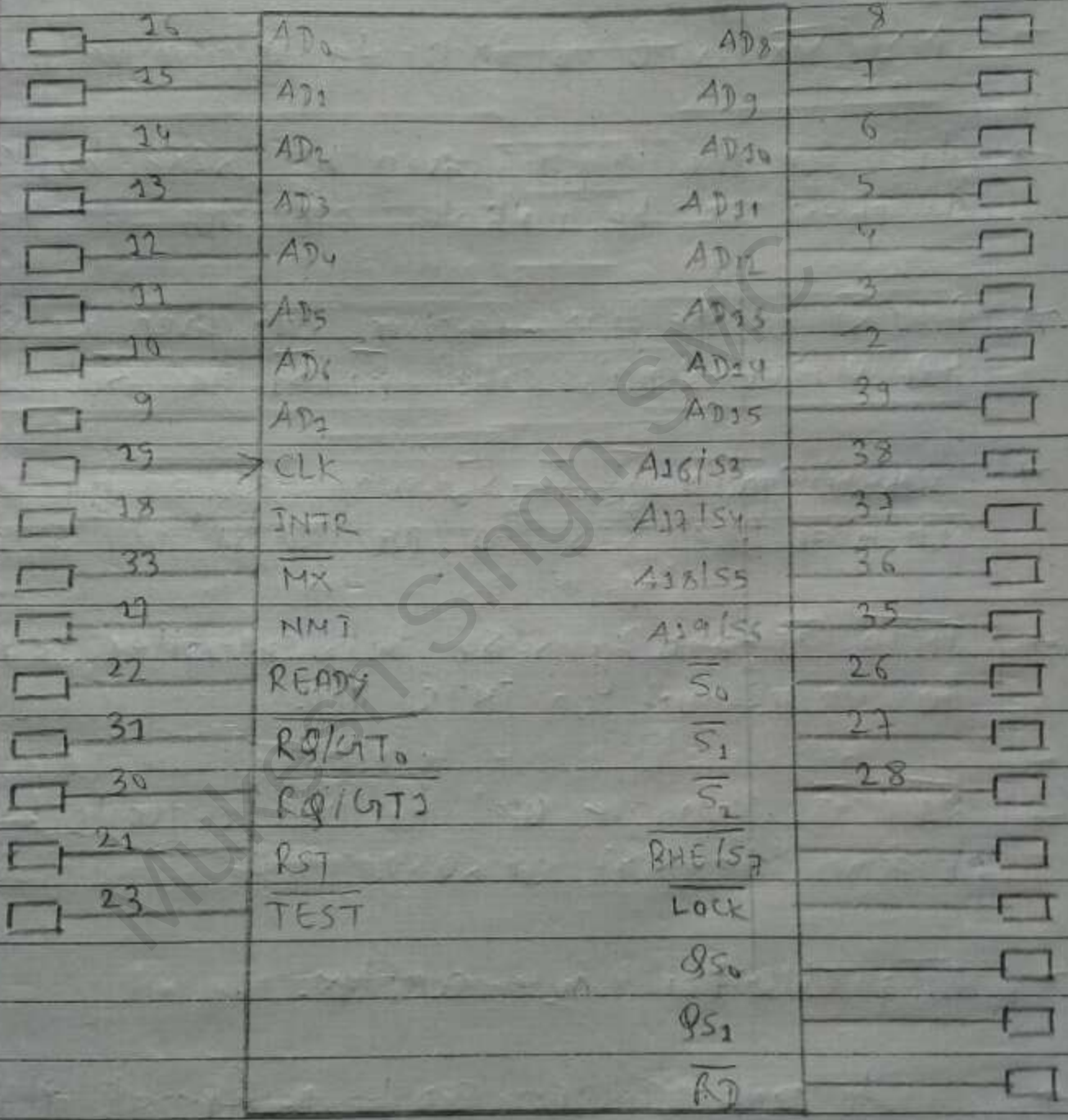


Fig: (b) 8086 Maximum mode

Mukesh Singh SMC

→ In maximum mode, there can be multiple processors with 8086, like 8087 & 8089.

→ MN/$\overline{MX}$ is 0 to indicate maximum mode.

→ ALE for the latch given by 8288 bus Controller as there can be multiple processors in the circuit.

→ DT/$\overline{R}$ for the trans-receivers are given by 8288 bus status signals called $\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$.

→ Status signals $\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ are decoded by a bus Controller like 8288 to produce control signals.

→ $\overline{INTA}$ is given by 8288 bus controller in response to an interrupt on INTR line.

→ $\overline{RQ}$ / $\overline{GT}$, liner are used for bus requests by other processor like 8087 or 8089.

→ The circuit is more complex.

→ As multiprocessing can be performed, it can give very high performance.

Mukesh Singh SMC

2.12 Bus Structures:

2.12.1. Synchronous Bus

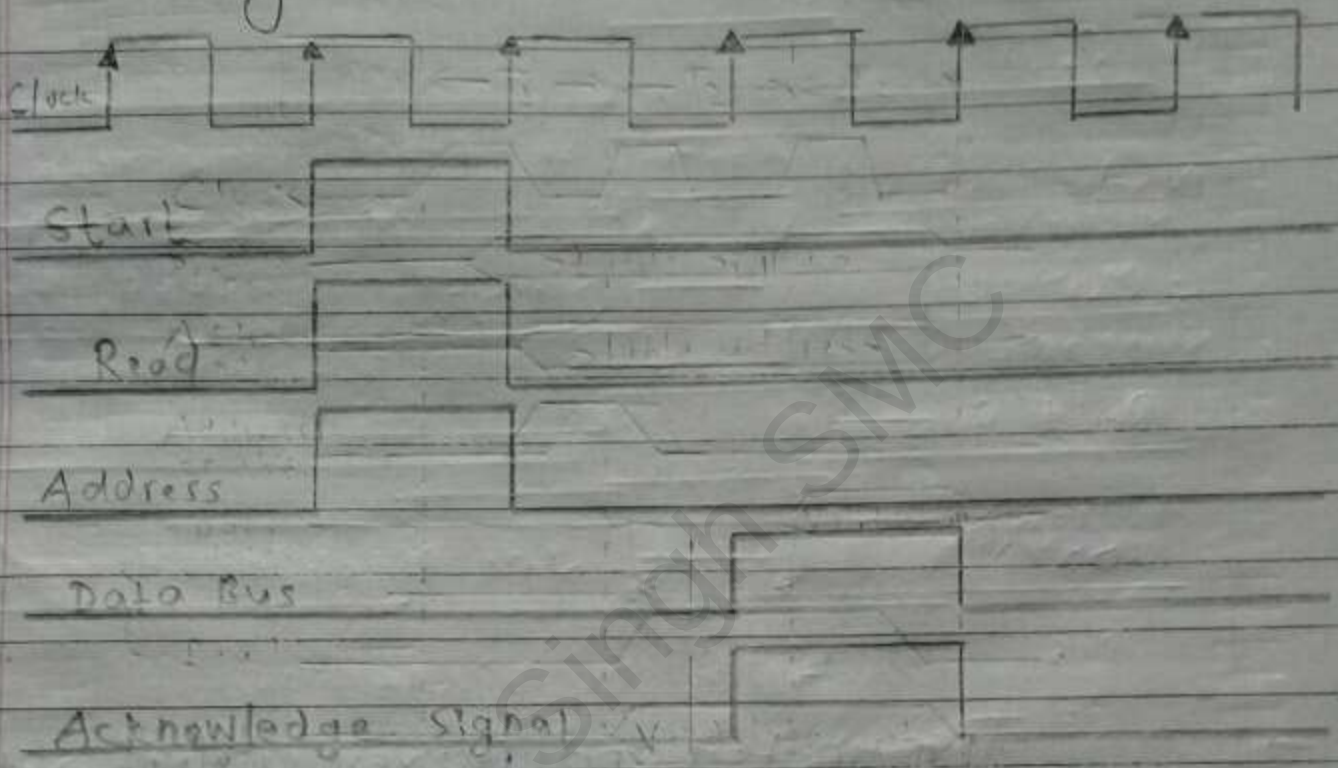


fig: Synchronous ~~Timing Diagram~~ Bus

→ Transmitters and receiver are ~~not~~ synchro-nized by clock.
→ Data bits are transmitted with synchroniza-tion of clock.
→ Character is received at constant rate.
→ Data transfer take place in blocks.
→ Start and stop bit are required to establish communication of each character.

Mukesh Singh SMC

→ Used in high-speed transmission.

2.12.2 Asynchronous Bus

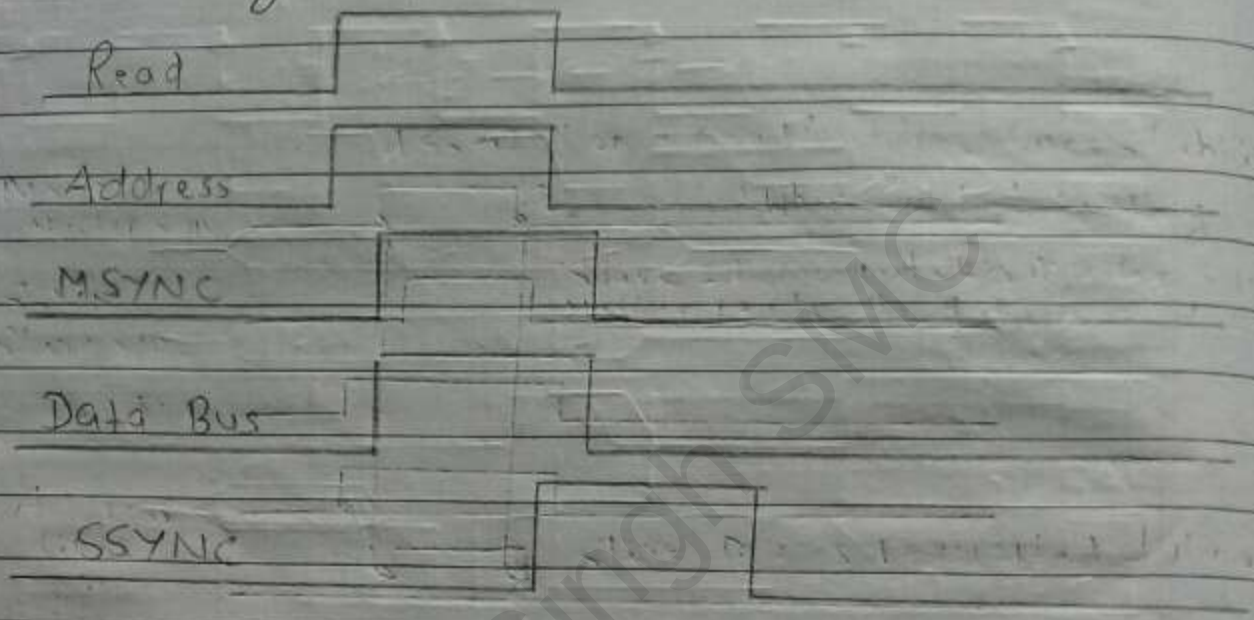


Fig: Asynchronous Bus

→ Transmitter and receiver are not synchronized by clock.

→ Bits of data are transmitted at constant rate

→ Character may arrive at any rate at receiver.

→ Data transfer is character oriented.

→ Start and stop bits are required to establish communication of each character.

→ Use in low-speed transmission.

Mukesh Singh SMC

Unit-3

Central Processing Unit

Page No. 93
Date

3.1. ~~CPU Structure and function~~

Central Processing Unit (CPU) is the portion of a computer system that carries out the instructions of a computer program, and is the primary element carrying out the computer's function. The CPU carries out each instructions of the program in sequence to perform basic arithmetical, logical and input/out operations of the system. A CPU on single chip is called Microprocessor.

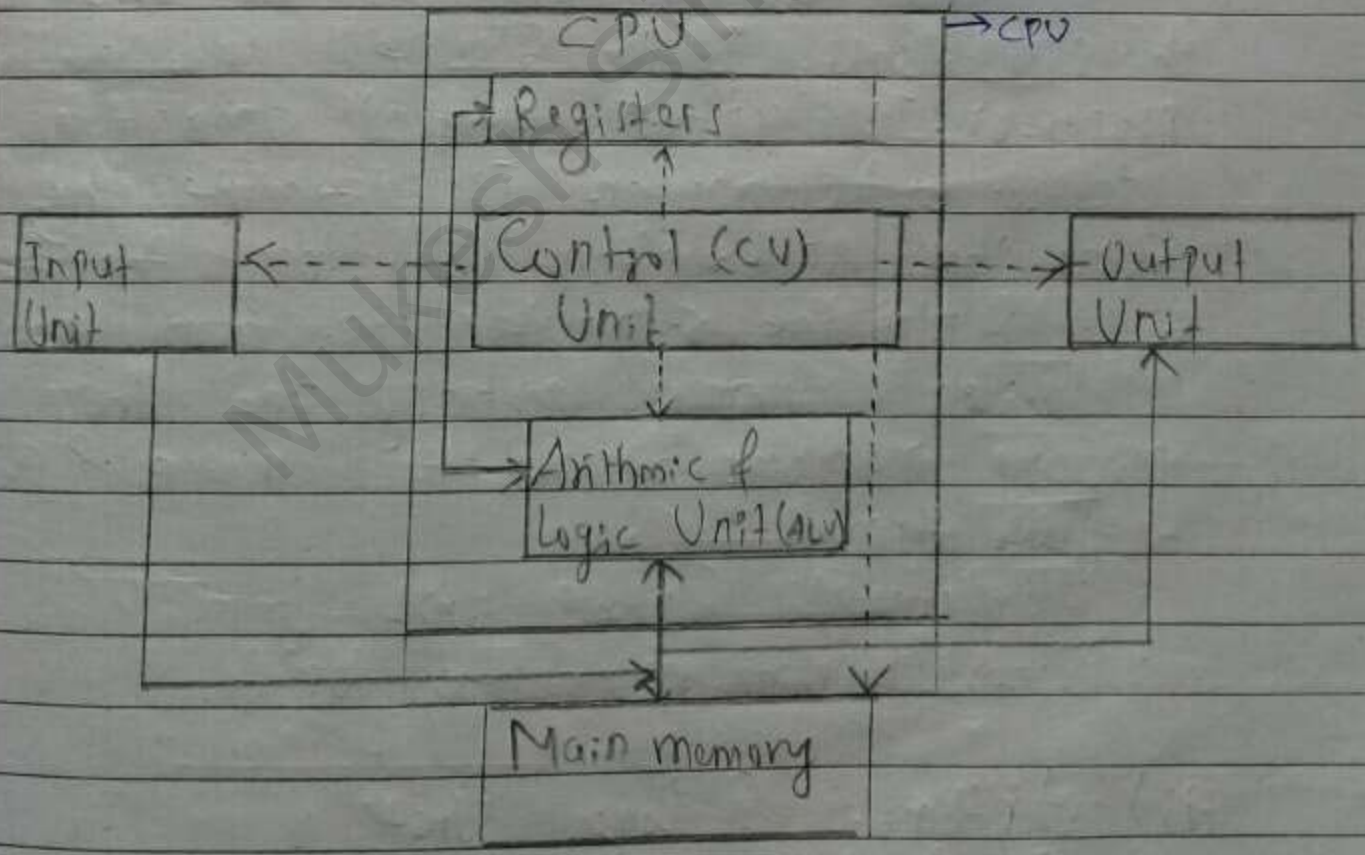


fig: Presenting CPU in Computer system

Mukesh Singh BSC

Primary Components of CPU are:

i) Registers:
Registers are primary memory of Computer system. They are mainly used to store data during the time of processing inside ALU.
- They are fastest, less capacity and temporary memory.

ii) Control Unit:
The Control Unit is the brain of the CPU itself. It is situated inside the processor. It controls overal operations and devices of computer. It executes the computer instructions. It directs the flow of data in the computer i.e. from input device to primary memory and from primary memory to output device & so on.
Two types:
- Hardwired Control Unit
- Micro programmed Control Unit.

iii) Arithmetic & Logic Unit
- ALU performs basic arithmetical operations such as addition, subtraction, multiplication and division.
- It performs logical operations such

Mukesh Singh SMC

as comparing greater than, equal to etc.

3.1. CPU Structure and Function

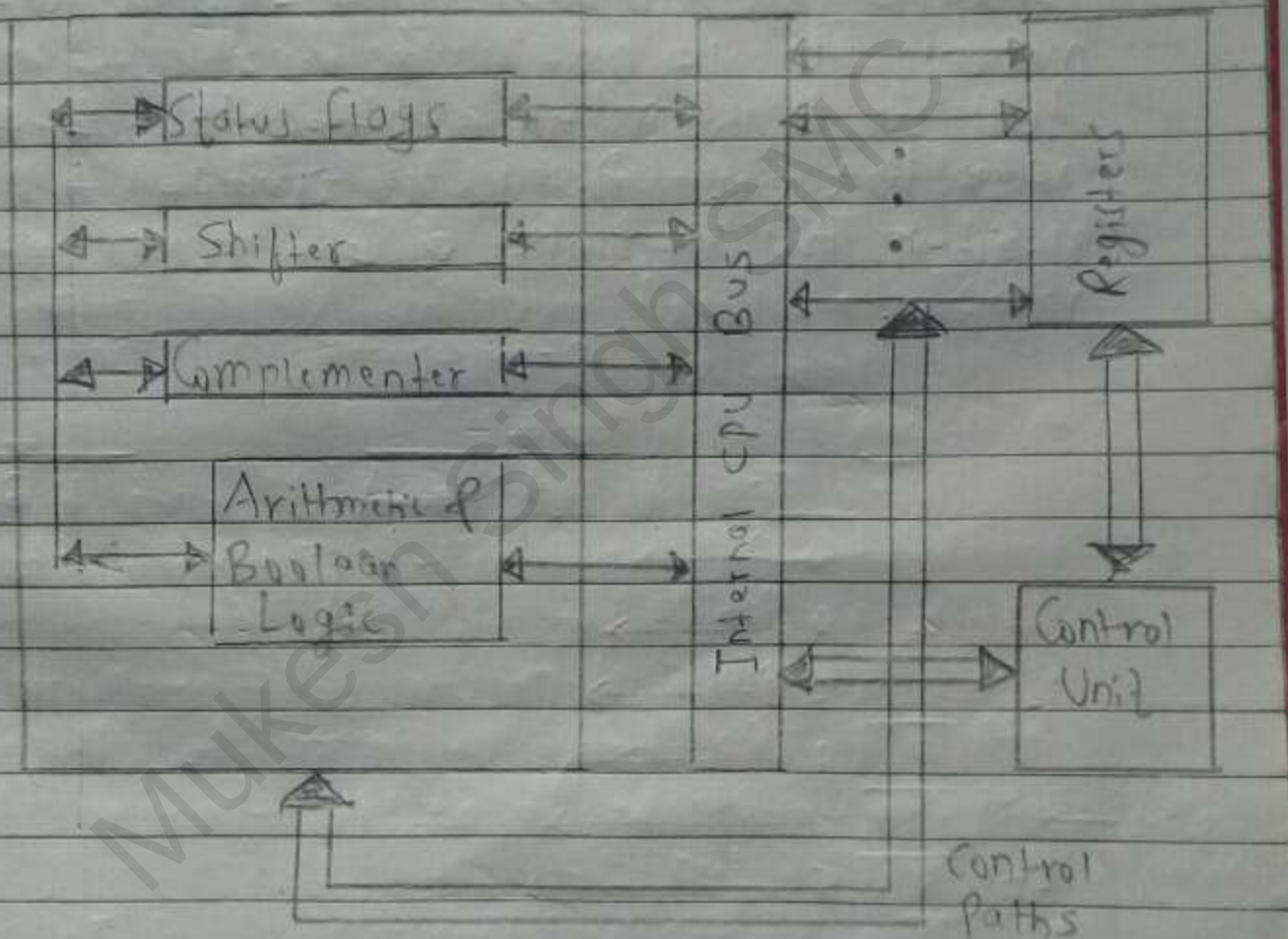


fig: Internal Structure of CPU

Functions of CPU:
- Fetch instructions
- fetch data
- Interpret Instructions
- Process Data
- Write Data

Mukesh Singh SMC

3.2 ~~ALU~~ Arithmetic and Logic Unit

An arithmetic and logic Unit (ALU) is a digital electronic circuit that performs arithmic and logical operations on data. All of the other elements of computer system - control unit, registers, memory, I/O are mainly to bring data into the ALU for it to process and then to take the result back out. Data are presented to ALU in register and the result of operation is stored in register.

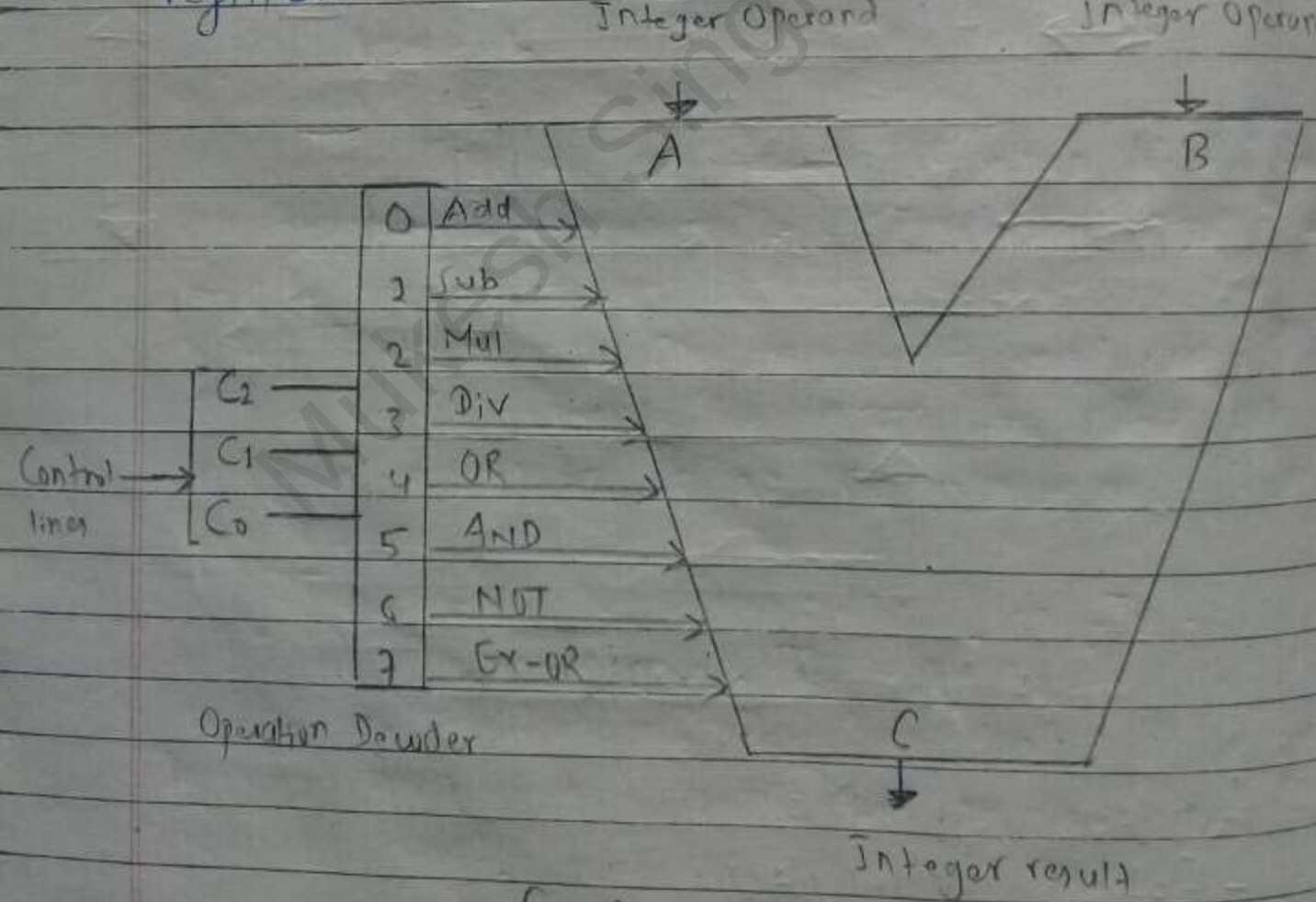


Fig: Block diagram of ALU

Consider an ALU having four Arithmetic

Operations ~~used~~ :- Addition, Subtraction, Multiplication and division. Also consider ALU having four logical operations : OR, AND, NOT, EX-OR

We need three control lines to identify any one of these operations. Control lines are used to identify any one of the four operations in a group. One possible combination is given below.

C_1	C_0	Arithmetic $C_2 = 0$	Logical $C_2 = 1$
0	0	Addition	OR
0	1	Subtraction	AND
1	0	Multiplication	NOT
1	1	Division	Ex-OR

A decoder is used to decode the instructions as given above in the block diagram.

Mukesh Singh CMC

3.3 Stack

The stack is a block of memory that may be used for temporarily storing the contents of the registers inside the CPU.

It is top-down data structure whose elements are accessed using the stack pointer (SP) which gets decremented by 2 as we store a data word into the stack and gets incremented by 2 as we retriere a data word from the stack back to the CPU register.

The process of ~~trans~~ storing the data in the stack is called 'pushing into' the stack. ~~and th~~

The reverse process of transferring the data back from the stack to the CPU register is known as 'popping off' the stack.

The stack is essentially Last-In-first-Out (LIFO) data segment. This means that the data which is pushed into the stack last will be on top of stack and will be popped off the stack first.

Mukesh Singh SMC

The stack pointer (SP) is a 16-bit register that contains the offset address of the memory location in the stack segment.

The stack segment (SS) register contains the base address of the stack segment in the memory.

The stack segment register (SS) and Stack Pointer (SP) register together address the stack-top as explained below:

SS ⇒ 5000H

SP ⇒ 2050H

Mukesh Singh SMC

3.4. Processor Organization:

Things CPU must do:

- **Fetch instruction:**
  The processor reads an instruction from memory (register, cache, main memory).

- **Interpret instruction:**
  The instruction is decoded to determine what action is required.

- **Fetch data:**
  The execution of an instruction may require reading data from memory or an I/O module.

- **Process data:**
  The execution of an instruction may require performing some arithmetic or logical operation on data.

- **Write data:**
  The results of an execution may require

Mukesh Singh MC

writing data to memory or I/O module

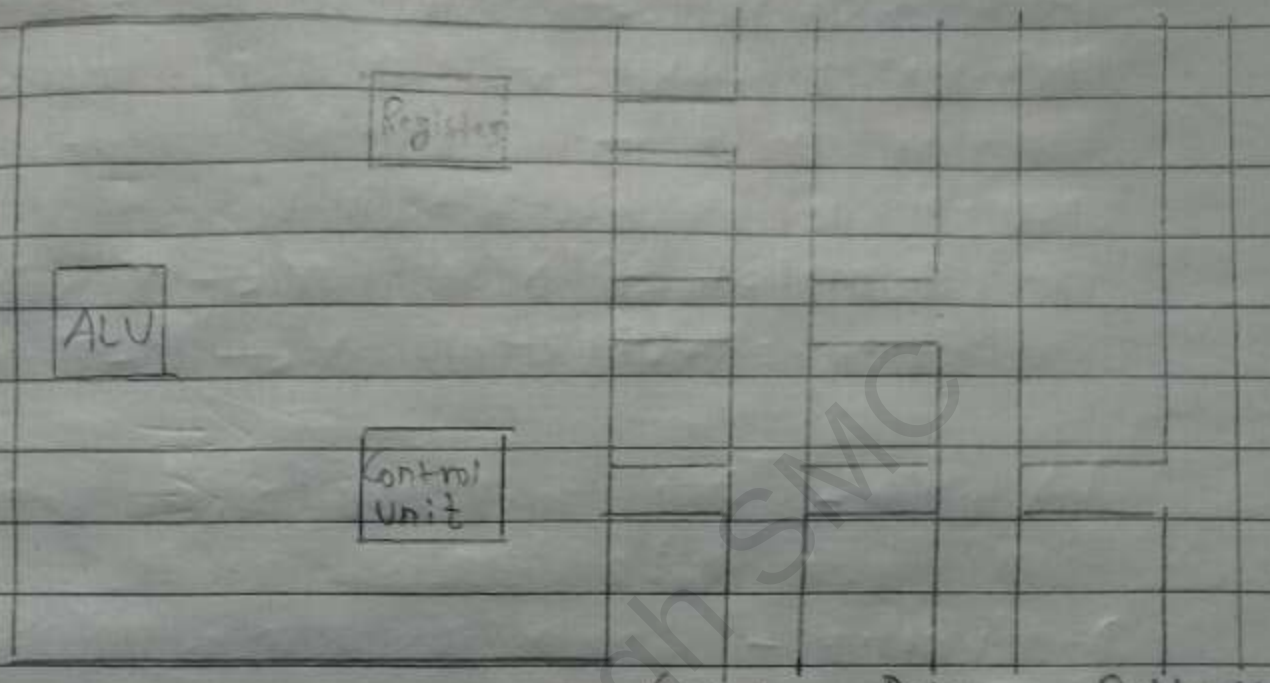


Fig: Simplified view of Processor

Components of Processor:

- ALU does the actual computation of processing of data.
- The Control Unit controls the movement of data and instructions into and out of the processor and controls the operations of the ALU.
- Register consists of set of storage locations.

Mukesh Singh SMC

3.6 Instruction formats:

Instruction format of a computer instruction usually contains 3 fields: operation code (opcode), address field and mode field. The number of address fields in the instruction format depends on the internal organization of CPU. On the basis of no. of address field, we can categorize the instruction as below:

i) Three address instructions:

Computers with three address instruction can use each address field to specify either a processor register or a memory operand.

~~Assembly language program to evaluate X = (A+B) * (C+D):~~

~~ADD R1, A, B~~

format: Op X, Y, Z ; X ← Y op Z

Example: ADD X, Y, Z ; X ← Y + Z

Advantage: It results in short programs when evaluating arithmetic expressions.

Disadvantage: The instructions requires too many bits to specify 3 addresses.

ii) Two address instructions:

Two address instructions are most common in commercial computers. Here again each address field can specify either a processor register, or a memory word. One address must do double duty as both operand and result.

Format: Op X, Y; $X \leftarrow X$ Op Y.

Example: SUB X, Y; $X \leftarrow X - Y$

Advantage - Tries to minimize the size of instruction.

Disadvantage - Size of program is relatively larger.

iii) One-address instructions:

One-address instructions uses an implied accumulator (AC) register for all data manipulation. All operations are done between AC and memory operand.

Format: Op X; $AC \leftarrow AC$ Op X

Example: MUL X; $AC \leftarrow AC * X$

Advantage:- Memory access is only limited to load and store.

Disadvantage:- Larger program size.

Mukesh Singh SMC

iv) Zero-address instructions:

A stack organized computer uses this type of instructions. The name "zero-address" is given to this type of computer because of the absence of an address field in the computational instructions.

format : TOS ← TOS Op (TOS-1)

Example : DIV; TOS ← TOS DIV (TOS-1)

Assembly programs to evaluate the following instructions:

X = (A+B) * (C+D) using zero, one, two or three

i) Three address instructions:

```
ADD R1, A, B ;   R1 ← M[A] + M[B]
ADD R2, C, D ;   R2 ← M[C] + M[D]
MUL X, R1, R2;   M[X] ← R1 * R2
```

ii) Two address instructions:

```
MOV R1, A;   R1 ← M[A]
ADD R1, B;   R1 ← R1 + M[B]
MOV R2, C;   R2 ← M[C]
ADD R2, D;   R2 ← R2 + M[D]
MUL R1, R2;  R1 ← R1 * R2
MOV X, R1 ;  M[X] ← R1
```

Mukesh Singh SMC

iii) One address instructions:

```
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]
MULT;      Ac ← Ac * M[T]
STORE X;   M[X] ← Ac
```

Here, T is temporary memory location required for storing the intermediate result.

iv) Zero address instructions:

```
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
```

Mukesh Singh SMC

3.8 Data Transfer and Manipulation:

Computers are given extensive set of instructions to give the user the flexibility to carryout various computational tasks. The actual operations in the instruction set are not very different from one computer to another although binary encodings and symbol name (operation) may vary. So, most computer instructions can be classified into 3 categories:

1. Data transfer instructions
2. Data manipulation instructions
3. Program Control instructions

1) Data transfer instructions:

Data transfer instructions cause transfer of data from one location to another without changing the binary information. The most common transfers are between the:

- Memory and Processor registers
- Processor registers and input output devices
- Processor registers themselves

Typical data transfer instructions:

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH

Input	IN
Output	OUT
Push	PUSH
Pop	POP

2) Data manipulation instructions:

Data manipulation instructions perform operations on data and provide the computational capabilities for the computer. There instructions perform arithmetic, logical and shift operations.

Arithmetic instructions

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

Logical and Bit manipulation Instructions:

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR

Mukesh Singh Sir

Exclusive OR	XOR
Clear Carry	CLRC
Set Carry	SETC
Complement Carry	COMC
Enable interrupt	EI
Disable interrupt	DI

Shift instructions

Name	Mnemonic
Logical Shift Right	SHR
Logical Shift Left	SHL
Arithmetic Shift Right	SHRA
Arithmetic Shift Left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through Carry	RORC
Rotate left through Carry	ROLC

3) Program Control Instructions

The program control instructions provide decision making capabilities and change the path taken by the program when executed in computer. These instructions specify conditions for altering the content of the program Counter. Some typical program control instructions are:

Mukesh Singh

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

3.9. RISC and CISC:

A. RISC

RISC is acronym for Reduced Instruction Set Computer. In this architecture a microprocessor is designed to perform a smaller number of types of instructions so that it can operate at a higher speed. Since each instruction type that a computer must perform requires additional transistors and circuitry, a larger list or set of computer instructions tends to make the microprocessor more complicated and slower in operation. That is why less instruction means fast in operation. Pipelining is one of the unique feature of RISC. It is performed by overlapping the execution of several instructions in a pipeline

Mukesh Singh SMC

fashion.

Example: Apple iPod and Nintendo DS etc

A RISC is a computer that can be divided into multiple instructions which perform low-level operation with single clock cycle, as its name suggest "Reduced instruction set"

6. CISC

CISC is acronym for Complex instruction set Computer. CISC architecture are designed to decrease the memory cost. The large programs need more storage, thus increasing the memory cost and large memory becomes more expensive.

To solve these problems, in CISC architecture, the number of instructions per program is reduced by embedding the number of operations in each instruction. Thereby the instructions are more complex.

Example: IBM 370/168, VAX 11/780, intel 80486 etc

A CISC is a computer where single instructions can execute several low-level operations or are capable of multi-step operations or addressing modes within single instructions, as its name suggest "Complex instruction set".

Mukesh Chaudhary

	RISC	CISC
1.	Simple instructions taking one cycle.	Complex instructions taking multiple cycle.
2.	Very few instructions refer memory.	Most of instructions may refer memory.
3.	Instructions are executed by hardware.	Instructions are executed by micro-program.
4.	Fixed format instructions.	Variable format instructions
5.	Few instructions.	Many instructions.
6.	Multiple register sets.	Single register set.
7.	Highly pipelined.	Not or less pipelined.
8.	Complexity is in Compiler.	Complexity is in micro-program.
9.	Eg: Apple ipod, Nintendo DS etc	Eg: IBM 370/168, Intel 80486

Mukesh Singh CMC

Understanding RISC & CISC architecture with example:

Let us take an example of multiplying two numbers:

A = A * B;

<<--- this is (statement)

The CISC Approach:

The primary goal of CISC architecture is to complete a task in as few lines of assembly as possible. This is achieved by building processor hardware that is capable of understanding & executing a series of operations, this is where our CISC architecture introduced.

for a particular task, a CISC processor would come prepared with a specific instruction (we'll call it "MULT"). When executed this instruction:

1.) Loads the two values into separate registers.
2) Multiplies the operands in the execution unit.
3) And finally third, stores the product in the appropriate register.

Thus, the entire task of multiplying two numbers can be completed with one instruction:

MULT A, B

<<--- this is assembly statement.

MULT is what is known as a "complex instruction". It operates directly on the computer's memory banks and does not require the programmer to explicitly call any loading or storing functions.

3.1

The RISC Approach:

RISC processors only use simple instructions that can be executed within one clock cycle. Thus, the "MULT" command described above could be divided into three separate commands:

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

In order to perform the exact series of steps described in the CISC approach, a programmer would need to code four lines of assembly:

```
LOAD R1, A
LOAD R2, B
PROD A, B
STORE R3, A
```

At first, this may seem like a much less efficient way of completing the operation. Because there are more lines of code, more RAM is needed to store the assembly level instructions. The compiler must also perform more work to convert a high-level language statement into code of this form.

Mukesh Singh SMC

3.10. 64- Bit Processor

A 64-bit processor is a microprocessor with a word size of 64 bits, a requirement for memory and data intensive applications such as computer-aided design (CAD) applications, database management systems (DBMS), technical and scientific applications, and high performance servers.

In simple words, 64-bit processors are computer chips that can support 64-bit computing. And by 64-bit computing, we mean software supports 64-bit virtual memory address.

Intel, IBM, Sun Microsystems, Hewlett Packard, and AMD currently develop or offer 64-bit processors.

Mukesh Singh's PC

Unit-4

Control Unit

121

4.1. Control Memory:

Control memory is type of RAM which contains addressable storage registers.

The function of control unit is to initiate sequences of micro-operations.

~~When control signals~~

→ When control signals are generated by hardware using conventional logic design techniques, the control unit is said to be hardwired.

→ A control unit whose binary control variables are stored in memory is called a microprogrammed control unit.

Control memory:

A memory that is a part of a control unit is referred to as a control memory.

- Each word in control memory contains within a microinstruction.
- A sequence of micro instructions constitutes a microprogram.

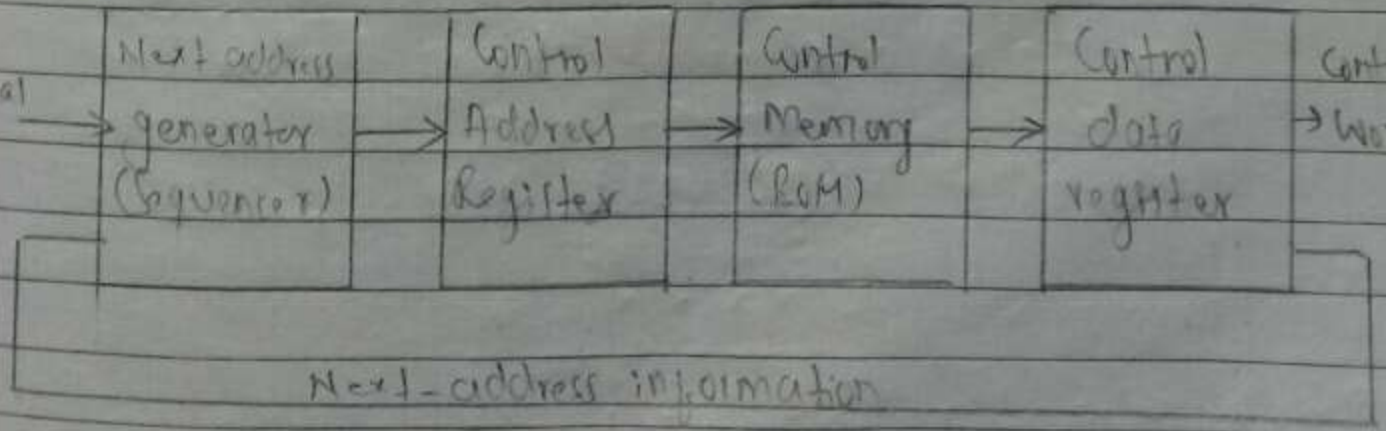


Fig: Microprogrammed Control Unit organization

7.2 Addressing Sequencing:

Each computer instruction has its own microprogram routine in control memory to generate the microoperations that execute the instructions.

Process of finding address of next micro-instruction to be executed is called address sequencing. ~~Address sequencer~~

Process of Address sequencing:

Address sequencer must have capabilities of finding address of next micro instruction in following situations:

1) Incrementing of the control address register
2) Unconditional or Conditional branch, depending on status bit conditions.
3) Mapping Process
4) A facility for subroutine call and return

Mukesh Singh SMC

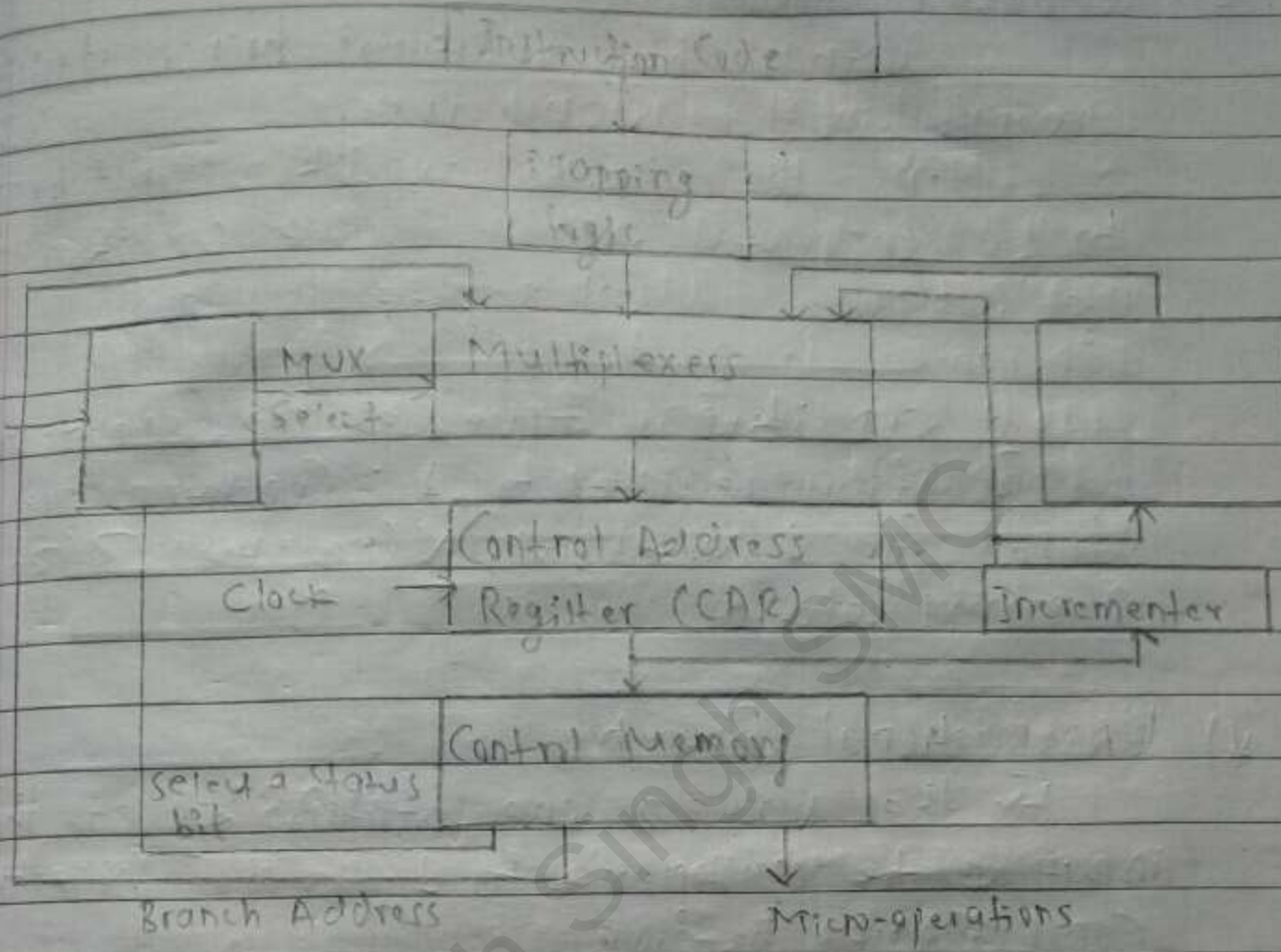


Fig: Block diagram of address Sequencer

OR

Selection of address for Control Memory

→ Control address register receives address of next micro-instruction from different sources.
→ Incrementer simply increments the address by one.
→ In case of branching, branch address is specified in one of the field of microinstruction.
→ In case of subroutine call, return address is stored in the register SBR which is used when returning from called Subroutine.

Mukesh Singh SMC

Conditional Branch:

Controls the conditional branch decisions generated in the Branch logic.

Tests the specified condition and branch to the indicated address if the condition is met, otherwise address register is simply incremented.

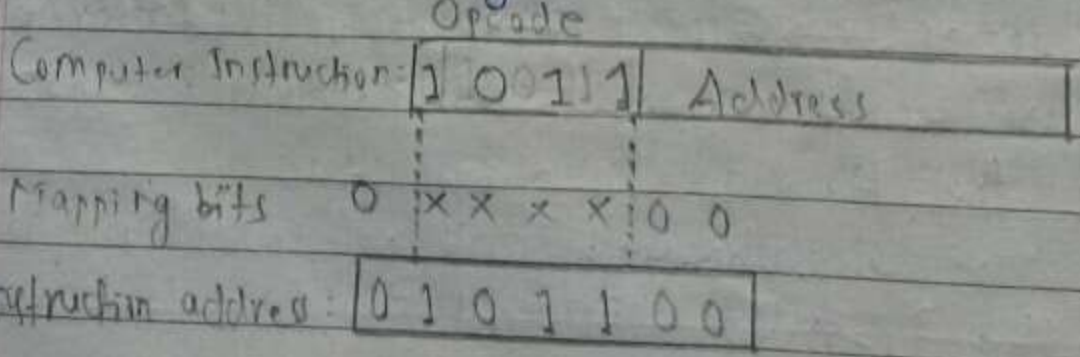
If condition is true, set the appropriate field of status register to 1. Conditions are tested for O (overflow), N (negative), Z (zero), C (carry), etc.

*) Unconditional Branch

Fix the value of one status bit at the input of the multiplexer to 1. So that branching can always be done.

*) Mapping of instructions:

One simple mapping process that converts the 4-bit operation code to a 7-bit address for control memory is shown below:



Mukesh Singh SMC

4-bit opcode = Specify up to 16 distinct instruction.

* Mapping Process: Converts the 4-bit opcode to a 7-bit control memory address
  → Place '0' in the most significant bit of address.
  → Transfer the 4-bit operation code bits.
  → Clear the last two significant bits of the control address register (CAR).

Mapping function: Implemented by mapping ROM or PLD.

Control Memory Size: 128 words (= 2)

* Subroutines:
  Subroutines are programs that are used by another program to accomplish a particular task.
  → Microinstructions can be saved by employing subroutines that use common sections of the micro-code.

Mukesh Singh SMC

4-3 Computer Configuration:

Fig: Computer hardware Configuration

Mukesh Singh SMC

4.1 Microinstruction format :

The microinstruction format for the control memory is shown in the figure below.

The 20 bits of the microinstructions are divided into four functional parts:

- The three fields F1, F2 and F3 specify micro-operations for the computer.
- The CD field selects status bit conditions.
- The BR field specifies the type of branch.
- The AD field contains a branch address.

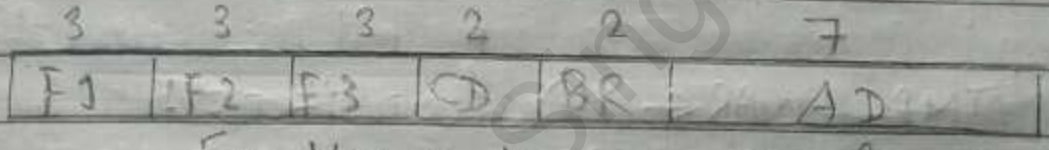


Fig: Microinstruction code format

F1, F2, F3 : Micro-operation fields
CD : Condition for branching
BR : Branch field
AD : Address field

Micro-instructions:

It is a single instruction in micro-code. Each microinstruction below is defined using register transfer statements and is assigned a symbol for use in symbolic microprogram.

Instructions that makes microprogrammed are called micro-instructions.

Mukesh Singh BVC

Description of CD:

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

Description of BR

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) ← PR(11-14), CAR(0,1,6) ← 0.

CD (condition field consists of two bits representing 4 status bits and BR (Branch) field (2-bits) used together with address field AD, to choose the address of the next micro instruction.

Mukesh Singh SMC

Microinstruction fields (f_1, f_2, f_3)

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

F2	Microoperation	Symbol
000	None	NOP
001	AC ← AC − DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0–10) ← PC	PCTDR

Mukesh Singh Jain

F3	Micro operation	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	—

Here micro-operations are subdivided into three fields of 3-bits each. These 3 bits are used to encode 7 different micro-operations. No more than 3 micro-operations can be chosen for a micro-instruction, one for each field. If fewer than 3 microoperations are used, one or more fields will contain 000 for no operation.

Mukesh Singh SMC

4.5 Symbolic Microinstruction:

Symbols are used in microinstructions as in assembly language. A symbolic microprogram can be translated into its binary equivalent by a microprogram assembler.

format of microinstruction:

Contains five fields → label; micro-ops; CD; BR; AD

Label: → May be empty or may specify a symbolic address terminated with a colon.

Micro-ops: → Consists of one, two or three symbols separated by commas.

CD: → one of {U, I, S, Z}

where, U: Unconditional Branch
I: Indirect address bit
S: Sign of AC
Z: Zero value in AC

BR → One of {JMP, CALL, RET, MAP}

AD → One of {Symbolic address, NEXT, empty (in case of MAP and RET)}

4.6 Symbolic Microprogram:

Fetch Routine:

- Read instruction from memory.
- Decode instruction and update PC.

Mukesh Singh SMS

Sequence of micro-operations in fetch routine/cycle:

AR ← PC
DR ← M[AR], PC ← PC+1
AR ← DR(0-10), CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

Symbolic microprogram for fetch routine/cycle:

```
        ORG 64
FETCH:  PC TAR      U  JMP  NEXT
        READ, INC PC U  JMP  NEXT
        DR TAR      U  MAP
```

Binary Microprogram for fetch routine:

Binary address	F1	F2	F3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	000101	00	00	1000010
1000010	101	000	000	00	11	0000000

Symbolic Microprogram

- Control Memory → 128 20-bit words
- first 64 words → Routines for 16 machine instructions
- Last 64 words → Used for other purpose (e.g. fetch routines & other subroutines)
- Mapping → OP-Code xxxx into 0xxxx00, the first address for 16 routines are 0(0000000), 4(0000100), 8, 12, 16, 20 ... 60.

Mukesh Singh SMC

Partial Symbolic Microprogram:

Label	Microoperations	CD	BR	AD
~~Add~~	ORG 0			
Add:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
	ORG 4			
BRANCH:	NOP	S	JMP	OVER
	NOP	U	JMP	FETCH
OVER:	NOP	I	CALL	INDRCT
	ARTPC	U	JMP	FETCH
	ORG 8			
STORE	NOP	I	CALL	INDRCT
	ACTDR	U	JMP	NEXT
	WRITE	U	JMP	FETCH
EXCHANGE:	ORG 12			
	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ACTDR, DRTAC	U	JMP	NEXT
	WRITE	U	JMP	FETCH

Mukesh Singh SMC

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

Binary Microprogram:

Micro Routine	Address Decimal	Address Binary	Binary Microinstruction F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000001
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
STORE	8	0001000	000	000	000	01	01	1000011
	9	0001001	000	101	000	00	00	0001010
	10	0001010	111	000	000	00	00	1000000
	11	0001011	000	000	000	00	00	1000000
EXCHANGE	12	0001100	000	000	000	01	01	1000011
	13	0001101	001	000	000	00	00	0001110
	14	0001110	101	101	000	00	00	0001111
	15	0001111	000	000	000	00	00	1000000

FETCH	64	1000000	110	000	000	00	00	1000001
	65	1000001	000	500	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDRCT	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

4.7. Control Unit Operation:

Mukesh Singh SMC

Unit-5

Computer Arithmetic

Page No. 55

~~5.1. Addition Algorithm~~

Computer arithmetic is a branch of computer engineering that deals with methods of representing integers and real values (eg. fixed- and floating-point numbers) in digital systems and efficient algorithms for manipulating such numbers by means of hardware circuits or software routines.

5.1. Addition Algorithm:

The standard algorithm for addition is a process that is normally used to solve multi-digit addition problems.

Algorithm

Step 1: Start the program.
Step 2: Initialize the contents of memory location ~~Step 2~~ in Ax & Bx registers
Step 3: Initialize the value 00 for CL registers.
Step 4: Add the value of Ax & Bx by using ADD mnemonics and store the result in a memory location.
Step 5: If there is carry, increment the value of CL register by 1, else move the contents of Ax to a location for result.
Step 6: Move the value of CL to a location for output

Mukesh Singh SMK

Step 7: Stop the program.

Program

```
Data Segment
a db 9h
b db 2h
c dw ?
data ends
    Code Segment
assume cs: code, ds: data
Start:
mov ax, data
mov ds, ax
mov al, a
mov bl, b
add al, bl
mov c, ax
int 3
Code ends
end start
```

Mukesh Singh SMC

5.2 Subtraction Algorithm

The algorithm that is currently taught for subtraction requires "regrouping", which has previously been called "borrowing". This algorithm is usually illustrated with manipulatives, in which 10 is exchanged for 10 ones and 100 is exchanged for 10 tens.

Algorithm

Step 1: Start the program.
Step 2: Initialize the contents of memory location in Ax & Bx registers.
Step 3: Initialize the value 00 for CL registers.
Step 4: Subtract the value of Ax & Bx by using SUB mnemonics and store the result in a memory location.
Step 5: If there is carry, increment the value of CL register by 1, else move the Contents of Ax to a location for result.
Step 6: Move the value of CL to a location for output.
Step 7: Stop the program.

Mukesh Singh SMC

Program for Subtraction

```
Data Segment
  a db  24h
  b db  13h
  c dw  ?
data ends
    Code Segment
assume cs:code, ds:data
Start:
mov ax, data
mov ds, ax
mov al, a
mov bl, b
sub al, bl
mov c, ax
int 3
Code ends
end start.
```

Program to input Character & print

```
!
.mode small
  .data
  .code
MOV AH, 1H
INT 21H
MOV DL, AL
MOV AH, 2H
INT 21H
End.
```

Mukesh Singh SMC

5.3 Multiplication Algorithm:

A multiplication algorithm is an algorithm (or method) to multiply two numbers. Depending on the size of the numbers, different algorithms are in use. Efficient multiplication algorithms have existed since the ~~development~~ advent of the decimal system.

Algorithm

Step 1: Start the program
Step 2: Initialize the contents for Ax & Bx from memory locations
Step 3: Multiple the Ax & Bx value using MUL mnemonics and store result in memory location.
Step 4: Move the contents of Ax and Dx to memory location to check the result
Step 5: Stop the program.

Mukesh Singh SMC

Program

```
Data Segment
a db 9h
b db 2h
c dw 9
cod data ends
Code segment
assume cs:code, ds:data
start:
mov ax, data
mov ds, ax
mov ax, 000h
mov bx, 000h
mov al, a
mov bl, b
mul b
mov c, ax
int 3
code ends
end start
```

Mukesh Singh SMC

5.4 Division Algorithm:

A division algorithm is an algorithm which, given two integers N & D, computes their quotient & or remainder, the result of division. Some are applied by hand, while others are employed by digital circuit designs and software.

Algorithm

Step 1: Start the program

Step 2: Initialize the contents for Ax & Bx from memory locations

Step 3: Multiple the Ax and Bx values using MUL mnemonics and store the result in a memory location.

Step 4: Move the Content of Ax and Dx to memory location to check the result.

Step 5: Stop the program

Mukesh Singh SMC

Program

```
Data Segment
a db 28h
b db 2h
c dw ?
data ends
Code Segment
assume CS:code, ds:data
Start:
mov ax, data
mov ds, ax
mov ax, 0000h
mov bx, 0000h
mov al, a
mov bl, b
div b
mov c, ax
int 3
Code ends
end start
```

Hello World Program

```
Data Segment
MSG DB "Hello World!!!$"
Ends
Code Segment
Assume CS: Code, ds: data
Start:
MOV AX, Data
MOV DS, AX
LEA DX, MSG
INT 21H
MOV AH, 4CH
INT 21H
End
End Starts
```

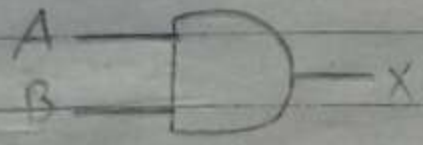
Mukesh Singh SMC

5.5 Logical Operations:

Gate level Logical Components:

i) AND

Symbol



VHDL Equation

X <= A and B

Or

X = A · B

Truth Table

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

iii) OR

Symbol

VHDL Equation

X <= A or B

Or

X = A + B

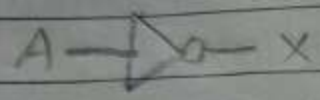
Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Mukesh Singh SMC

iii) NOT

Symbol



VHDL Equation

X <= not A

Or

X = A'

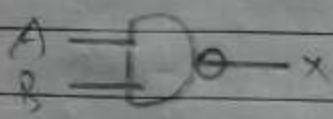
Truth Table

A	X
0	1
1	0

iv) Composite Logic Gates

iv) NAND

Symbol



VHDL Equation

~~X <= not (A and B)~~

X <= not (A and B)

Or

X = (A.B)'

Truth table

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

v) NOR

Symbol

VHDL Equation

X <= not (A or B)

Or

X = (A+B)'

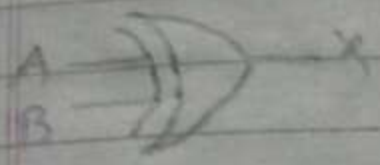
Truth Table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Mukesh Singh SMC

(ii) XOR

Symbol



VHDL Equation

X <= A xor B

Or

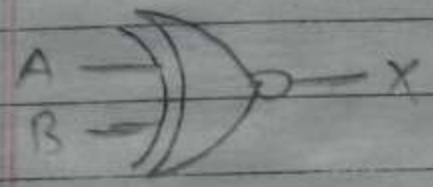
X = A ⊕ B

Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

(iii) X-NOR

Symbol



VHDL Equation

X <= A xnor B

Or

X = (A ⊕ B)'

Truth Table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

MukeshSinghSMC

Unit-6

Input/Output Organization

~~6. Microcomputer Memory~~:

6.1. Peripheral devices:

Peripheral devices are devices that are attached/connected to a computer system to enhance its capabilities. Peripherals devices includes input devices, output devices, storage devices and communication devices.

Examples - keyboard, mouse, printer, scanner, monitor, speaker, etc.

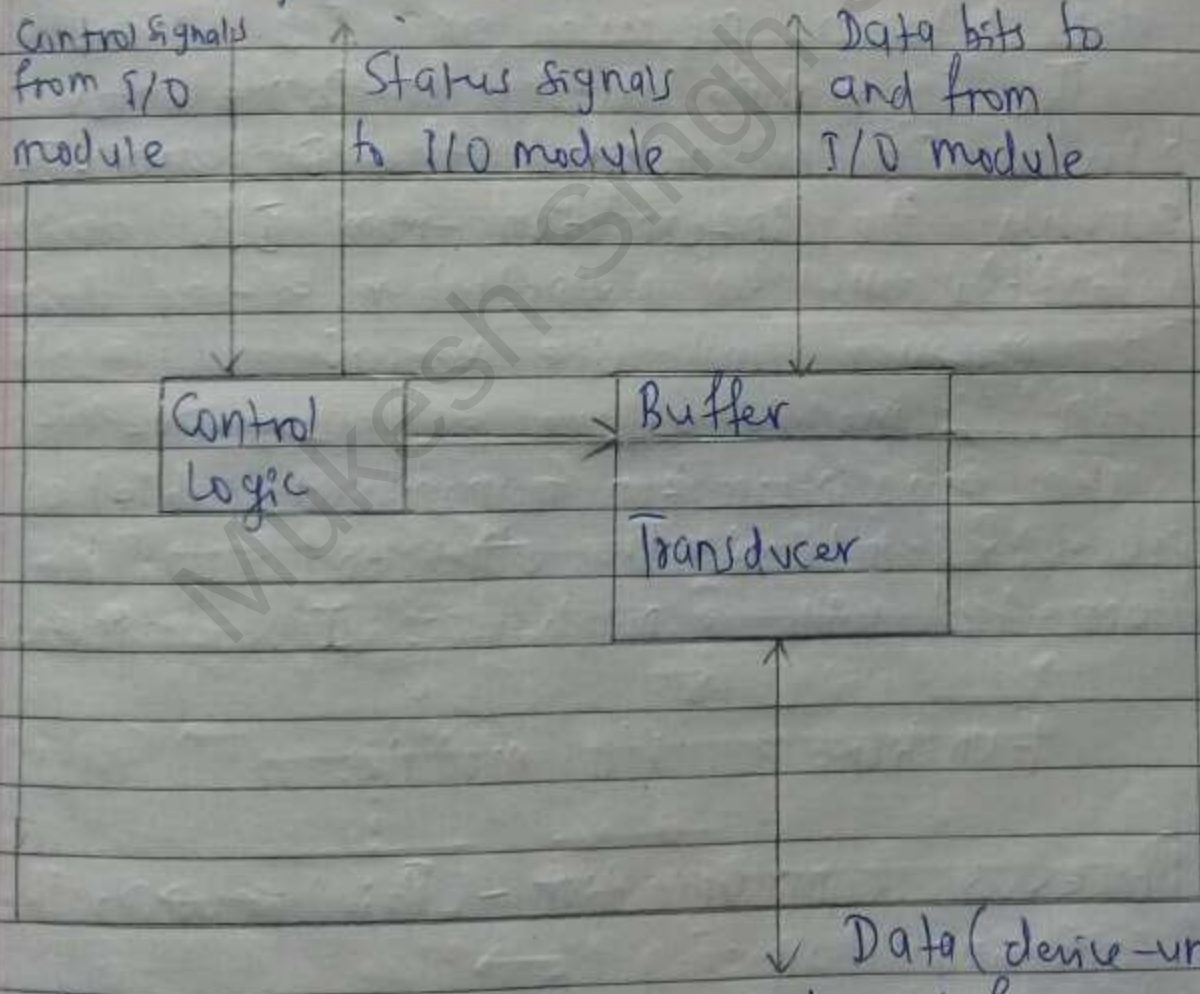


Fig: Block diagram of Peripheral device

Mukesh Singh SMC

6.2 I/O modules:

In computing, input/output or I/O module is the communication between an information processing system, such as a computer, and the outside world, possibly a human or another information processing system.

OR

I/O module is a technique for exchanging data between processor and I/O device.

The I/O module works as a mediator between I/O devices and the processor. It conveys the information from I/O devices (Sometimes called as peripheral or external device) to processor and vice versa.

An I/O module at one ends connected to the system bus (information transmission cable) of processor and at the other end may be connected with a number of I/O devices.

Functions of I/O module:

i) Processor Communication - This involves the following tasks:

a. exchange of data between processor and I/O modules.

b.

b. Command decoding - I/O module accepts

commands sent from the processor.

c. Status reporting → The device must be able to ~~perform device communication to~~ report its status to the processor.

d. Address recognition → Each I/O device has a unique address and the I/O module must recognize this address.

ii) Device Communication - The I/O device must be able to perform device communication such as status reporting.

iii) Control & Timing - The I/O module must be able to co-ordinate the flow of data between the internal resources and external devices.

iv) Data buffering - ~~This is necessary as there is a speed mismatch between~~ The data is buffered in the I/O module and then sent to the peripheral device at its rate.

v) Error detection - The I/O module must also be able to detect errors and report them to the processor.

Mukesh Singh SMC

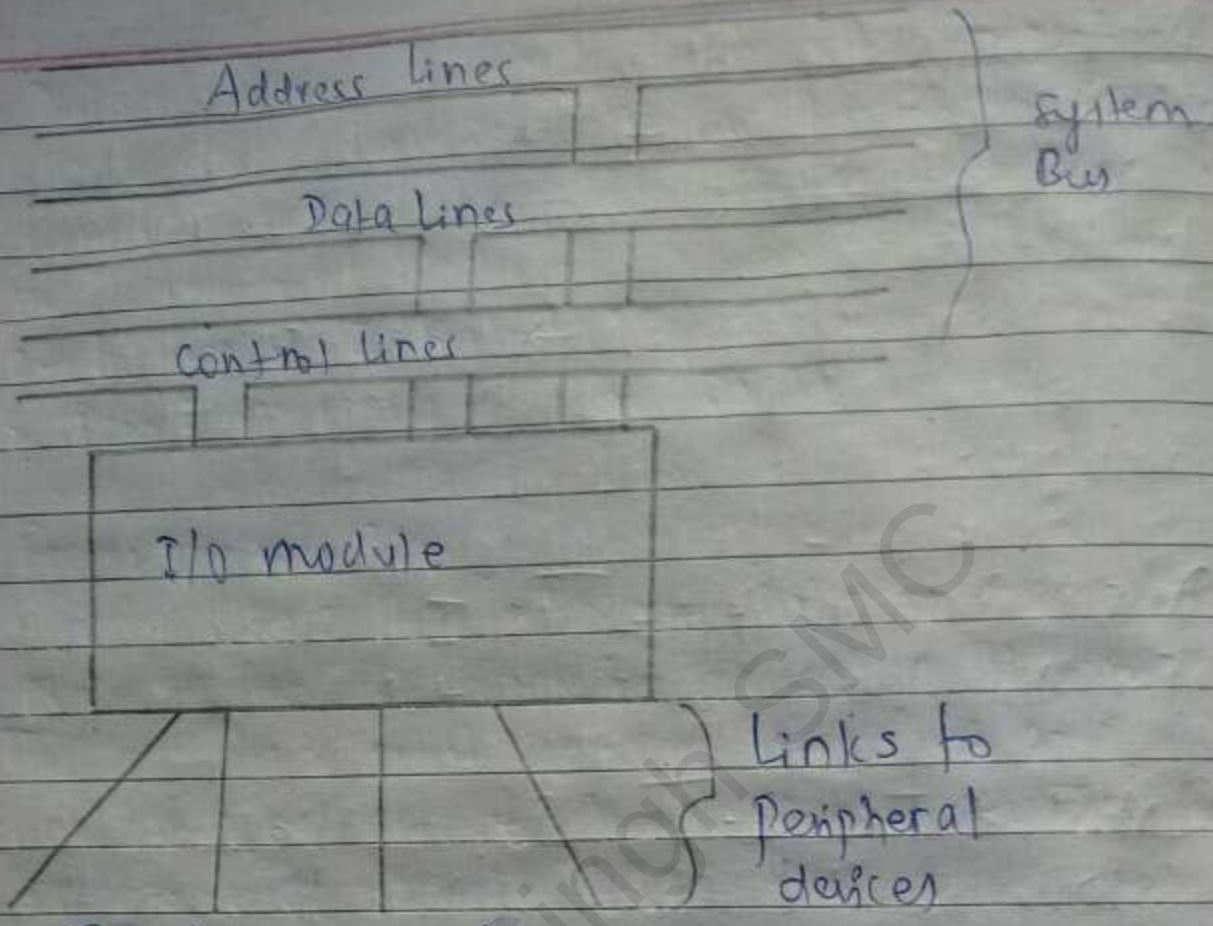


Fig: Model of I/O module

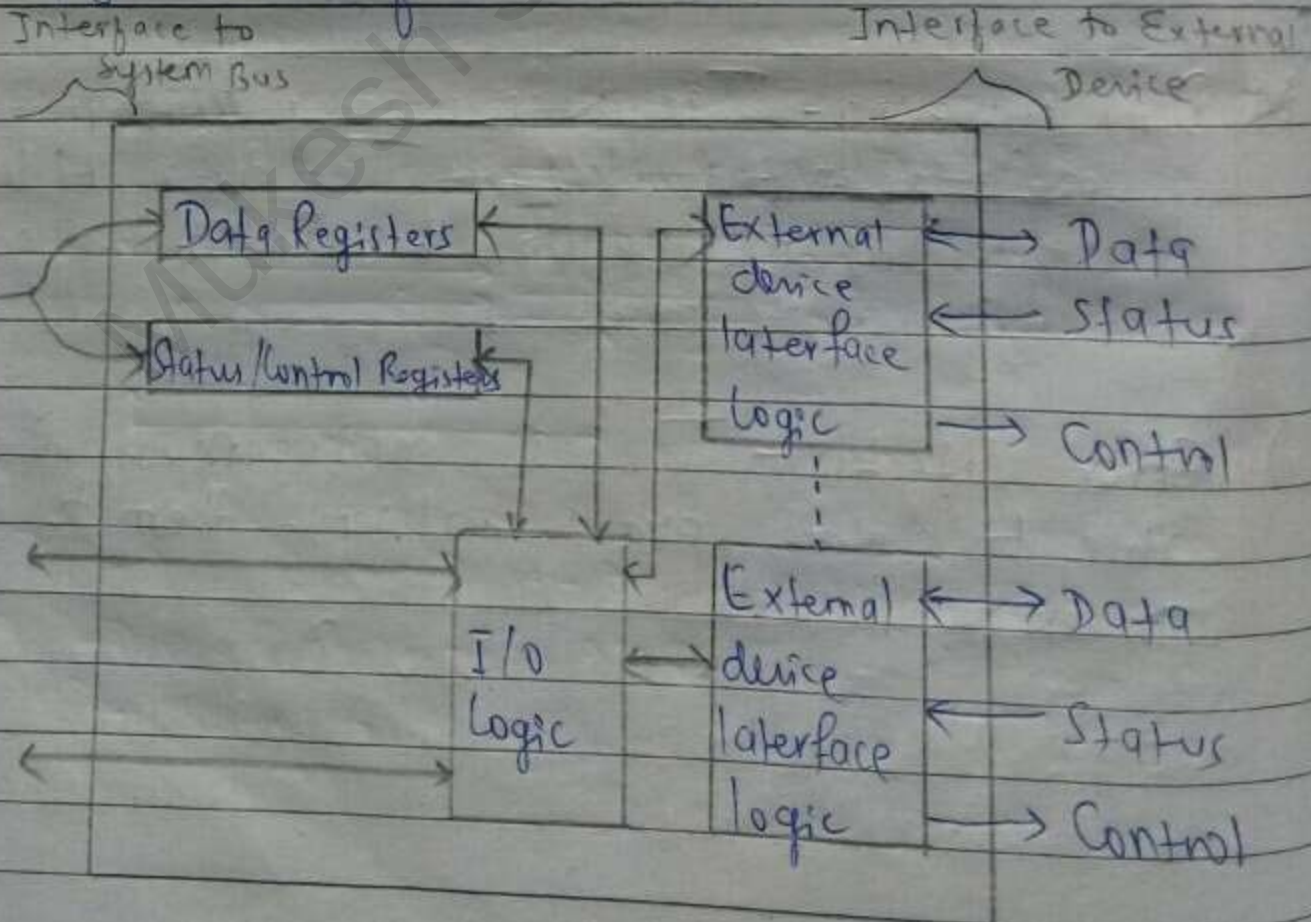


Fig: I/O module Diagram

Mukesh Singh SMC

6.3. Input/Output interface:

Input/Output interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit. The purpose of the communication link is to resolve the differences that exist between the ~~central computer~~ CPU and each peripheral.

OR

The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface. The CPU is interfaced using special communication links by the peripherals connected ~~key~~ to any computer system. These communication links are used to resolve the differences between CPU and peripheral.

6.4. Modes of transfer:

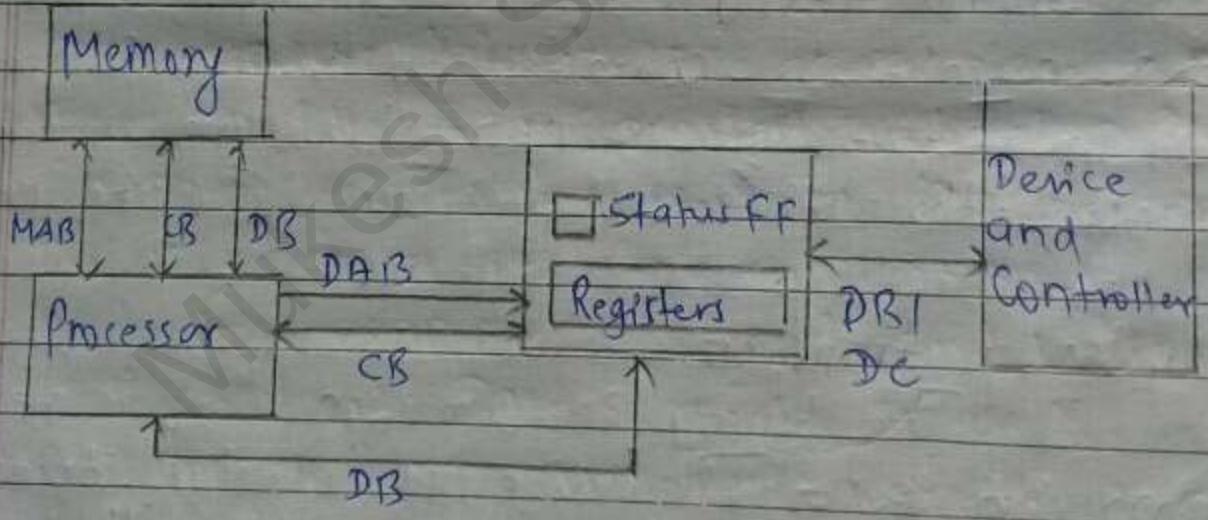
The binary information that is received from an external device is usually stored in the ~~more~~ memory unit. The information that is transferred from the CPU to the external device is originated from the memory unit. CPU merely processes the information but the source and target is always the memory unit. Data transfer between CPU and the I/O devices may be done in different modes.

Data transfer to and from the peripherals may be done in any of the three possible ways:

1. Programmed I/O
2. Interrupt-initiated I/O
3. Direct memory access (DMA).

6.5. Programmed I/O.

Using this technique, data transfer takes place under the direct control of the processor. The processor must continuously check an I/O device and hence it cannot do another task. This method is hence inefficient (slow).



DB - Data Bus DAB - Device Address Bus
DC - Device Control signals MAB - Memory Address Bus
CB - Control Bus

Mukesh Singhyal

Characteristics of Programmed I/O:

i) In programmed I/O, the I/O operations are completely controlled by the CPU.
ii) Used in real time and embedded systems.
iii) Used in CPUs ~~have~~ which have a single input and a single output instruction.

6.6. Interrupt-Driven Input/Output (I/O):

In the interrupt driven I/O, the processor issues a READ/WRITE instruction to the device and then continues doing its task. When the interface buffer is full, and it is ready to send data to the processor, the interface sends a signal to the processor informing it that data is ready. The signal is called as the interrupt signal. When the processor receives the interrupt ~~signal~~ signal, it knows that the data is ready; it suspends its current job and transfers data from buffer to its own register.

Role of CPU in interrupt-driven I/O:

i) The device issues an interrupt signal to the processor.
ii) The processor finishes ~~the~~ execution of the current instruction. It then responds to the interrupt signal.
iii) The processor sends an acknowledgement signal to the device that sent the interrupt. The device then removes ~~the~~ its interrupt signal.
iv) The processor then attends to the device ~~and~~ that issued the interrupt signal.

6.7. Direct Memory Access:

DMA is a method/technique of transferring data from main memory (RAM) to a device without passing it through the CPU. It is a technique to speed up data transfer without using the CPU.

While most data that is input or output from your computer is processed by the CPU, some data does not require processing, or can be processed by another device. In this situation, DMA can save processing time and is a more efficient way to move data from the computer's memory to other devices.

This means computers having DMA channels can transfer data to and from devices much more quickly than computers without a DMA channel can. This is useful for making quick backups and for real-time applications.

For example, a sound card, video cards that support DMA can also access the system memory and process graphics without needing the CPU. Ultra DMA hard drives use DMA to transfer data faster than previous hard drives that required the data to first be run through the CPU.

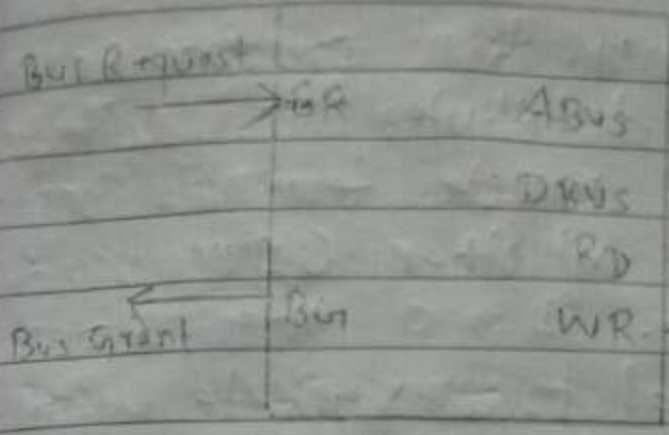


Fig: CPU Bus Signals for DMA Transfer

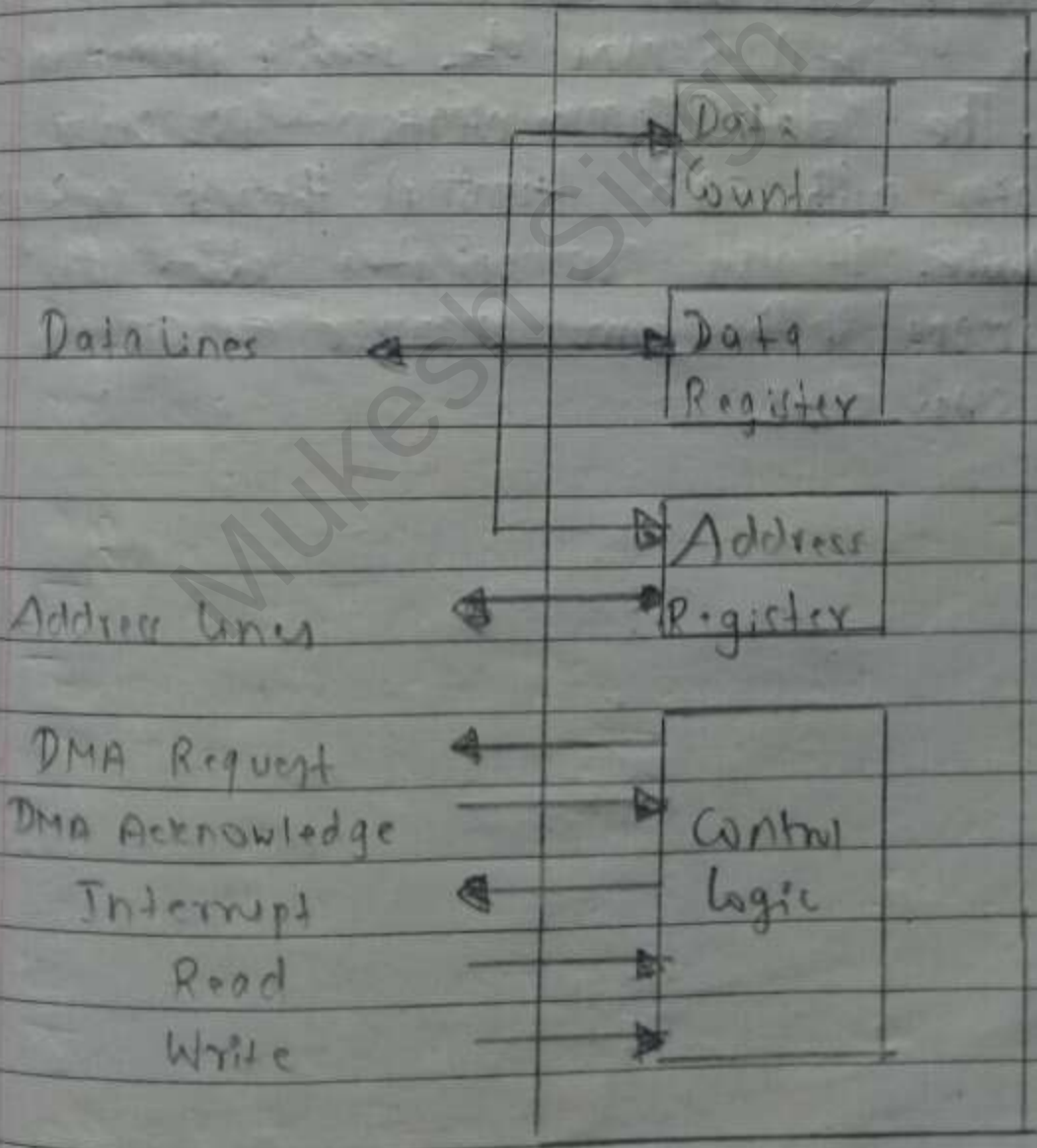


Fig: Block diagram of DMA

Mukesh Singh SMC

6.8 Data Communication Processor:

A data communication processor is an I/O processor that distributes and collects data from numerous remote terminals connected through telephone and other communication lines to the computer. It is a specialized I/O processor designed to communicate with data communication networks.

Such a communication network consist of variety of devices such as printer, display devices, digital sensors etc. serving many users at once. The data communication processor communicates with each terminal through a single pair of wire. It also communicates with CPU and memory in the same manner as any I/O processor does.

Mukesh Singh SMC

Unit-7
Memory Organization

Page No. 67
Date

Memory is the storage device of computer that stores data either temporarily or permanently.

Memory organization is a classification of memory or a class presentation of memory in which the memory is categorized or sub divided according to their work.

7.1. Microcomputer Memory:

A microcomputer is a small sized personal computer (PC) that is designed for an individual having a microprocessor inside it. A microcomputer contains a microprocessor (a CPU on microchip), memory in the form of read-only memory (ROM) and random access memory (RAM), I/O ports and a bus or system of interconnecting wires housed in a unit that is usually called a motherboard.

- Memory is essential component of microcomputer system.
- It stores binary instructions and datum for the microcomputer.
- The memory is the place where the computer holds current programs and data that are in use.
- None technology is optimal in satisfying the memory requirements for a computer system.

Nilesh Singh SMC

7.2 Characteristics of memory systems:

The memory system can be characterized with their following properties:

1. Location
2. Capacity
3. Unit of Transfer
4. Access method
5. Performance
6. Physical Type
7. Physical Characteristics
8. Organization

1. Location:

It deals with the location of memory device in the computer system. There are three possible locations:

- CPU: This is often in the form of CPU registers and small amount of cache.
- ~~DO~~
- Internal memory: This is main memory like RAM or ROM.
- External Memory: It consists of secondary storage devices like hard disk, magnetic tapes that are accessible to processor through controllers.

Mukesh Singh SMC

2. Capacity:
The capacity of any memory device is expressed in terms of :
. Word size: Words are expressed in bytes (8 bits). Commonly used word sizes are: 1 byte, 2 bytes and 4 bytes.
. Number of Words:- They specifies number of words available in the particular memory device. Common word lengths are 8, 16, 32 bits etc.

3. Unit of Transfer:
It is the maximum number of bits that can be read or written into the memory at a time.
. Internal Memory: Unit of transfer is equal to the word size.
. External Memory: They are transferred in blocks which is larger than a word.

4. Access Methods:
It is a fundamental characteristics of memory devices. It is the sequence or order in which memory can be accessed.
. Sequential Access: Access must be made in specific linear sequence.
Eg: Magnetic tape

Nilesh Singh SMC

- Direct Access: Access is accomplished by jumping (direct access) to general vicinity plus a sequential search to reach the final location.
  Eg: magnetic disk

- Random Access: Any location can be selected out randomly and directly addressed and accessed.
  e.g. RAM

- Associative Access: A word is retrieved based on a portion of its contents rather than its address.

5. Performance: The performance of the memory system is determined using three parameters:

- Access time: for random access memory, it is time taken to perform read/write operation.
  for non-random access memory, it is time taken to position read/write mechanism at desired location.
  - Time between presenting the address and getting the valid data.

- Memory Cycle time: It is the total time that is required to store next memory access operation from the previous memory access operation

Mukesh Singh SMC

. Transfer rate: It is defined as the rate which data can be transferred into or out of ~~Computer~~ memory unit.

6. Physical Type:
   . Semiconductor - RAM
   . Magnetic - Disk & Hard disk
   . Optical - CD & DVD.

7. Physical Characteristics:
   . Volatile - Information decays or is lost when power is off.
   . Non-volatile - Information ~~once~~ ~~data~~ is not lost when power is off.
   . Erasable - Data in memory can be erased are called erasable.
   . Non-erasable - Data once programmed cannot be erased or modified.

8. Organization:
   Organization refers to the physical arrangement of ~~words~~ bits to form words. It is not always obvious.

Mukesh Singh SMC

7.3 The Memory Hierarchy:

Memory hierarchy is a concept that is necessary for the CPU to be able to manipulate data. This is because it is only able to get instructions from cache memory.

Computer memory hierarchy is a program structure that is commonly used to illustrate the significant differences among memory types.

The memory hierarchy system consists of all storage devices employed in a computer system from slow but high capacity auxiliary memory to a relatively faster cache memory accessible to high speed processing logic.

The figure in the next page illustrates memory hierarchy:

Mukesh Singh SMC

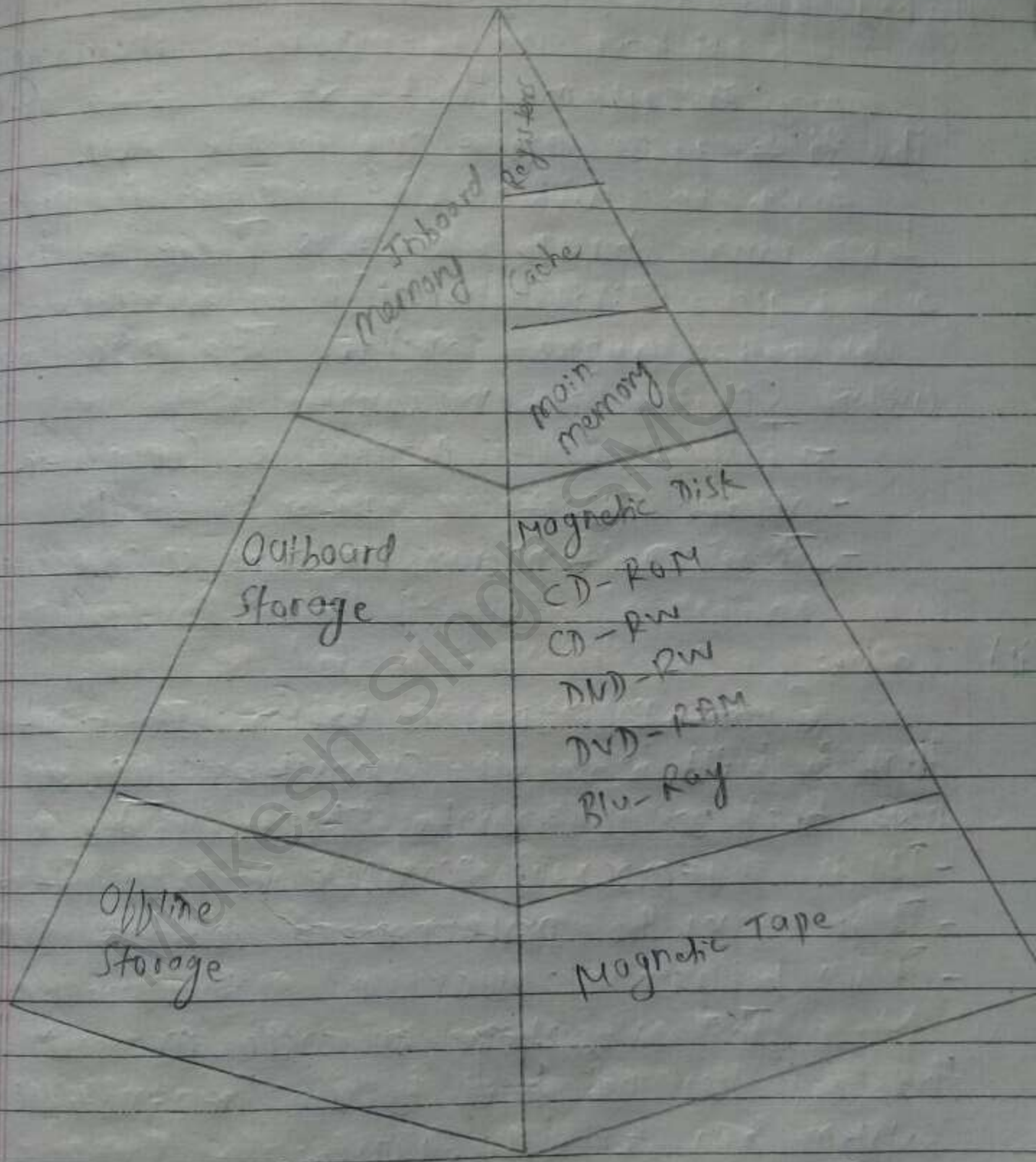


Fig: The Memory Hierarchy

Mukesh Singh SMC

i) Registers:

Registers are temporary memory of computer system that stores data during the time of processing inside ALU.

Registers within CPU are special purpose temporary storage locations. There are two types of registers: (a) General purpose register and (b) Specific purpose register. The important register within CPU is Program Counter (PC).

- They are fastest memory of computer.
- Storage capacity of register is small.
- They are temporary memory of computer.

ii) Cache:

Cache memory is extremely fast memory that is built into a computer's CPU, or located next to it on a separate chip. ~~The~~ a

- It is a temporary memory of computer.
- It increases the performance & speed of computer.
- It lies between processor and main memory.

The advantage of cache memory is that the CPU does not have to use the motherboard's system bus for data transfer.

Two processes in cache:

* Cache Hit ⇒ At first, processor send address of desired data to cache, if data is available then event is called Cache Hit.

Mukesh Singh SVC

* Cache Miss:→ When processor sent address is not present in cache, event is called cache miss. After cache miss, data of main memory is copied to cache for next cache hit.

Two main levels of cache:

* L1 Cache:→ L1 is "level-1" cache memory, usually built onto the microprocessor chip itself. for example, the Intel microprocessor comes with 32 thousand bytes of L1.
  - It is accessed without any delay.

* L2 Cache → L2 is "level-2" cache memory that is on a separate chip that can be accessed more quickly than the larger "main" memory". A popular L2 cache memory size is in terms of megabytes.
  - It takes more clock cycle to access than L1 cache.

iii) Main Memory:

Main memory is principal internal memory system of computer. It is fast and less capacity memory of computer.
- It is directly connected to the CPU.
- It is semiconductor based memory.
- It comprises mainly of RAM and small amount of ROM.
- These memories are located on motherboard.

Mukesh Singh SVC

iv) Magnetic Disk: and Optical Disk:

Magnetic disks are flat circular plates of metal or plastic, coated ~~with~~ on both sides with iron oxide.

A magnetic disk offers high storage capacity, reliability and fast to access data. It access data randomly. It is slower since it is not directly connected to CPU.

The types of magnetic disks are hard disk, floppy disk.

Optical Disk

An optical disk is data storage medium that can be written to and ~~read~~ read using a low powered laser beam. It is mainly used for music, movies, and software programs. It is compact, lightweight, durable. Optical disks are: CD-ROM, CD-RW, DVD, Blu-Ray disc etc.

v) Magnetic Tape: magnetic

A ~~memory~~ tape is sequential storage device of computer. It is not common storage device now a days. It is mainly used in mainframe and super computers. It is slow and sequential flow of data. It is very useful for storing huge volume of data economically.

A magnetic tape is a continuous strip of plastic

Mukesh Singh SMC

Advantages

- It is cheaper.
- It can be reused.
- Data from magnetic tap can be directly transferred onto hard disk.

Disadvantages

- It is sequential storage device.
- Data access time of magnetic tape is slow.

→ As we go down in the hierarchy:

- Cost per bit decreases.
- Capacity of memory increases.
- Access time increases.
- Frequency of access of memory by processor also decreases.

Mukesh Singh SMC

7.4 Internal and External Memory:

A. Internal Memory:

Internal memory is also called primary memory or main memory of computer. It is fast and less capacity of computer. They are mainly used to store data during time of processing. These memory have great effect on processing speed of computer.

There are mainly two types of primary or internal memory:

(1) RAM
(2) ROM

1. RAM

RAM stands for Random Access Memory. It stores data only when computer is on. It take the form of integrated circuits that allow stored data to be accessed in any order (that is random). The advantage of using RAM is to store whatever you are working on at any moment and RAM is very fast memory.

There are two types of RAM:

(i) SRAM
(ii) DRAM

Mukesh Singh SMC

	SRAM	DRAM
1.	Mostly, it is formed by using transistor.	1. Mostly, it is formed by using capacitors.
2.	It is expensive.	2. It is less expensive
3.	It has low density of data.	3. It has high density of data.
4.	It is faster than DRAM.	4. It is slower than SRAM.
5.	It does not need periodic refreshing.	5. It needs periodic refreshing.
6.	It is mainly used as cache memory.	6. It is mainly used as main memory.
7.	It consumes less electrical power.	7. It consumes more electrical power.

2. ROM:

ROM stands for Random Access Memory. It is a class of storage media used in computers. Data stored in ROM cannot be modified, or can be modified only slowly or with difficulty, so it is mainly in distributive firmware. The manufacturing company already installed program in ROM.

Mukesh Singh SMC

Types of ROM:

i) PROM:
PROM stands for Programmable Read Only Memory. It is the memory chip on which data can be written only once. It retains their contents when the computer is turned off. The difference between PROM and ROM is that a PROM is manufactured as blank memory where as ROM is programmed during the manufacturing process.

ii) EPROM:
EPROM stands for Erasable Programmable Read Only Memory. It is special type of memory that retains its contents until it is exposed to ultraviolet light. The ultraviolet light clear its contents making it possible to reprogram the memory.

iii) EEPROM:
EEPROM stands for Electrically Erasable Programmable Read Only Memory. It is a special type of ~~memory~~ PROM that can be erased by exposing to an electrical signal instead of UV-light. It can be erased and rewritten at byte level.

Mukesh Singh

RAM	ROM
1. It stands for Random Access Memory.	1. It stands for Read only memory.
2. Contents will be erased when power is off.	2. Contents won't be erased when power is off.
3. It usually has higher memory space than ROM.	3. It usually has less memory space than RAM.
4. It is less expensive.	4. It is more expensive
5. It is faster than ROM.	5. It is slower than RAM.
6. Its types are SRAM & DRAM.	6. Its types are PROM, EPROM and EEPROM.

B External Memory

External memory is also known as secondary memory or auxiliary memory. It is slower and cheaper form of memory. CPU does not access the secondary memory directly. The contents in it must be copied into the primary storage, RAM for CPU to process. Secondary memory device includes hard drive, floppy disks, CDs, DVDs.

Mukesh Singh Sir

Types of External Memory.

1) Magnetic Tape:
A magnetic tape is sequential storage device of computer. A magnetic tape is the strip of plastic coated with magnetic recording medium (i.e. magnetic oxide). Data can be recorded and read as a sequence of character through read/write head. It is not common storage nowadays. It is mainly used in mainframe and supercomputers. It is slow and sequential flow of data.

2) Magnetic Disk:
Magnetic disks are flat circular plates of metal or plastic, coated ~~with~~ on both side with iron oxide.
Magnetic disc offers high storage capacity, reliability and fast to access data. It access data randomly. It is slower since it is not directly connected to the CPU.
Types of ~~hard disk~~ magnetic disks:

i) Hard Disk:
The hard disk is non-volatile and random memory of a computer system. The hard disk has enormous storage capacity compared to

main memory. The hard disk is usually contained in the system unit of a computer. The hard disk is used for long term storage of programs and data. Hard disk has a storage capacity in terms of gigabytes such as 20GB, 40GB, 80GB, 320 GB etc.

ii) Floppy Disk:

Floppy disks are portable magnetic storage devices. Floppy disks are not as fast as hard disk, nor they store much information. They are mostly used for transferring software between computers. They are small, removable media storage devices. They record data on a thin, circular magnetic field encased in a flat, square plastic jacket.

iii) Optical Disk:

An optical disk is a data storage medium that can be written to and read using a low powered laser beam. It is mainly used for music, movies and software programs. It is compact, lightweight, durable. Optical disks are: CD-ROM, CD-RW, DVD-ROM, DVD-RW, Bluray disks etc.

4. Flash memory:

Flash memory is non-volatile data storage and thus is capable of retaining its data even

When its power source has been turned off. Flash memory is small, fast, lightweight and makes no noise. Flash memory is reliable and allows us to specify which data we want to keep. Example of flash memory are pendrive, memory-card etc.

7.5 Cache Memory Principles:

Principles

- ~~Intend to~~ Cache memory is intend to give memory speed approaching that of fastest memories available but with large size at the price of less expensive semiconductor memories.
- Cache is checked first for all memory references.
- If not found, the entire block in which that reference resides in main memory is stored in a cache slot, called a line.
- Each line includes a tag which identifies which particular block is being stored.
- Locality of reference implies that future reference will likely come from this block of memory, so that cache line will probably be utilized

Nikesh Singh SMC

repeatedly.

- The proportion of memory references, which are found already stored in cache, is called the hit ratio.

Cache memory is based on the principle of locality or locality of reference.

~~The locality of ref~~

According to locality of reference, if programs and data which are accessed frequently are placed in fast memory, the average access time can be reduced.

Locality of reference is divided into two categories:

i) Spatial locality:

It reflects tendency of a program to access data locations sequentially, such as when processing a table of data.

ii) Temporal locality:

It refers to the tendency of for a processor to access memory locations that have been used frequently.

Mukesh Singh-SMC

7.6. Elements of Cache Design:

7.6.1. Cache Size

The size of cache should be small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that overall average access time is close to that of the cache alone.

- Large cache are ~~smaller~~ slower than smaller ones.
- The available chip and board also limits cache size.

7.6.2 Mapping function:

The transformation of data from main memory to cache memory is referred to as memory mapping process.

As there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines.

There are three different types of mapping functions:

1) Direct Mapping:

The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line.

This technique is simple and inexpensive to use.

ii) Associate Mapping:

Associate mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache.

iii) Set-Associative Mapping

Set-associative mapping is a compromise that exhibits the strength of both direct and associative approaches. With set associative mapping, block can be mapped into any of the lines of set j.

7.6.3. Replacement Algorithm:

Once the cache has been filled, when a block is brought into the cache, one of the existing blocks must be replaced.

- for direct mapping, there is only one possible line for any particular block, and no choice is possible.
- for the associative and set associative techniques, a replacement algorithm is needed.
- To achieve high speed, such an algorithm must be implemented in hardware.

Last Recently Used (LRU), Least Frequently Used (LFU), first in first Out (FIFO) are some replacement algorithms.

7.6.4. Write Policy

· Write Through:

→ The simplest policy is called write through. Using this technique, all write operations are made to main memory as well as to the cache, ensuring that main memory is always valid.

· Write back:

An alternative technique, known as write back, minimizes memory writes. With write back updates are made only in the cache. When an update occurs, a dirty bit, or use bit associated with the line is set. Then, when a block is replaced, it is written back to main memory. if and only if the dirty bit is set.

7.6.5 Number of Caches:

1. Multilevel Caches: L1 and L2 cache

Most contemporary design includes both on-chip and external caches. The simplest such organization is known as two-level caches with the internal cache (on-chip) designed as level 1 (L1) and the external cache (off-chip) designated as level 2 (L2). There can also be 3 or more levels of cache. This help in reducing main memory accesses.

B. Unified and Split Cache:

- Earlier on-chip cache designs consisted of a single cache used to store both instructions and data. This is the unified approach.

More recently, it has become common to split the cache into two: one dedicated to instructions and other dedicated to data. These two caches exist at the same level. This is the split cache.

Using a unified cache or a split cache is another design issue.

Mukesh Singh SMC