INTRODUCTION.

1) Micro ~~chip~~ processor: Ex: Computers
2) micro computer: Used for general purpose
3) micro controller: Ex: Washing machine

Unit-I INTRODUCTION TO ARCHITECTURE OF MICROPROCESSOR

Definition: A up is a semiconductor device consisting of electronic logic circuits, manufactured either by a) LSI (Laorge scale Integration)
b) VLSI (Very large " ") techniques.
accepts data from input devices, process data according to instructions stored in memory and gives the results to the external world.

⇒ The up is a multipurpose programmable clock driven, register based electronic device that accepts binary instructions from memory, reads binary data as i/p and process data according to these instructions and provides result as o/p.

Historical Background of up:

- Mechanical Age (Abacus to Punched cards)
- Electrical Age (1970- programmable Pinched cards
  1946 - ENIAC [Electronic numerical Integrator and calculator]
  1946 - EDVAC [Electronic discrete Variable Automatic computer]
  1949 - EDSAC [Electron delay storage automatic calculator]

- UNIVAC [Universal Automatic Computer]

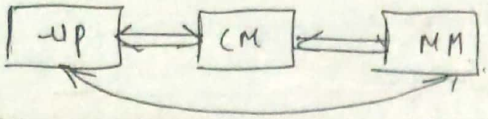
* µp Age / Electronic Age

- INTEL-4004 (1971)
  - → 4-bit µp (Nibble)
- INTEL-8080, 8085
  - → 8-bit µp, 8-bit data lines, 16-bit address lin
  - → 2^{16} - 64 kb Addressable memory space.
  - → CISC [complex Instruction Set Computer]
- Z-80, Zilog-Z8
- motorola, MC-6800

* Modern microprocessor Age (1978)

- INTEL-8086
  - → 16-bit µp, 16-bit data length, 20-bit address lin
  - → 2^{20} ⟶ 1 mb. Address of memory space.
- MC-68000 motorola
- MC-68000C (advanced)
- INTEL-80286 (P2) (1983)
  - w.r.t → 16-bit datalines, 16-bit processor, 16 mb adress memory space.
  - → INTEL-80287 - co-processor.
- INTEL-P3 - 80386 (1986)
  - → 32-bit processor, 32-bit data lines, 32-bit address l
  - → 4-GB addressable memory space - 2^{32}
  - 2^{32}
  - → 80387 → co-processor.
  - → Supports graphical user interface (GUI)
- INTEL-80486 µp (1989) (P4)
  - → 32-bit µp,
  - → 32-data lines,

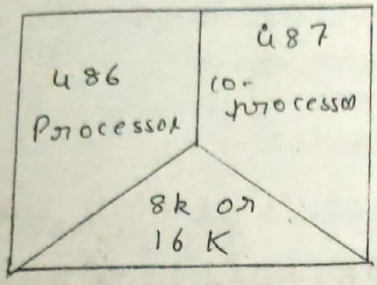
→ 8KB Cache Memory (4GB + 8KB cache memory)



→ It uses 80487 co-processor.

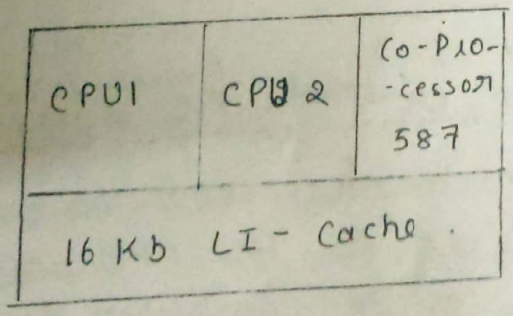
→ 80486DX processor.

→ L1 - 16KB cache memory.



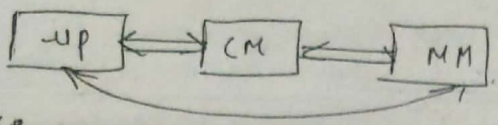
L2-Cache.

- **Pentium up (1993)**
  - → 80586 up.
  - * It has dual processors
  - * Executes 2-instructions per clock.
  - * 64-bit up, 64-data lines, 4-GB addressable memory space.
  - * 16 KB - L1 cache.
  - * High frequency upto 100MHz (mega) (double clock).



- **Pentium Pro Processor (P6)**
  - → Has three processors, improves parallel computing (3 instructions per cycle).
  - → 64-bit data lines
  - → 64-GB addressable memory space.
  - → 16-KB L1 cache.

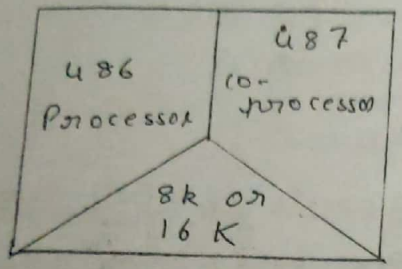
→ 8KB Cache Memory (4GB + 8KB cache memory)



→ It uses 80487 co-processor.

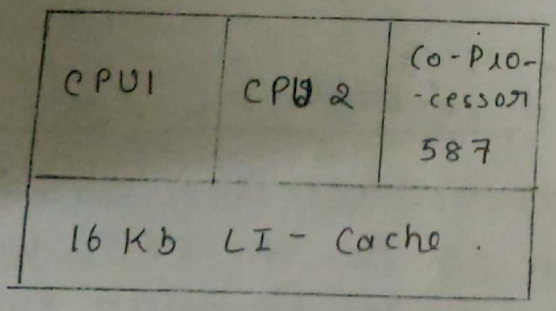
→ 80486 DX processor.

→ L1 - 16KB cache memory.



L2 - Cache.

- **Pentium up (1993)**
  - → 80586 up.
  - * It has dual processors
  - * Executes 2-instructions per clock.
  - * 64-bit up, 64-data lines, 4-GB addressable memory space.
  - * 16 KB - L1 cache.
  - * High frequency upto 100MHz (mega) (double clock).



- **Pentium Pro Processor (P6)**
  - → Has three processors, improves parallel computing (3 instructions per cycle).
  - → 64-bit data lines
  - → 64-GB addressable memory space.
  - → 16-KB L1 cache.

256 KB L2-cache.
→ High frequency upto 150MHz

FFFFh
0000h } 64K
2^{16}
16 Address lines

CPU 1	CPU 2	CPU 3	Co-processor 587
16 Kb L1-cache			
256 Kb L2-cache.			

[Address is unidirectional]

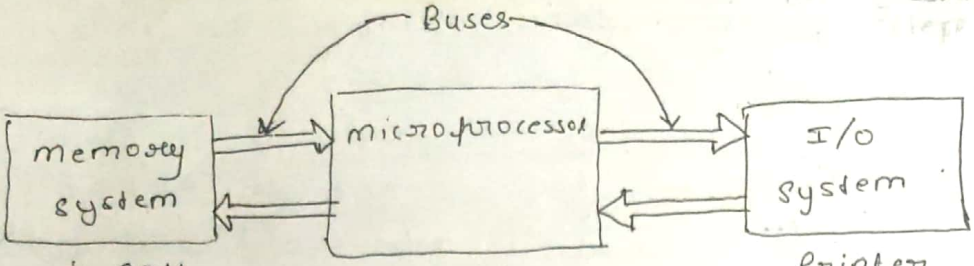
- Pentium - II - (1997)
  - → Pentium - Xeon.
  - → 64-bit datalines, 3 CPU's, 64 GB addressable space
    32-KB L1-cache, 256 KB L2-cache
  - → 350 MHz
  - → Xeon series are used in servers.
  - ⇒
- Pentium - III, Pentium - IV & Core-2, microprocessor,
  - → 3 CPUs, 64 bit-datalines, 32 KB L1 cache.
  - → 256 KB or 512 KB L2-cache,
  - ⇒ Execution of multiple instructions (multithread
- Future up.s Itanium Series up
  - → i- series.
  - → Intel + HP

* Microprocessor based ~~control~~ P.C system:

The heart of microprocessor based comp system is the up referred to as the CPU, which is the controlling element in an computer system.

It controls memory and I/O through a ser of connections called buses.

The block diagram of a up based PC syste can be written as.

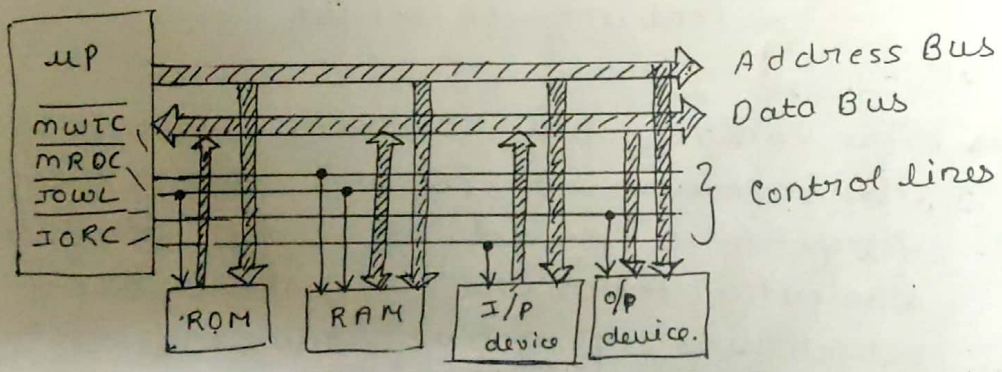


Dynamic RAM
static RAM
Cache
ROM
EEPROM
SDRAM
DDR RAM

8086
186
286
.....
Pentium series
.... core-2
i-series

Printer
Serial communication.
Floppy drive
Hard disk drive
Mouse, keyboard
monitor
CDROM
Plotter, scanner, etc

The block diagram of computer system showing the address, data and control bus structure can be written as



MWTC – memory Write Control
MRDC – memory Read control
IOWL – I/O write line.
IORL – I/O read line.

The microprocessor performs three main tasks for PC based system.

1) Data Transfer operations
2) Airthmetic and logical operations
3) programme flow via simple decisions

Address bus : The address bus requires a memory location from memory or an I/O location from the I/O devices. If I/O is addressed the address bus contains a 16-bit I/O address from 0000h to FFFFh (8086 2^{16} 2^{20}). If memory is addressed the address bus contains a memory address which varies in width with different versions of μps. (Ex: 8086 μp), 20 address lines, 2^{20} → 1MB addressable memory space) 00000h to FFFFFh

Data bus : It transfers information between the μp & it its memory or I/O. Data transfer vary in size from 8-bits to 128 bits wide.

Ex: INTEL 8085 — 8-bit μp
8086 — 16-bit μp
Pentium — 64-bit μp

Control bus : The control bus contains lines that select the memory or I/O and cause them to perform a read or write operation. Basically there are 4 control bus connection which are in active low are ~~MWTC~~

i) MWTC - Memory Write Control
ii) MRDC - Memory Read Control
iii) IOWC - I/O Write Control
iv) IORC - I/O Read Control.

* Operation of μp

→ The μp reads the contents of a memory location by sending the memory an address through ~~or~~ address bus.
→ It sends the memory control signal (MRDC) to cause the memory to read data
→ The data read from the memory are passed to the μp through data bus.

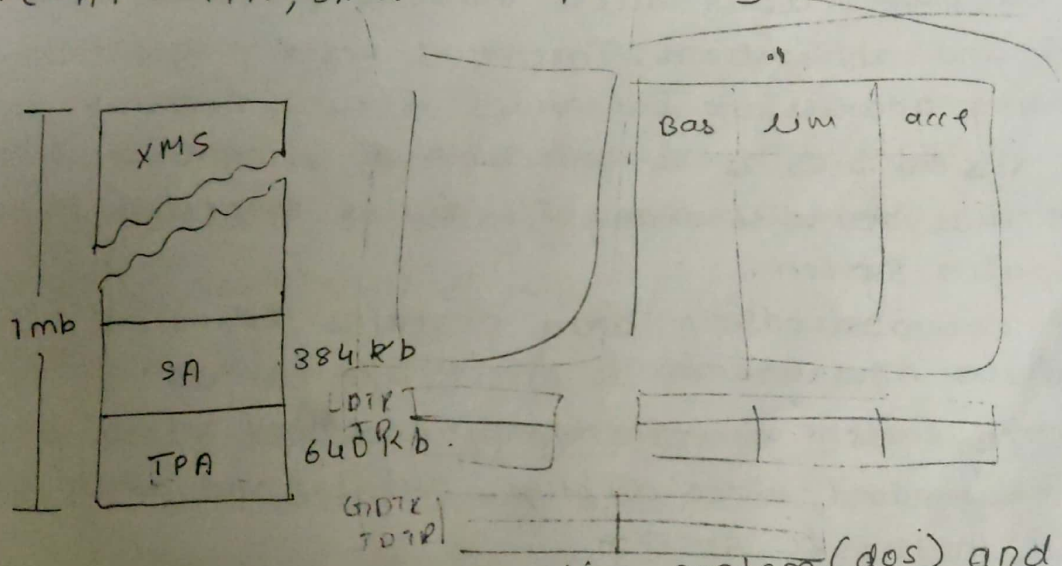
→ If a memory write, I/O write or I/O read occurs, the same sequence repeats [step 1, 2, 3] repeats except that different control signals are issued and the data flow out of the µP through its data bus for a write operation.

The memory and I/O System

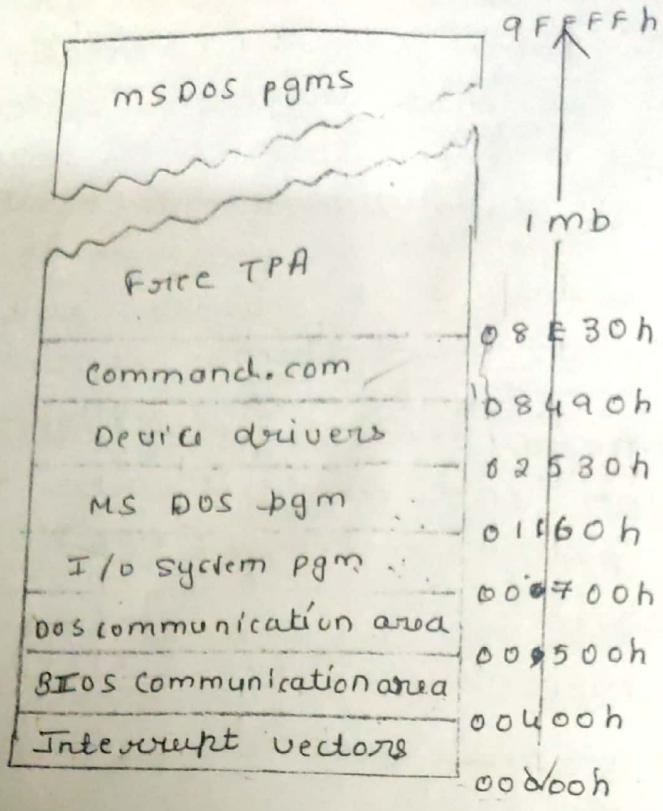
The memory structure of all INTEL based PCs is similar and divided into 3 main parts

1. Transient Program Area (TPA)
2. System Area (SA)
3. Extended memory System (XMS)

[PC-XT – TPA & SA → present
NO XMS.
PC-AT – TPA, SA & XMS present]



TPA :- It holds the disc operating system (dos) and other programs that controls the computer system. The TPA is a dos concept and not really applicable in windows. The TPA also stores any currently active or inactive dos application programs. The length of TPA is 640 Kb. The memory map of TPA can be written as in a PC based system.



i) The interrupt vectors access various features of dos, BIOS and applications. Interrupt vectors holds the starting address of Interrupt service Routines.

ii) The system bias is a collection of programs stored in ROM that operates many I/O devices connected to computer system.

iii) DOS communication area contains transient data used by programs to access I/O devices

iv) The I/O system contains programs that allow DOS to use keyboard, video display, printer and other I/O devices found in the PC system.

v) The command.com [comand processor] controls the operation of the computer from the keyboard when operated in DOS mode. It processes the DOS command If the command.com program is erased the PC cannot be used from the keyboard in DOS mode.

Never delete

command.com
IO.sys
MSDOS.sys

} to make work for other s/w, your computer will never run.

PROGRAMMING MODEL

The architecture of a up describes the internal configuration and operating functions of the individual components (registers) of up.

The programming model of 8086 through core-2 is considered to be program visible, because its registers are used during application programming and are specified by the instructions. Other registers are to be considered program invisible, because they are not addressable directly during application programming, but may be used indirectly during system programming.

Only 80286 and above contain the program invisible registers used to control and operate protected memory (more than 1 Mb)

⇒ PROGRAMMING MODEL ... 0

can be written as.

64-bit Regs	32-bit Regs	16-bit Regs	
RAX	EAX	AH	AL
RBX	EBX	BH	BL
RCX	ECX	CH	CL
RDX	EDX	DH	DL
RBP	EBP	BP	
RSI	ESI	SI	
RDI	EDI	DI	
RSP	ESP	SP	

General Purpose Registers (GPR's)

AX, BX, CX, DX

GPRs

R8			
⋮		⋮	
R14			
R15			

FLAG Register

RFLAGS	EFLAGS	FLAGS

Instruction pointer

RIP	EIP	IP

← 16-bit →

Segment Registers

CS
DS
ES
SS
FS
GS

8-bits – Byte - B
16-bits – Word - W
32-bits – Double - D
64-bits – Quad - Q

Eg:

```
• DATA      byte
OP1  DB  05h
OP2  DB  03h
     |
    Data
```

- 8086 through 80286 contain 16-bit architecture,
- 80386 through Core-2 contain 32-bit to 64-bit archite

* AX

8-bit, 8-bit

AX → AH, AL

BX → BH, BL

CX → CH, CL

DX → DH, DL

General purpose or multipurpose registers.

1) RAX → EAX → AX → AH, AL
64 32 16 8 8

Accumulator — * Generally used to store data temporarily
* Specially used to perform airthmethic and logical operations and also for some adjustment operations.
(decimal adjust and ASCII adjust)

2) RBX → EBX → BX → BH, BL (Base Register)

Base — * Generally used to store data temporarily during computations.
* Specially used to hold the offset address of a location in the memory.

3) RCX → ECX → CX → CH, CL

Count — * Generally used to store data temporarily
* Specially used as a count register along with branch instructions [(REP - Repeat) & LOOP]

4) RDX → EDX → DX → DH, DL

Data — * Generally used to store data temporarily during computations.
* Specially used to hold memory data during I/O transformations.

5) [IN AX, DX
OUT DX, AX]

5) RBP → EBP → BP (only 16-bit cannot be written as 8)

Base pointer- * GPR - General purpose register.
* points to a memory location for memory data transfer.
*

6) RDI → EDI → DI (Destination Index)

DI — * GPR
* Used to address string destination data during string manupulation instructions.
*

7) RSI → ESI — SI (Source Index)

SI — * GPR
* Used to address string source data during string manupulation instructions.

8) RSP → ESP → SP [Stack pointer)

SP — * special purpose register.
* It addresses an area of memory called the
* The stack memory stores data through this po

9) R8 through R15

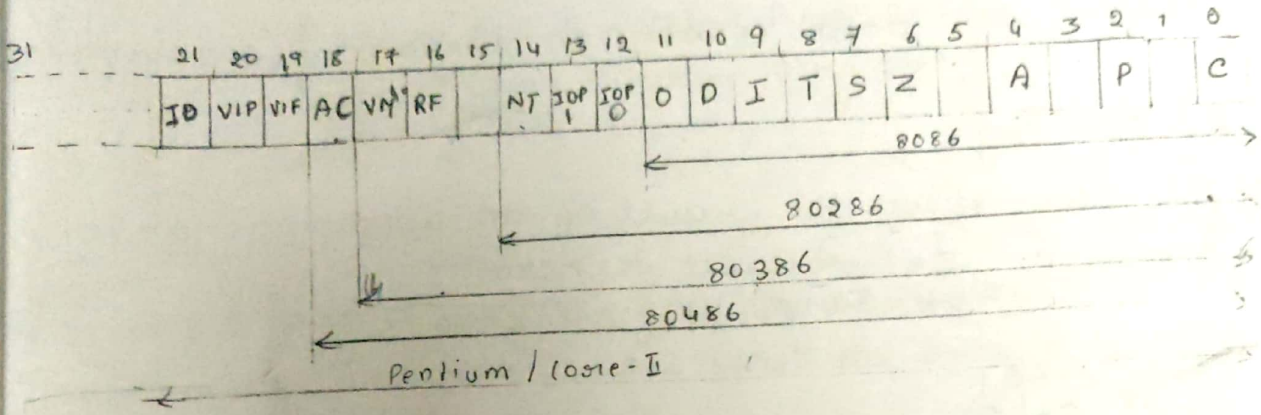
* GPR (Found in Pentium-4 and Core-2 ups only]
*

10) ~~FLAGS~~ RIP → EIP → IP (Instruction Pointer).

* Special Purpose Registers.
* It addresses the next instruction in a section of memory defined as a code segment.
* It finds the next sequential instruction in a program located within the code segment.
* It can be modified with a jump (JMP) or call (CALL instructions.

RFLAGS → EFLAGS → FLAGS (Flag register or Status register)

* Flags indicate the condition of the up. and control its operation.
* The contents of the flag register reflects the results of the up.
* The bit position / format of flag register can be written as.



- The rightmost 5 flags and overflow flag are chosed by most arithmetic and logic operations. [status registers]
- Data transfer operations do not change flags.

MOV POP XCHG PUSH

⇒ Flags:

i) C- Carry flag:

- This flag holds the carry after addition or the borrow of the subtraction.
  1 → carry generated
  0 → No carry

i) P- Parity flag:

- 0 → for odd parity
  1 → for even parity
  [No. of 1's in the result.]
- used to check the validity of data.

iii) A – Auxiallary Carry:

•) This flag holds the carry (half carry) after addition or the borrow after subtraction b/w lower four bits (Lower nibble) of the data.

Ex: 1100 (1001) ← lower nibble
0101 (1011)

0

1-carry

[∴ A=1]

•) Used for data conversion operations (BCD operati

iv) Z - Zero flag

•) This flag shows the result of an airthmetic or logi operation. Z=1 → Result is zero
Z=0 → Result is not zero

1010 1001
0101 0110

AND 0000 0000 ∴ Z=1
op

v) S – Sign Flag:

•) This flag holds the airthmetic sign of the result after an airthmetic or logic operation.
S=1 → Result is -ve.
S=0 → Result is +ve.

vi) T - Trap bit (Highest priority Interrupt)

•) If T=1, enables on chip debugging feature.
If T=0, debugging feature disabled. (∴

vi) I – Interrupt flag

•) This flag controls the operation of an interrupt. (INTR pin) [Interrupt request]
If I=1, INTR enabled (allows external interrupt)
STI (set interrupt)
If I=0, INTR disabled
CLI (clear interrupt)

vii) D - Direction flag.

•) This flag selects either the increment or decrement mode for the DI (Destination Index) and SI (Source Index) registers during string instructions.

•) If D=1, the registers are automatically decremented. (top to bottom)

If D=0, the registers are automatically incremented. (bottom to top)

[STD → set Direction Flag D=1]
[CLD → Clear Direction Flag D=0]

viii) O - Overflow Flag

•) Overflow occurs when signed numbers are added or subtracted. This flag is set when the results exceeds the capacity of the machine.

Ex: ① 1111 + 1111 = 0 (overflow)

•) For unsigned operation. this flag is ignored.

) IOP → IOPL (Input Output Privileged level)

•) It is used in protected mode to select the privilege level of I/O devices

•) 0 0 → Highest priority
0 1 –
1 0 –
1 1 → lowest priority

NT - Nested Task.

)This flag indicates that current task is nested within another task in protedted mode operation

) RF - Resume Flag.

•) This flag is used with debugging to control the resumption of execution. after the next instruction.

xii) VM - Virtual mode.

-) ~~when VM=1~~ This flag selects Virtual mode ope in a protected mode system.
-) It allows system program to execute multiple OS programs and not application program (d
-) It is used to simulate dos in the modern windo environment.

xii) AC - Alignment check.

-) This flag activates if a word or double word is addressed on a non-word or non-double word boundary.

xiii) VIF - Virtual Interrupt Flag.

-) It is a copy of the interrupt flag in protected mod system available to the pentium processors.

xx) VIP - Virtual Interrupt Pending

-) This flag provides info about virtual mode inte
-) It is used in mutlitasking environments to pro the O.S with VIF. and interrupt pending info.

xx) ID - Identification

-) This flag indicates that the pentium up's suppor CPUID instruction. This instruction provides th
-) ~~It~~ information about the up version / manufa

Segment Registers:

- These registers generate the physical memory addresses when combined with other registe (offset registers).

1) Code Segment (CS)

+ It is a section of memory that holds the c (programs & procedures) used by the up.

* This register defines the starting address of the code section (upper 16-bit).
* In real mode it defines the start of 64 Kb section of memory. (64 Kb - DOS)
* In protected mode, it selects a descripter that describes starting address and length of a section of memory holding code. (4 GB - windows)

) DATA SEGMENT (DS)

* It is a section of memory that contains most data used by a program.
* Data are accessed in the data segment by an offset address or the contents of other registers that holds the offset address.

) STACK SEGMENT (SS)

* It defines the area of memory used for the stack.
* The stack entry point is determined by the stack segment (SS) and SP or BP register.

EXTRA SEGMENT (ES)

* It is an additional data segment used by some of the string instructions to hold destination data

) FS & GS (Supplement / general).

* These are supplemental segment registers available in 80386 through core-2 cpu, to allow additional memory segments for data

6) DEFAULT SEGMENT AND OFFSET REGISTERS

Segment	offset	Purpose
CS	IP	Instruction/code add
SS	SP/BP	Stack address
DS	BX, SI, DI & 8-bit . 16-bit displacement	Data address
ES	DI	String destinatio

Eg 1) CS = 1400h
IP = 2300h
PA = CS + IP
= 1400h
2300h
PA 16300h

2) SS = 2000h
BP = 3000h
PA = SS + BP
= 2000h
3000h
PA = 23000h

3) DS = 1000h
DI = 2000h
PA = DS + DI
= 10000
20000
PA = 12000h

4) SS = 8000h
SP = 3A00h
PA = 80000
3A00
PA = 83A00h.

5) If a physical memory address is 5A230h when CS = 5200h, ~~Find~~ what will it be if CS changed 7800h

5A230
− A800
52930h

PA = 5A230h
5 2000
8230h

7800X
8230
FA30

7800X
8230
8F230

Determine PA of the operands for the following instructions.

```
mov AL,[SI]
mov AX,[BP+12]
```

Let DS = 1000h
BP = 4600h
SI = 2500h
SS = 6000h.

DS * 10 + off set

```
 10000
+ 2500
-------
 12500
```

PA

AL, SI → DS

∴ PA = DS + SI

```
 1000 0h
+ 2500h
---------
 12500h
```

PA = SS + BP.

BP = 4600 − 12

4612 + SS

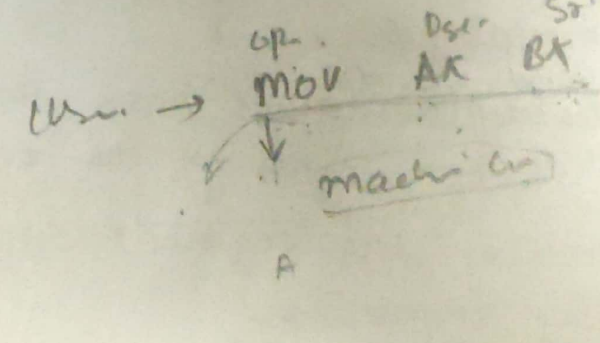
```
PA = 6000 0h
     4612h
-----------
 64612h
```

Real mode memory Address

Real mode operation allows the up to address only the first 1mb of memory space. The first 1mb of memory is called the real memory / conventional / DOS memory.

The 8086 and 8088 operate in exclusively in real mode. The 80286 and above operate in either real or protected mode.

The DOS O.S requires that the up operates in the real mode. Windows does not used real mode. Real mode operation allows application software written for 8086 through 80186 which only contain 1mb memory. The pentium through core-2 operate in 64-bit mode cannot execute reals mode application hence DOS mode can will not execute in 64-bit machines..



UNIT – 2

Protected Mode Addressing

Protected mode memory addressing allows access data and programs located above the 1Mb of memory, as well as with in the first 1Mb of memory.

Addressing this extented section of memory is segment and offset addressing scheme but in pl segment address, the segment regist contains a selector that selects a descriptor from a descriptor table.

The descriptor gives the memory segments length, location and access rights.

Protected memory Instructions are identi to real mode Instructions.

Selectors and Descriptores:

* The selector located in the segment register selects one of the descriptors (8192) from one of two tab

The format o written as

31 | 16
0000 000
B15 | B

15

The base ad the starting

The segmen segment.

Therfore p

Ex: Base
Limit
PA

The acess segment in the sy

The form as.

descriptor u

Not suitable for general purpose and multitasking computers (systems.)

Addressing modes:-

Addressing modes indicates a way of locating data & or operands.

Depending on data types used in the instructions and the memory addressing modes, any instruction may belong to one or more addressing modes.

The different methods in which an operand is specified in the instruction is called its addressing mode.

In intel series µps there are 3 categories of addressing modes.

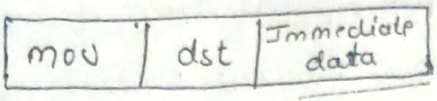
i) Data related addressing modes
ii) Code / Program related addressing modes
iii) Stack related addressing modes.

Data related Addressing modes (mov dst, src)

Immediate AM:-

This addressing mode transfers the

immediate byte, word, double word or quad wo
data into the distination register.

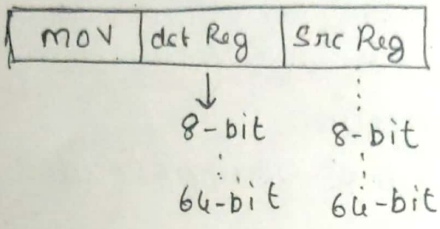


Eg: mov AL, 05h
mov Cx, 1234h } Data

[The data is 8-bit or 16-bi
long and is a part of I
instruction]

* Register addressing mode:

This addressing mode transfers a copy of a b
or word from the source register to the destin
register.



Eg: mov AL, BH → 8-bit
mov CX, DX → 16-bit.
mov AH, CX → not allowed.
8-bit 16-bit
mov ES, DS → not allowed. [We cannot load segme
both are segment registers. registers directly

* Direct Addressing Mode:

In this mode the effective address (offset) of the operand directly available in the instruction itself. This address mode moves a byte or word between a memory locati a register.

mov EA → Data

mov dst, Src Reg/memory
↓
reg/memory

mov mem, mem (Not allowed)

mov AX, DATA1
mov BX, ~~ALPHA~~

Eg:- mov AX, [8A03h]
↳ EA (Effective Address of operand)
(memory location)

mov [1254h], BX
memory location

mov AX, TEMP memory address is defined in the DS.

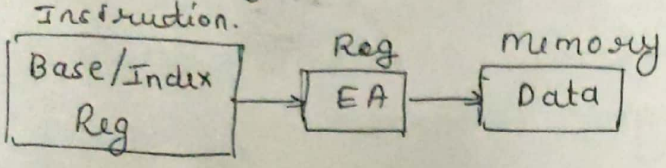
MOV RESULT, AX
↳ memory address of DS.

MOV RESULT, TEMP → not allowed.

Indirect Addressing Mode: (Register Indirect AM)

This addressing mode transfers a byte or word a register and a memory location addressed by index or base register.

The index and base registers are BX, BP, SI & DI. effective address of the data is in a base register index register i.e. specified in the instruction.

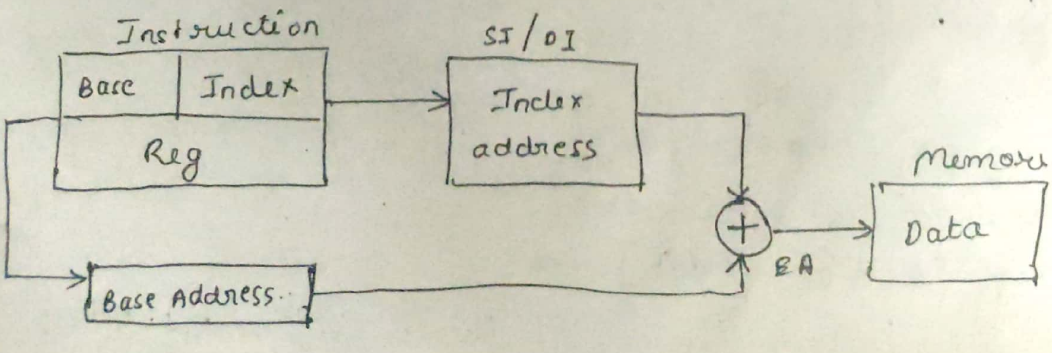



```
MOV AX,[BX]
MOV CX,[BP]
MOV AX,[SI]
MOV [DI],CX
```

Base + Index addressing:-

This addressing mode transfers a byte or word between a register and a memory location addressed a base register (BP or BX) + an index register SI, DI).

The effective address (EA) is sum of base register and an index register. Both of which are specified by instruction. Therefore EA = BP/BX + SI/DI.

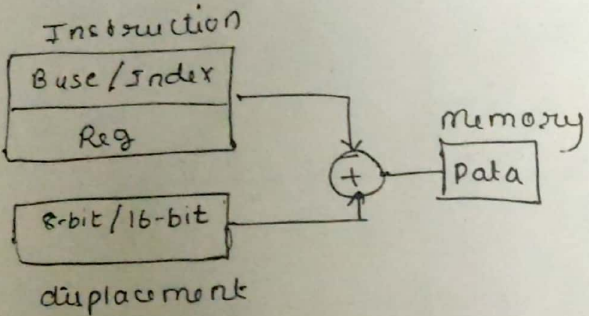


Eg: mov AX, [BX+SI].
mov [BP+DI], CX.

Register Relative addressing: (Displacement) (8-bit, 16-bit

This addressing mode transfers a byte or word between a register and a memory location addressed by an index register (SI or DI) or base register (BX or BP) plus a displacement (8-bit or 16-bit). The EA is the sum of base register (BX or BP) or index register (SI or DI) plus an 8-bit or 16-bit displacement. Therefore the EA = BX / BP or SI / DI + 8-bit or 16-bit displacement

Eg: mov AX, [SI+09h]
mov AX, [BX+1234h]
mov [DI+8A00h], AX.

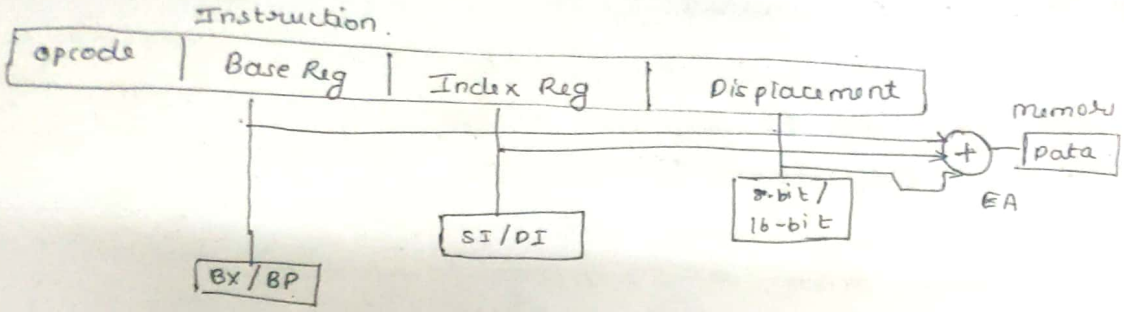


Relative Base Index Addressing mode:

This addressing transfers a byte or word between the register and memory location addressed by a base register and an index register plus a displacement.

The E.A. is the sum of base register (BX or BP) and index register (SI or DI) plus an 8-bit or 16-bit displacement. Therefore E.A. = Base Reg (BX/BP) + Index Reg (SI/DI) + 8-bit/16-bit displacement

Eg: mov AX, [BX+SI+08h]
mov CX, [BP+SI+1234h]
mov [BX+DI+8A00], DX.



Scaled-Index Addressing mode:

This addressing is available only in the 80386 through pentium ups.

The second register of a pair of registers is modified by a scaling factor of two, four or eight to generate the operand memory address

Eg: mov EAX, [EBX + ESI * 2]

MOV [EBX + EDI * 4], ECX.

...lative Instruction Pointer Addressing mode:

This addressing mode is available only to the ...-bit up. (Pentium-4 to core-II up)

This mode allows access to any location in the memory system by adding a 32-bit displacement to the contents of 64-bit Instruction Pointer (IP)

E.A is used to calculate the address of Interrupt ...rvice Routines (ISR)

...put/output Addressing (I/O PORT Addressing):-

This addressing is used to transfer a byte or word ...om the I/O devices to the up. The address of the I/O devices ...u specified by DX register.

Eg: IN AX, DX.

OUT DX, AX

Implied Addressing mode:

In this addressing, the instruction assumes the data is predefined in a register, and does not involve any operand.

Eg: Data conversions instructions.

DAA, AAA, AAS, AAM
(sub, multi)

Decimal adjust accumulator / ~~after~~ addition
ASCII adjust after addition

→ Status flag Instructions:

STD, CLC, CMC

⇒ Program Related addressing modes: (code).

These addressing modes are used with uncond branch instructions (JMP)(Jump) and function or subroutines, ~~call~~ (CALL) instructions, consists of 3

i) Direct
ii) Relative
iii) Indirect

* Intrasegment Direct Addressing mode (AM)

[Relative Program memory Addressing].

i) In this mode, the address to which the control is be transferred. lies in the same segment, The displa is computed relative to the contents of the instruction pointer only.

ii) In this mode the value of CS do not change, the v of displacement is -128 to +127 (SHORT JUMP).

iii) If the displacement is -32768 to +32767 (LONG JUM

Eg: JMP 8A00h
CALL 1234h

* Intra segment Indirect Addressing mode:-

i) In this mode the branch address is found as the content of the register or memory location.

ii) In this mode the CS value do not change, the IP value will be held by any register.

Eg: CALL BX
JMP CX

* Inter segment Direct Addressing

i) In this mode the address to which the control is to be transferred lies in a different code segment.

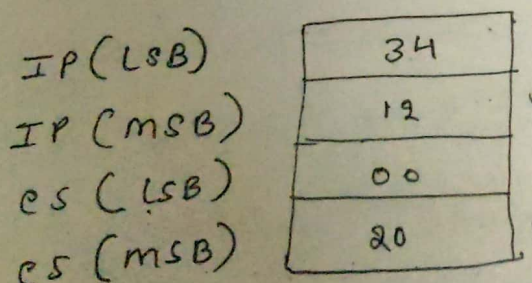
ii) This addressing mode provides branching from one CS to another CS.

iii) The CS and IP of the destination address are specified directly in the instruction.

Eg: CALL 2000h : 1234h.
CS IP

* Intersegment Indirect Addressing mode

In this mode the address to which the control is to be transferred lies in a different code segment and is passed to the instruction indirectly using the contents of a memory block containing 4-bytes.



* Stack memory addressing mode : (PUSH, POP).

The stack plays an important role in all ups. It holds data temporarily and stores the ~~writte~~ return address used by procedures.

The stack memory is LIFO, ... way that the data are stored and removed ... the stack.

Data are placed on to the stack using ... instruction and removed using POP instruction.

The CALL instruction also uses the stack to hold the return address for procedures and return instruction (RET) to remove the return address from the stack.

Syntax:

```
PUSH src
POP  dst
```

The stack memory is maintained by two registers the stack pointer (SP) and the stack segment register (SS). Therefore EA of stack → SP/BP

PA = [SS + SP/BP]

Operation

1) PUSH src

SP ← SP - 2

(SP+1) : SP ← Data

2) POP dst

Data ← (SP+1) : SP

SP ← SP + 2. ✓

Eg: 1) PUSH AX
PUSH BX

2) POP AX
POP BX

NOTE:- NEAR PTR → Intra segment (points)

FAR PTR → Inter segment. (points)

Ex: JMP NEAR PTR [BX] (Assembler Directives).

(Intra segment Indirect Am)

Instructions:

PUSHF (Push flag

POPF

PUSH 1234h (imm

Problems:

Suppose tha

a) mov

b) mov

c) mov [

ow:

a) m

[PA

PA

e) m

PA=

PA=

PA

PA

Suppose

a) P.A =

P.A =

b) PA =

=

PA

Problems:

1) Suppose that DS = 0200h ... Determine the P.A accessed
BX = 0300h by the following instructions
DI = 4000h.

a) mov AL, [1234h]
b) mov AX, [BX]
c) mov [DI], AL

Sol: a) mov AL, [1234h] (Direct)

[PA = DS x 10 + offset]

= 0200h x 10 + 1234h

PA = 03234h

b) mov AX, [BX]

PA = 0200h x 10 + 0300h

PA = 02300h

(Indirect)

c) mov [DI], AL

PA = 0200h x10 + 4000h

~~PA = 06000h~~

PA = 02400h. (Indirect)

PA = 02400h

2) Suppose that DS = 1200h
SI = 2024h
ARRAY = 0012h.
BX = 1012h

Find P.A for the following
a) mov AL, ARRAY
b) mov AL, ARRAY[BX]
c) mov AL, ARRAY[BX][SI]

a) P.A = 2024h x 10 + 0012h
P.A = ~~20252h (Direct)~~ (12012h) (Direct)

b) PA = 1200h x 10 + 1012h
= 13024h (~~Indirect~~) (Register relative).

c) PA = 1200h x10 + 1012h + 2024. (Relative Base Index Am)
= 12000 + 3036
= ~~15036h~~
= 15048h

Scanned by CamScanner

3) At a certain instant during execution of pgm the up has the following data in the registers

AX = 1234h
BX = 5678h
CX = ABCDh
DI = CDEFh
CS = 3456h
IP = 789Ah
DS = ES = 4567h

State the addressing modes used and calculate the PA for the following instructions.

1) MOV AX, BX
2) MOV [BX], AX
3) MOV WORD PTR [BX + DI + 3456h], 9ABCh
4) MOV AX, [9ABCh]
5) LODSW

1) MOV AX, BX (Register A.M)
2) MOV [BX], AX (Indirect AM)
PA 5678 × 10 + 1234h.

3) Relative Base Index AM.

4) MOV AX, [9ABCh] (Direct)

5) LODSW → Load String word.
ES + DI = PA
PA = 45670h
CDEFh

4) Determine the addressing modes for the following instructions.

1) PUSH BX — Stack AM
2) CALL BX — Intra segment indirect AM
3) JMP DWORD PTR 6200h [BX] — Inter segment
4) OR 0ABCDh [BX][SI], CX.
Displacement Base Index AM.

e) Indicate the default segment registers and calculate the PA. for the following instructions

CS = 0000h
DS = 2000h
ES = 4000h
SS = 6000h
BP = 4000h
SI = 3000h
SP = 1000h

a) ADD [BP], 2000h → (Indirect AM)
b) PUSH SI → stack related AM
c) LES BX, 42h [SI] ~~Indirect~~ → Rel relative
d) OR DS: OF 246 [BP], DX ~~Interseg~~ Seg relative
e) JMP 1234h. Intra direct AM

a) 40000h
   2000h
   42000h

b) SS + SP/BP.

c) Register Relative AM
Load Effective addr ing

d) DS + F246 + BP
~~Relative Base Index~~ AM
Register Relative

e) Write AM

1) CALL 2400h — Intra-direct
2) CALL WORD PTR [SI] Intra-indirect
3) CALL 1200h : 2400h - Inter-direct
4) CALL DWORD PTR [SI] - Inter - indirect.