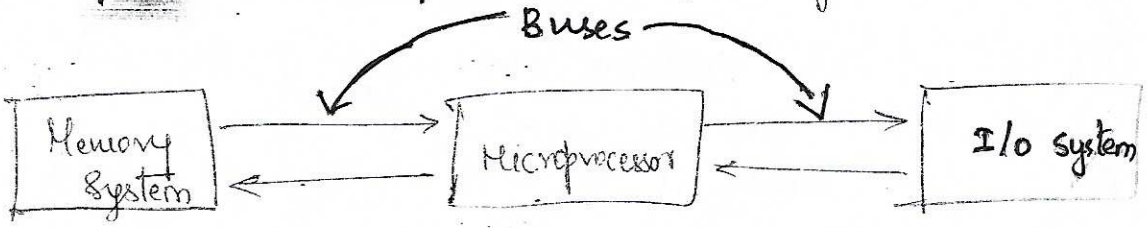
Microprocessor based personal Computer System



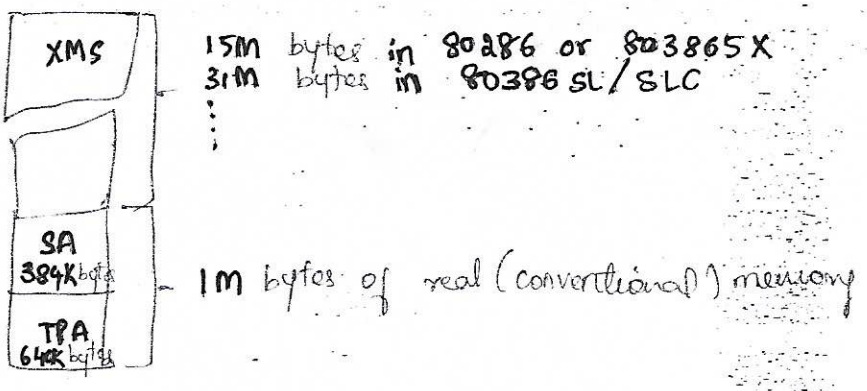
Memory and I/o System

① Memory structure of all Intel based personal computers is similar from 8088 to Pentium 4 or Core-2 they are same.

② Memory system is divided into 3 main parts
   i) TPA (Transient Program Area)
   ii) System Area
   iii) XMS (Extended Memory System)

③ Whether an extened memory is there or not is decided by the type of microprocessor used.
   If it is really old 8086 or 8088 (PC or XT), the TPA and System Area exist, no XMS.
   Systems based on 80286 through Core 2 not only contain TPA and SA but also XMS.

Memory Map of a personal Computer



④ PC and XT contains 640K bytes of TPA and 384K bytes of system memory.
Totally it is 1024K bytes (1M bytes)

⑤ First 1M byte of memory is called real or conventional memory system because each Intel microprocessor is designed to function in this area by using its real mode of operation.

⑥ Machines having TPA, SA and XMS are called as AT class machines.

Eg: PS/1 and PS/2 produced by IBM.

↓
They are also referred to as ISA (Industry standard Architecture) or EISA (Extended ISA)

Changes that were introduced from ~~Pentium~~ ATX class machine and Pentium microprocessor

① Addition of a bus called PCI (Peripheral Component Interconnect) bus.

→ ISA machines contains 8-bit peripheral bus which is used to interface 8-bit devices to the computer 8086/8088 based PC or XT systems.

→ AT class machine (ISA machine) uses 16-bit peripheral bus for interface & contain 80286 and above µp.

→ EISA bus has 32-bit peripheral interface bus found in older 80386 DX and 80486 based systems.

(2) Addition of a bus type called VESA local bus or VL bus.

→ They interface disk and video to microprocessor at the local bus level which allows 32-bit interfaces to function at the same clocking speed as the microprocessor

↓

Recent modification to VESA local bus supports 64-bit data bus

(3) Addition of 3 new buses in ATX class systems.

(i) First was USB (Universal Serial bus)

↓

It is used for connecting peripheral devices such as keyboards, mouse, modems, sound cards to µp through a serial data path & twisted pair of wires.

↓

a) Main idea was to reduced sln cost by reducing wires.
b) Sound system can have separate power supply → so less noise.

(ii) AGP (Advanced Graphics port) for video cards

↓

It transfers data b/w video card and µp at higher speed.

(iii) SATA (Serial ATA interface) bus

↓

transfers data from PC to hard disk drive.

TPA

① It holds DOS (Disk Operating System) and other pgms that control computer system.

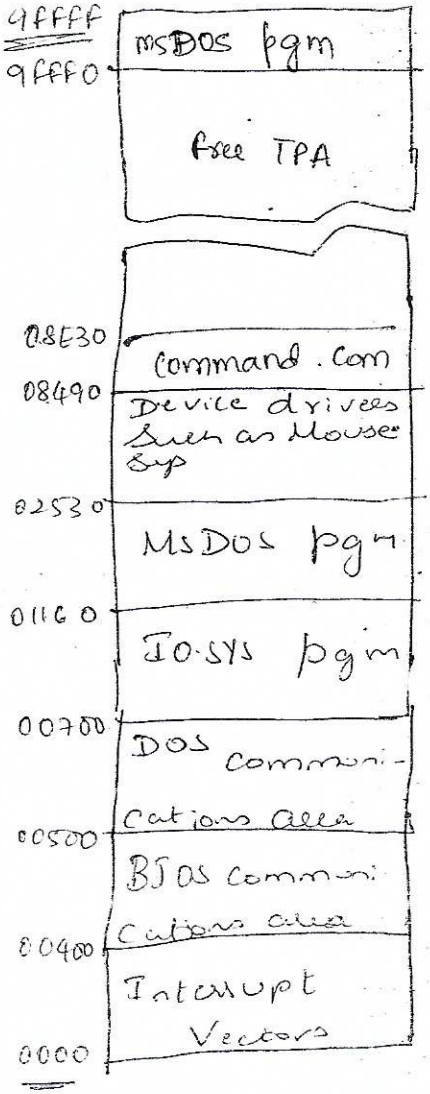
↓

It is a DOS concept & is not really applicable in Windows.

② TPA stores any currently active or inactive DOS appln pgms

③ length of TPA is 640 K bytes.

Memory Map of the TPA in a personal computer.



Organization of TPA in a computer system running DOS.

* To the left of each area is a hexadecimal number that represents the memory addresses that begin and end each data area.

* Hexadecimal memory addresses or memory locations are used to number each byte of memory system.

* Hexadecimal number is a number represented in radix 16 or base 16, with each digit representing a value from 0 to 9 and A to F.

We often end a hexadecimal number with an H to indicate that it is hexadecimal value.

Eg: 1234H is 1234 hexadecimal.

1234 can also be represented as 0x1234.

* Interrupt vectors:

They access various features of DOS, BIOS and appl's.

* BIOS Communication area:

System BIOS is a collection of pgms stored in either a read only (ROM) or flash memory that operates many I/O devices connected to computer system.

* DOS Communication area

BIOS and DOS Contain Transient data used by pgms to access I/O devices & internal features of Comp. System.

* IO.SYS program

It is a program that loads into TPA from disk whenever an MSDOS system is started.

* They contain pgms that allow DOS to use keyboard, video display, printer and other I/O devices.

* Drivers

↓

They are pgms that control installable I/O devices such as mouse, disk cache, CD-ROM, DVD as well as pgms.

* COMMAND.COM (command processor)

↓

Controls the operation of the computer from the keyboard when operated in DOS mode.

↓

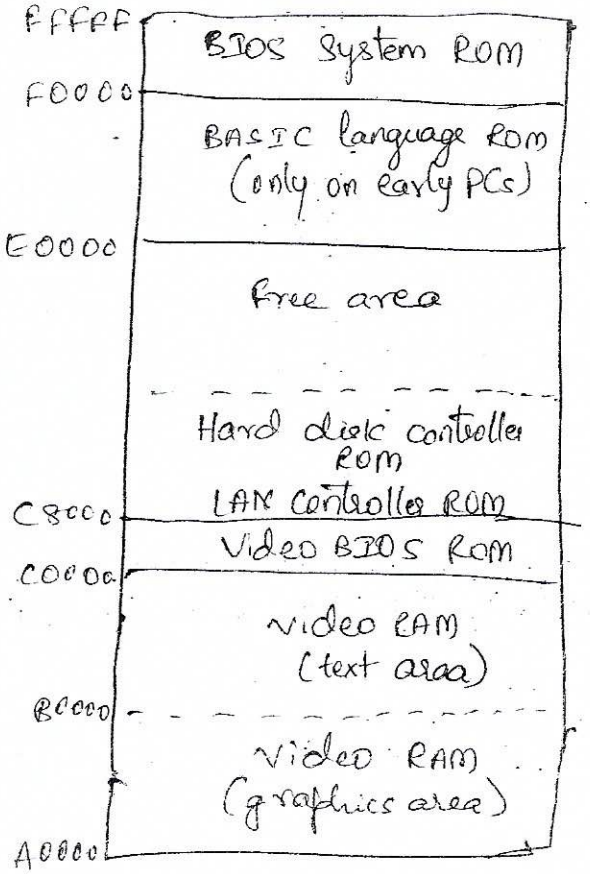
They process the DOS commands as they are typed from keyboard.

Eg: if DIR is typed, it displays a directory of the disk files in current disk directory.

System Area

* System area contains programs on either a ROM or flash memory and areas of read/write (RAM) memory for data storage.

System Area of a Typical personal Computer



* Video RAM (display) and video control programs or ROM or flash memory.

↓

This area starts at location A0000H and extends to location C7FFFH.

↓

Size and amount of memory used depends on the type of video display adapter attached to system

* Display adapters generally have their video RAM located at A0000H - AFFFFH, which stores graphical or bit-mapped data and the memory at B0000H - BFFFFH stores text data.

* VIDEO BIOS

↓

It is located on a ROM or flash memory is at locations C0000H - C7FFFH and contains pgms that control DOS video display.

* Area at locations C8000H - DFFFFH is open or free.

↓

This area is used for the expanded Memory System in a PC/XT or for upper memory system in an AT System.

* Memory locations E0000H - EFFFFH contain the cassette BASIC language on ROM found in early IBM systems.

↓

This area is often open or free in newer computer systems

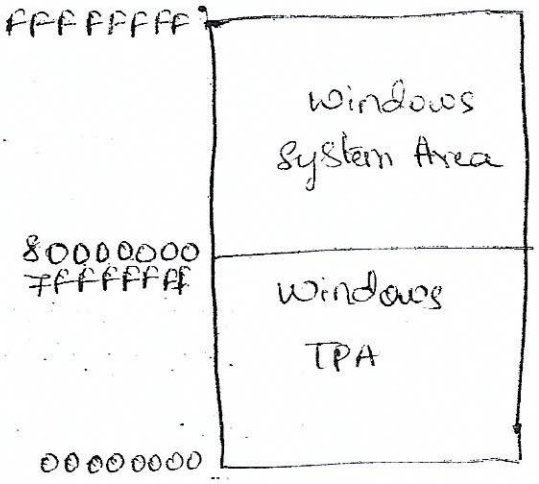
* System BIOS ROM is located in the top 64K bytes of System Area (F0000H - FFFFFH).

↓

First part of the system BIOS often contains programs that set up the computer

Second part contains procedures that control basic I/O system.

Windows Systems



- Windows TPA is the first 2G bytes of memory system from location 00000000H to 7FFFFFFFH.
- Windows system area is last 2G bytes of memory from locations 80000000H to FFFFFFFFH.
- System area is where the system BIOS is located and also video memory.
- Also the actual windows program and drivers are located in system area.

I/O Space

* I/O Space in Computer system extends from I/O port 0000H to port FFFFH.

* I/O devices allow microprocessor to Communicate between itself and outside world.

* I/O area Contains two major Sections
  (1) Area below I/O location 0400H is Considered reserved for system devices.
  (2) Remaining area is available I/O space for expansion that extends from I/O port 0400H through FFFFH.

The Microprocessor

(1) Microprocessor, referred as CPU is the controlling element in a Computer System.

(2) Microprocessor controls memory and I/O through a series of Connections called buses

③ Buses select an I/O or memory device, transfer data b/w I/O device or memory and microprocessor.

The three main tasks performed are
① Data transfer b/w itself and memory or I/O system
② Simple ALU operations
③ Program flow via simple decisions

* Power of microprocessor is its capability to execute billions of millions of instructions per second from a pgm or software stored in memory system.

* Data widths are variable and include
byte (8 bits)
word (16 bits)
doubleword (32 bits)
Quadwords (64 bits)
Octalwords (128 bits)

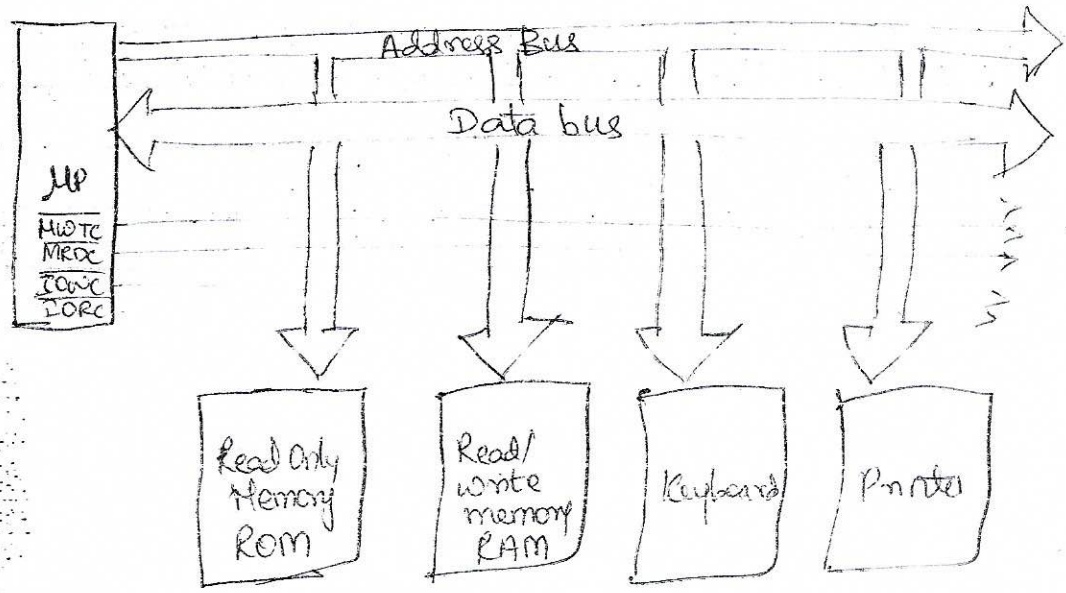
* Another feature that makes microprocessor powerful is its ability to make simple decisions based upon neumerial facts.
Eg: A microprocessor can decide if a number is zero, if it positive and so forth.

Decision	Comment
① Zero	test a no for zero or nonzero
② Sign	test a no for positive or negative
③ Carry	test for a carry after addition or a borrow after subtraction.
④ Parity	test a no for an even or an odd number of ones.
⑤ Overflow	test for an overflow that indicates an invalid result.

Buses

* A bus is a common group of wires that interconnect components in a computer system.

* In the microprocessor based computer system, three buses exist for the transfer of information
  ① Address ② Data ③ Control.



Block diagram of a computer system showing address, data and control bus structure

* Address bus requests a memory location from the memory or an I/O location from I/O devices.

If I/O is addressed, address bus contains a 16-bit I/O address which inturn selects one of 64k different I/O devices.

If memory is addressed, address bus contains a memory address.

* Data bus transfers information between the microprocessor and its memory and I/O address space.

Data transfers vary in size.

* Control bus contains lines that select the memory or I/O and cause them to perform read or write operation.

The four control bus connections are

1) $\overline{MRDC}$ (Memory Read Control)

2) $\overline{MWTC}$ (Memory write Control)
3) $\overline{IORC}$ (I/o read Control)
4) $\overline{IOWC}$ (I/o write Control)

* Overbar indicates the control signal is active low. i.e, it is active when a logic zero appears on the control line.

Eg: If $\overline{IOWC} = 0$, the microprocessor is writing data from the data bus to an I/O device whose address appears on the address bus.

Steps:

① µp reads the contents of a memory location by sending the memory an address through address bus.

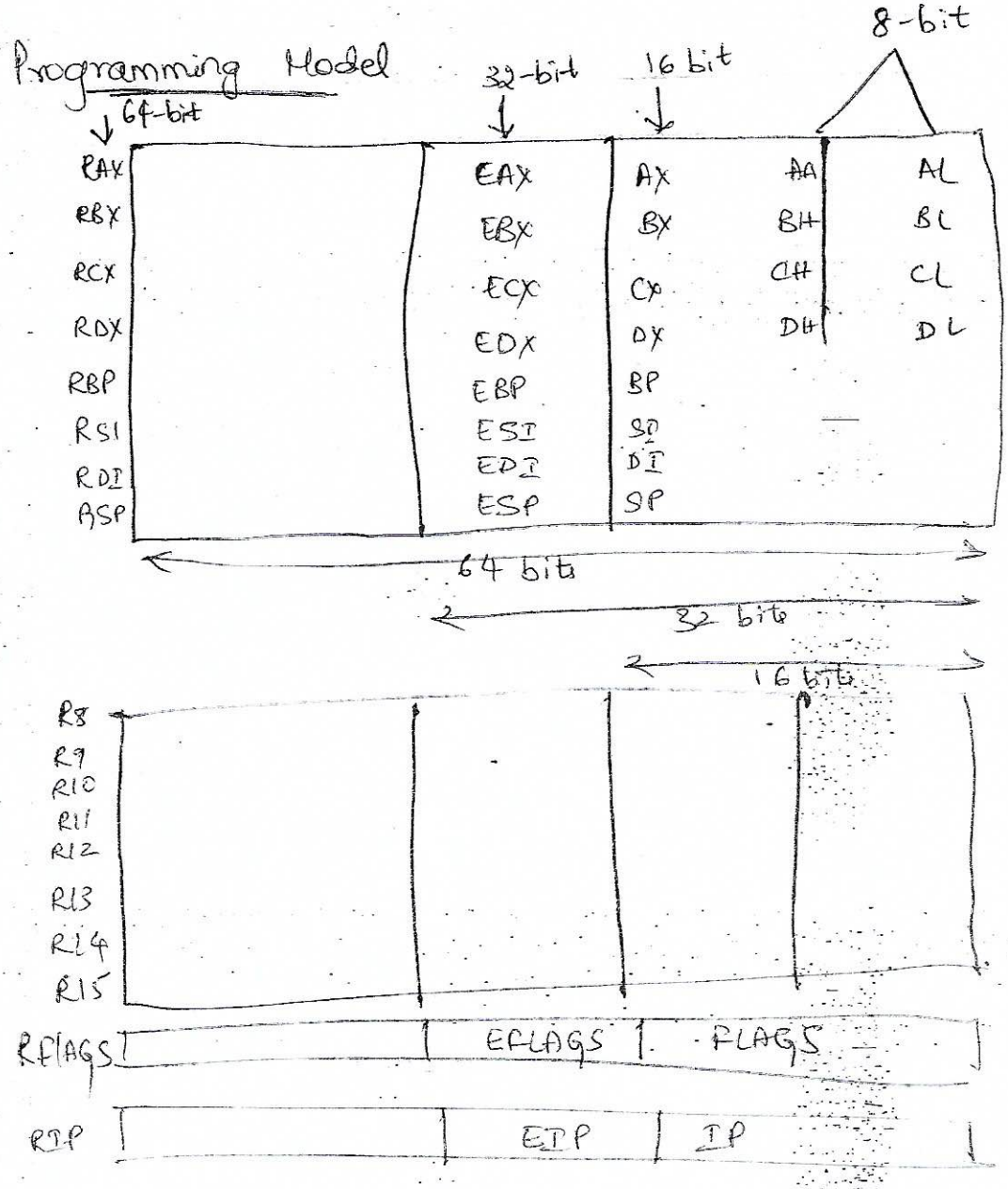
② Next it sends memory read control ($\overline{MRDC}$) to cause memory to read data.

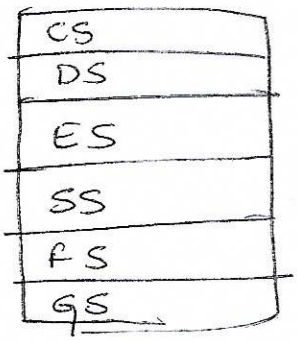
③ Finally, data read from memory are passed to µp through data bus

Microprocessor and its Architecture

Internal Microprocessor Architecture

Programming Model





* Programming model of 8086 through Core2 is called program visible because its registers are used during appln pgming and are specified by instructions.

* Other registers are considered as program invisible because they are not addressable directly during appln pgming but can be used indirectly during system pgming.

* Pgming model contains 8, 16 and 32 bit registers.

* 8 bit registers are AH, AL, BH, BL, CH, CL, DH, DL.

Eg: ADD AL, AH

adds 8 bit content of AH to AL.

* 16 bit registers are AX, BX, CX, DX, SP, BP, IP, SI, DI, FLAGS, CS, DS, ES, SS, FS and GS.

* Multipurpose registers include EAX, EBX, ECX, EDX, EBP, EDI and ESI.

* R8 - R15 are additional 64-bit registers.

* The additional 64-bit registers (R8 - R15) are addressed as a byte, word, doubleword or quadword

Flat mode 64-bit access to numbered registers

Register Size	Override	Bits Accessed	Eg:
8 bit	B	7-0	MOV R9B, R10B
16	W	15-0	MOV R10W, AX
32	D	31-0	MOV R14D, R15D
64	-	63-0	MOV R13, R12

Eg: MOV R11D, R8D
Copies the low-order doubleword from R8 to R11.

Multipurpose Registers

① RAX (accumulator)

→ RAX is referenced as 64-bit register (RAX), 32-bit register (EAX), 16 bit register (AX) or as either of 2 8-bit register (AH and AL)
→ Accumulator is used for instructions such as multiplication, division and some of adjustment instructions

② RBX (Base Index)

→ BX register holds offset address of a location in the memory system.

③ RCX (Count)

→ It holds the count for various instructions

→ Instructions that use a count are the repeated string instructions (REP/REPE/REPNE) and shift, rotate and LOOP/LOOPD instructions

→ Shift and rotate use CL as count
Repeated string use CX as count
LOOP/LOOPD instructions use either CX or ECX

④ RDX (Data)

→ It holds a part of the result from a multiplication or part of the dividend before a division.

→ They can also address memory data.

⑤ RBP (Base Pointer)

→ It points to a memory location in all versions of microprocessor for memory data transfers.

⑥ RDI (Destination Index)

→ It often addresses string destination data for string instructions.

⑦ RSI (Source Index)

→ It addresses source string data for string instructions.

→ As 16-bit register, it is addressed as SI

32-bit ⟶ ESI

64-bit ⟶ RSI

(8) R8 through R15

→ These registers are found only in Pentium 4 and Core 2 if 64-bit extensions are enabled.

→ In these registers, 8-bit portion is the rightmost 8-bit only.
Bits 8 to 15 are not directly addressable as a byte.

Special purpose Registers

* Special purpose registers include RIP, RSP and RFLAGS.

① RIP (Instruction Pointer)

* It address the next instruction in a section of memory defined as a code segment

→ This register is IP (16 bits) when µp operates in real mode and EIP (32 bits) when 80386 operate in procted mode.

→ It is used by µp to find the next sequential instruction in a program located within code segment.

② RSP (stack Pointer)

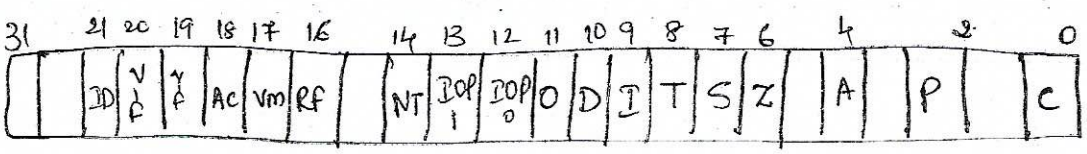
→ RSP addresses an area of memory called the stack.

→ stack memory stores data through this pointer.

(3) RFLAGS :

→ RFLAGS indicate the condition of the microprocessor and control its operation.

→ 8086 - 80286 Contain FLAG register (16 bits)
80386 and above contain EFLAG (32-bits)



EFLAG and FLAG register counts for entire 8086 & Pentium µp family

* The rightmost five flag bits and overflow flag change after many arithmetic and logic instructions execute.

* Flags never change for any data transfer or program control operation.

① C (Carry)

→ It holds carry after addition or borrow after subtraction.
→ They also indicate error conditions

② P (Parity)

→ It is a logic 0 for odd parity
logic 1 for even parity

Eg: If a number contains 3 binary 1 bits, it has odd parity
If a number contains no one bits, it has even parity.

③ A (Auxiliary Carry)

→ It holds the carry (half-carry) after addition or the borrow after subtraction b/w bit positions 3 and 4 of the result.

④ Z (zero)

→ It shows the result of an arithmetic or logic operation is zero.
If Z = 1, the result is 0
Z = 0, result is not zero.

⑤ S (Sign)

→ It holds arithmetic sign of the result after an arithmetic or logic instruction executes.
If S = 1, sign bit is set or negative
S = 0, sign bit is cleared or positive

⑥ T (trap)

→ It enables trapping through an on-chip debugging feature.
→ If T flag is enabled, µp interrupts flow of pgm on conditions as indicated by debug registers and control registers.

→ If T flag is logic 0, trapping (debugging) feature is disabled.

⑦ I (Interrupt)

→ It controls the operation of INTR (interrupt request) input pin.

If I = 1, INTR pin is enabled
I = 0, INTR pin is disabled.

→ State of I flag was controlled by STI (Set I flag) and CLI (clear I flag) instructions.

⑧ D (direction)

→ It selects either increment or decrement mode for DI and/or SI registers during string operations.

→ If D=1, registers are automatically decremented.
D=0, registers are incremented.

(9) O (Overflow)

→ It occurs when signed numbers are added or subtracted.
Overflow indicates that result has exceeded capacity of the machine.

(10) IOPL (I/O privilege level)

→ It is used in protected mode operation to select the privilege level for I/O devices.

→ If current privilege level is higher or more trusted than IOPL, I/O executes normally.

→ If is is lower, an interrupt occurs causing execution to suspend.

→ IOPL of 00 is highest or most trusted
IOPL of 11 is lowest or least trusted

(11) NT (Nested Task)

→ This flag indicates current task is nested within another task in protected mode operation.

(12) RF (Resume)

→ It is used with debugging to control the resumption of execution after the next instruction.

(13) VM (Virtual mode)

→ VM flag bit selects virtual mode operation in a protected mode system

(14) AC (Alignment check)

→ It is activated if a word or doubleword is addressed on a non-word or non-doubleword boundary.

(15) VIF (Virtual Interrupt)

→ It is a copy of interrupt flag bit available to Pentium 4 microprocessors.

(16) VIP (Virtual Interrupt Pending)

→ VIP provides information about virtual mode interrupt for Pentium 4 µp

(17) ID (Identification)

→ ID flag indicates that Pentium - Pentium 4 microprocessors support CPUID instruction.
→ CPUID provides information such as its version number and manufacturer.

Segment Registers

① CS (Code)

→ It holds the code (pgms and procedures) used by the microprocessor.

→ Code segment register defines starting address of the section of memory holding code.

→ In real mode operation, it defines the start of a 64K-byte section of memory

In protected mode, it selects a descriptor that describes starting address and length of a section of memory holding code.

② DS (Data)

→ 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 hold offset address.

⑧ ES (Extra)

→ It is an additional data segment that is used by some of the string instruction to hold destination data.

⑭ SS (Stack)

→ It defines area of memory used for the stack.

→ Stack entry point is determined by the stack segment and stack point registers.

⑤ FS and GS

→ These are supplemental segment registers

available in 80386 - Core 2 microprocessors to allow two additional memory segments for access by programs.

Real Mode Memory Addressing

→ Real mode operation allows the microprocessor to address only the first 1M byte of memory space.

→ DOS operating system requires µp to operate in real mode, Windows does not use real mode.

→ Each of these (80286 & above) begins operation in the real mode by default whenever power is applied or the microprocessor is reset.

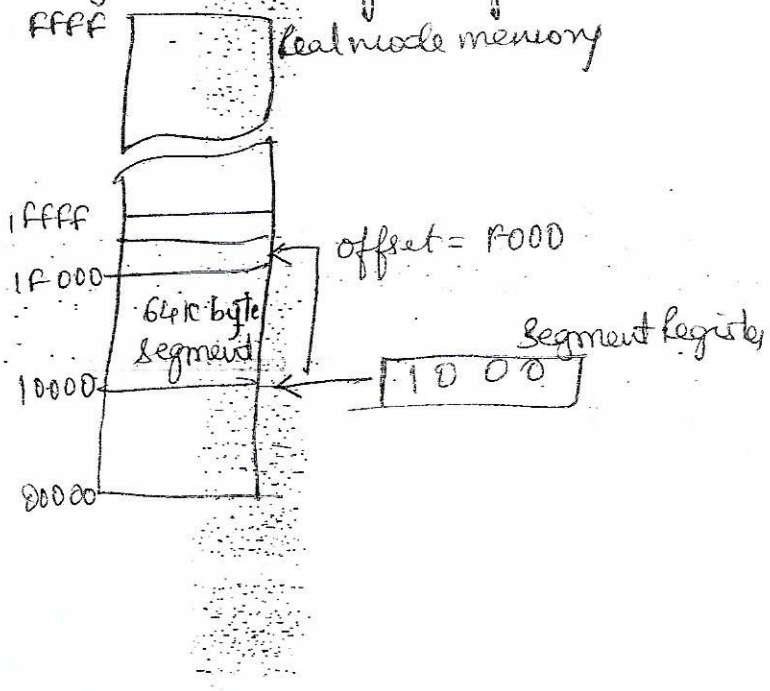
Segments and offsets

* Combination of a segment address and an offset address accesses a memory location in the real mode.

Real mode memory address = Offset address + Segment address

* Segment address, located in one of the segment registers, defines the beginning address of any 64K-byte memory segment.

* Offset address selects any location within the 64K byte memory segment.



* As indicated in above figure, segment register contains 1000H, yet it addresses a starting segment at location 10000H.

* In real mode, each segment register is internally appended with 0H on its rightmost end.

↓

This forms 20-bit memory address allowing it to access a start of a segment.

Eg: when a segment register contains 1200H, it address 64K-byte memory segment beginning at location 12000H.

* Because of internally appended 0H, real mode segments can begin only at a 16-byte boundary in memory system. This 16-byte boundary is called paragraph.

* Since real mode segment of memory is 64k in length, if the beginning address is known, ending address is found by adding FFFFH.

Eg: If segment register contains 3000H,
first address of segment = 30000H
last address = 30000 + FFFFH.
= 3FFFFH.

* The offset address, which is a part of address is added to the start of the segment to address a memory location within memory segment.

Eg: If segment address is 1000H,
offset address = 2000H
Up addresses memory location 12000H.

because start of the segment is 10000H + offset address (2000H)
10000H + 2000H = 12000H

* Some addressing modes combine more than one register and an offset value to form an offset address.
  Eg: ~~Segment address = 4000H~~

Example of real mode segment addresses

Segment Register	Starting Address	Ending Address
2000H	20000H	2FFFFH
2001H	20010H	3000FH
2100H	21000H	30FFFH
AB00H	AB000H	BAFFFH

Default segment and offset registers

* The code segment register is always used with the instruction pointer to address the next instruction in a program.
  This combination is CS:IP or CS:EIP.

* The code segment registers defines the start of the code segment and instruction pointer locates the next instruction within code segment

Eg: If CS = 1400H and IP/EIP = 1200H, µp fetches the next instruction from memory location

14000H + 1200H = 15200H

* Stack data is referenced by either stack pointer (SP/ESP) or base pointer (BE/EBP)
* These combinations are referred to as SS:SP (SS:ESP), or SS:BP (SS:EBP)

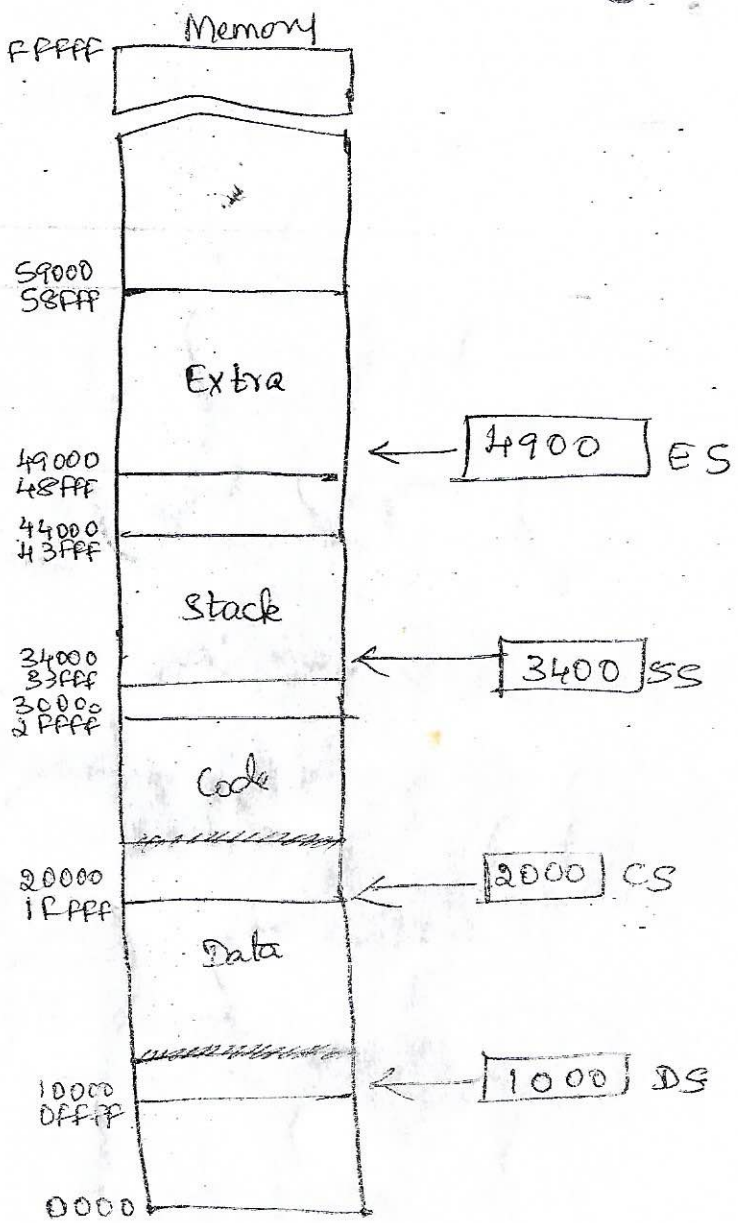
Eg: If SS = 2000H and BP = 3000H
µp addresses memory location
20000H + 3000H = 23000H

Default 16-bit segment & offset combinations

Segment	Offset	Special purpose
CS	IP	Instruction address
SS	SP or BP	Stack address
DS	BX, DI, SI	Data address
ES	DI for string instructions	String destination address.

* when an application program is placed in the memory system by DOS, it is loaded in the TPA at the first available area of memory above drivers and other TPA programs. This area is indicated by free-pointer that is maintained by DOS.

→ Program loading is handled automatically by program loader located within DOS.



A memory system showing placement of four memory segments.

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
− 4800
5230h

PA = 5A230h
5 2000
8230h

7800
8230
FA30

7800
8230
8F230

* 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)

3) 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.

1 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

1 FS & GS (Supplement / General).

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

Real mode – 1M
Protected " – 1M

xi) VM – Virtual mode.

- ~~When VM=1~~ This flag selects Virtual mode operation in a protected mode system.
- It allows system program to execute multiple OS programs and not application program (dos)
- It is used to simulate dos in the modern windows 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 mode system available to the pentium processors.

xiv) VIP – Virtual Interrupt Pending

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

xv) ID – Identification

- This flag indicates that the pentium µp's support CPUID instruction. This instruction provides the information about the µp version / manufacturer

Segment Registers:

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

1) Code Segment (CS)

* It is a section of memory that holds the code (programs & procedures) used by the µp

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

ter logic

ii) 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.
- For unsigned operation, this flag is ignored.

) IOP → IOPL (Input output Privileged level)

t after :) 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 protected mode operation

) RF - Resume Flag.

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

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 operation)

iv) Z - Zero flag

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

1010 1001
0101 0110
AND op 0000 0000 ∴ Z=1

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)

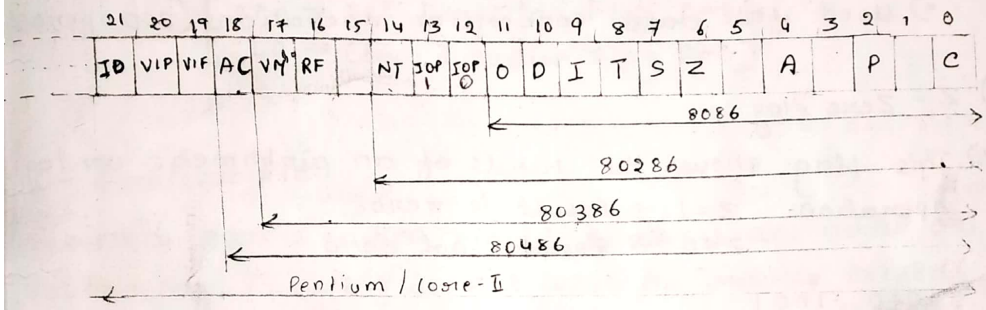
- If T=1, enables on chip debugging feature.
If T=0, debugging feature disabled.

vii) 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)

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 chased by most arithmetic and logic operations. [status registers]
- Data transfer operations do not change flags.

MOV POP XLATG PUSH

⇒ Flags:

) C- Carry flag:

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

) 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.

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 durin 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 (CA instructions.

8-bit, 8-bit

AX → AH, AL
BX → BH, BL
CX → CH, CL
DX → DH, DL

General purpose or multipurpose registers.

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

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

B) 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.

C) 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]

D) RDX → EDX → DX → DH, DL

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

[IN AX, DX
OUT DX, AX]