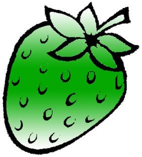
STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com

INTRODUCTION TO MICROPROCESSORS

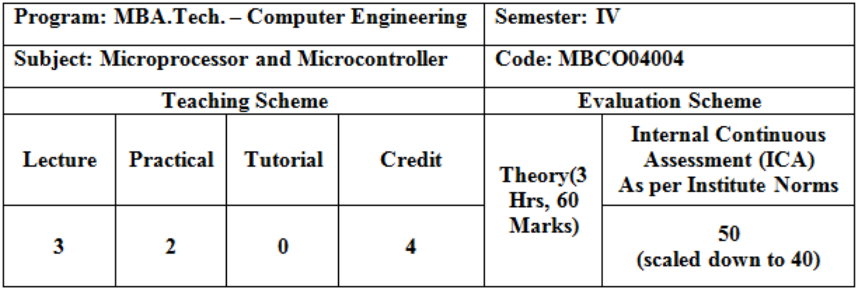
Richa Upadhyay Prabhu

NMIMS's MPSTME

richa.upadhyay@nmims.edu

January 7, 2016

Course Design



Program: MBA.Tech. – Computer Engineering				Semester: IV	
Subject: Microprocessor and Microcontroller				Code: MBCO04004	
Teaching Scheme				Evaluation Scheme	
Lecture	Practical	Tutorial	Credit	Theory(3 Hrs, 60 Marks)	Internal Continuous Assessment (ICA) As per Institute Norms
3	2	0	4		50 (scaled down to 40)

Prerequisite: Digital Logic Design

SYLLABUS

1. Intel 8086/8088 microprocessor family
2. Programming of 8086
3. 8086 Interrupt Structure
4. Programmable Interface and Peripheral device
5. 8087 Math Co-processor
6. Introduction and hardware of 8051 Microcontroller
7. 8051 Assembly Language Programming
8. Microcontroller Design and Interfacing

Books

1. Douglas Hall,(2006) "Microprocessors Interfacing and Programming", Tata McGraw Hill Publication.
2. Muhammad Ali Mazidi, (2011) "Microcontroller and Embedded system", Second Edition Prentice Hall.

Let's Begin

What are the Types of Computers?

COMPUTERS

Types of Computers

- Mainframes
- Minicomputers
- Microcomputers

COMPUTERS

MAINFRAMES

- Largest, Fastest and most powerful
- Massive memory
- 64 bit data words
- uses:
  - military defence control
  - creating computer graphics for science fiction machine

COMPUTERS

MINICOMPUTERS

- Scaled down versions of Mainframe
- small data words i.e 32 bits
- lesser memory as compared to mainframes
- uses:
    - scientific research
    - industrial control for eg. oil refinery

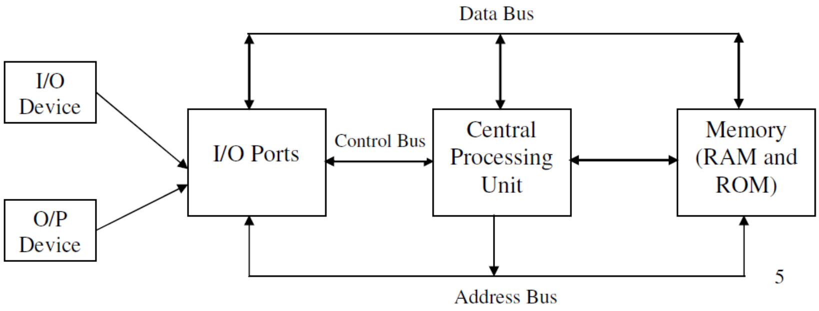
COMPUTERS

MICROCOMPUTERS

- small computers
- 4 bit data word
- can address thousands of bytes of memory with 64 bit address word
- CPU is single IC called **MICROPROCESSORS**
- Used in everything from smart sewing machines to CAD systems

Microprocessors are CPUs to which ROM, RAM and input/output ports are added to make a (micro)Computer.

Block Diagram of Microcomputer



Block Diagram of Microcomputer

MEMORY

- RAM + ROM
- Purpose :
  - Storing binary codes for sequence of instructions that computer would carry out
  - Storing binary coded data with which the computer is going to be working

Block Diagram of Microcomputer

INPUT/OUTPUT

- Allows computer and user to interact with each other
- I/O ports are simply set of parallel D - flipflop
- Data is transferred from/to latches when they are enabled by a control signal by CPU

Block Diagram of Microcomputer

CENTRAL PROCESSING UNIT

- control operation of computer
- CPU is the MICROPROCESSOR
- CPU performs the following functions:
    - fetches binary coded instructions from memory
    - decodes instructions in series of simple actions
    - carries out actions in sequence

We will be studying this in detail.....

Block Diagram of Microcomputer

ADDRESS BUS

- Consists of 16,20,24 or 32 parallel signal lines
- These are used by CPU to send address of memory location that is to be written to or read from
- N address lines can directly address 2^N memory locations
- Eg: 16 address lines can address $2^16 = 65,536$ memory locations.

Block Diagram of Microcomputer

DATA BUS

- Consists of 8, 16 or 32 parallel signal lines
- Bi-directional
- These help CPU to read data from memory or port
- Also Send data on memory or port

Block Diagram of Microcomputer

CONTROL BUS

- Consists 4 or 10 parallel signal lines
- Typical control bus signals are :
  - Memory Read and Write
  - I/O Read and Write

OUR CONCERN IS ONLY WITH THE CPU i.e. MICROPROCESSOR

EVOLUTION OF MICROPROCESSOR

INTEL 4004

- Year of introduction 1971
- 4-bit microprocessor (4 bit register)
- 4 KB main memory
- No. of Transistors : 2300
- was first programmable device which was used in calculators

The World's first Microprocessor

EVOLUTION OF MICROPROCESSOR

INTEL 8008

- Year of introduction 1972
- 8-bit version of 4004
- 16 KB main memory
- No. of Transistors : 3500

EVOLUTION OF MICROPROCESSOR

INTEL 8080

- Year of introduction 1973
- 8-bit microprocessor
- 64 KB main memory
- 10x faster than 8008
- No. of Transistors : 4500
- Drawback was that it needed three power supplies.
- Small computers (Microcomputers) were designed in mid 1970s using 8080 as CPU

EVOLUTION OF MICROPROCESSOR

INTEL 8085

- Year of introduction 1975
- 8-bit microprocessor-upgraded version of 8080
- 64 KB main memory
- No. of Transistors : 6500
- Intel sold 100 million copies of this 8-bit microprocessor
- uses only one +5v power supply.

EVOLUTION OF MICROPROCESSOR

INTEL 8086/8088

- Year of introduction 1978 for 8086 and 1979 for 8088
- 16-bit microprocessors
- Data bus width of 8086 is 16 bit and 8 bit for 8088
- 1 MB main memory
- No. of Transistors : 29000
- 6 byte instruction cache for 8086 and 4 byte for 8088
- Other improvements included more registers and additional instructions
- In 1981 IBM decided to use 8088 in its personal computer

EVOLUTION OF MICROPROCESSOR

INTEL 8086 family overview

Parameters	**80286**	**80386**	**80486**
Year	1982	1985	1989
speed	12MHz	16MHz	25MHz
Transistors	134000	275000	1200000
Register and databus	16-bit	32-bit	32-bit

EVOLUTION OF MICROPROCESSOR

- Intel Pentium (1993)
- Pentium Pro (1995)
- Pentium II (1997)
- Pentium III (1999)
- Pentium IV (2000)
- Core 2 series
  - Core 2 duo
  - Core 2 Quad
  - Core 2 Extreme
- Core i series
  - Core i3, i5, i7, i7 Extreme

FEATURES OF 8086

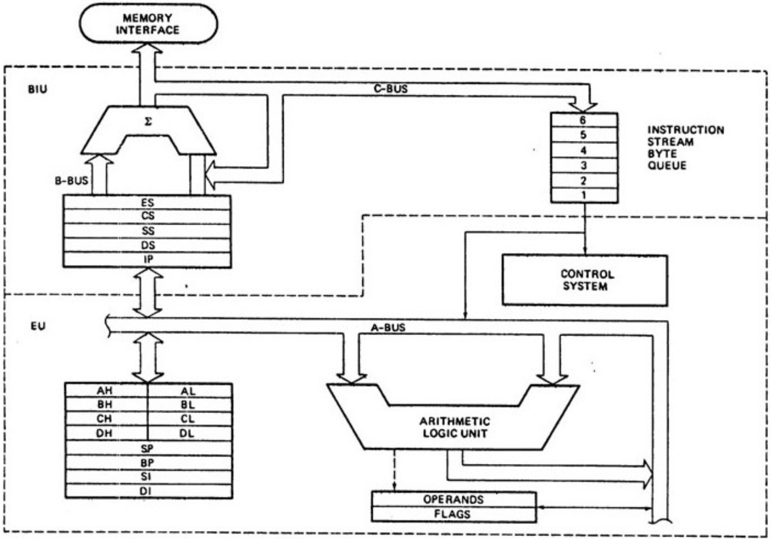
- 8086 is a 16-bit processor. Its ALU, internal registers works with 16-bit binary word
- 8086 has a 16-bit data bus. It can read or write data to a memory/port either 16-bits or 8 bit at a time
- 8086 has a 20-bit address bus which means, it can address upto $2^{20} = 1MB$ memory locations
- 8086 is a 40 pin dual in line package IC

Many other features which we will come across as we study ....

8086 INTERNAL ARCHITECTURE

- 8086 Microprocessor is divided into two independent functional parts
  - Bus Interface Unit (BIU)
  - The Execution Unit (EU)
- This division into two units speeds up processing
- BIU handles all tranfers of data and addresses on the buses for EU
- EU guides BIU about from where to fetch instructions or data, it also decodes and executes instructions

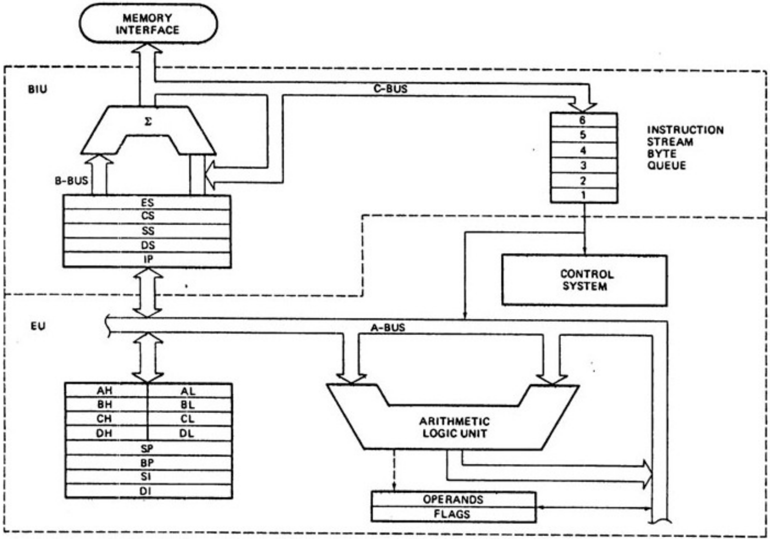
8086 INTERNAL ARCHITECTURE



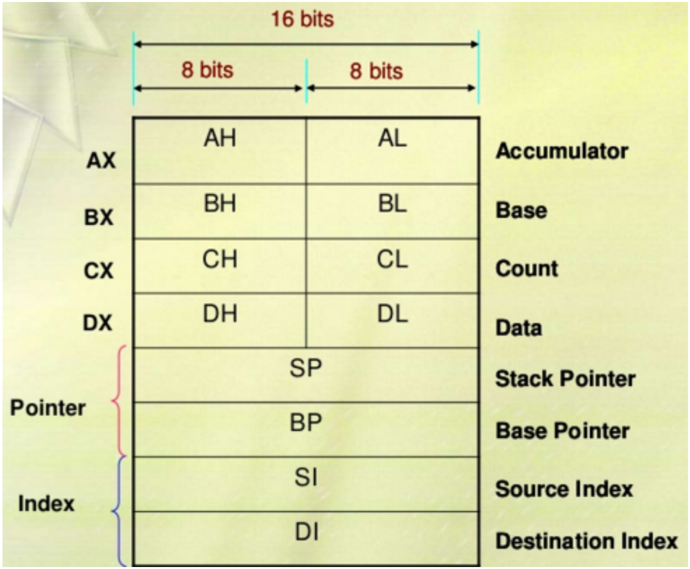
EXECUTION UNIT

- *CONTROL CIRCUITRY*
  - directs internal operations
- *INSTRUCTION DECODER*
  - translates the instructions fetched from memory into series of operations which the EU performs
- *ALU*
  - 16 bit arithmetic logic unit which can add, subtract, AND, OR, XOR, increment, decrement, complement, shift binary numbers etc.

8086 INTERNAL ARCHITECTURE



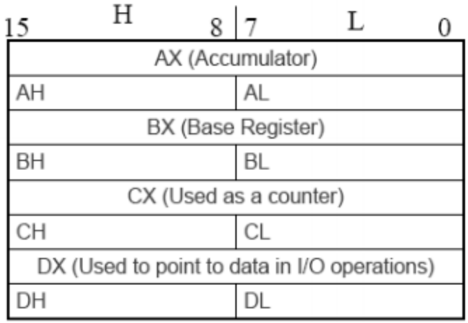
EXECUTION UNIT : General Purpose Registers



EXECUTION UNIT : General Purpose Registers

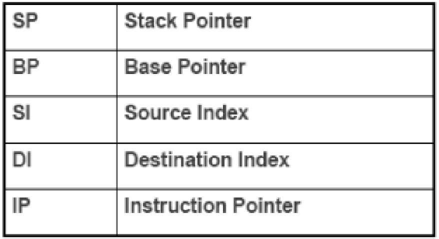
DATA REGISTERS :

- Normally used for storing temporary results
- Each register is 16 bit wide (AX, BX, CX, DX)
- Can be accessed as either 16 bit or 8 bit e.g., AH : upper half of AX , AL: lower half of AX



15 H 8	7 L 0
AX (Accumulator)	
AH	AL
BX (Base Register)	
BH	BL
CX (Used as a counter)	
CH	CL
DX (Used to point to data in I/O operations)	
DH	DL

EXECUTION UNIT : Pointer and Index registers

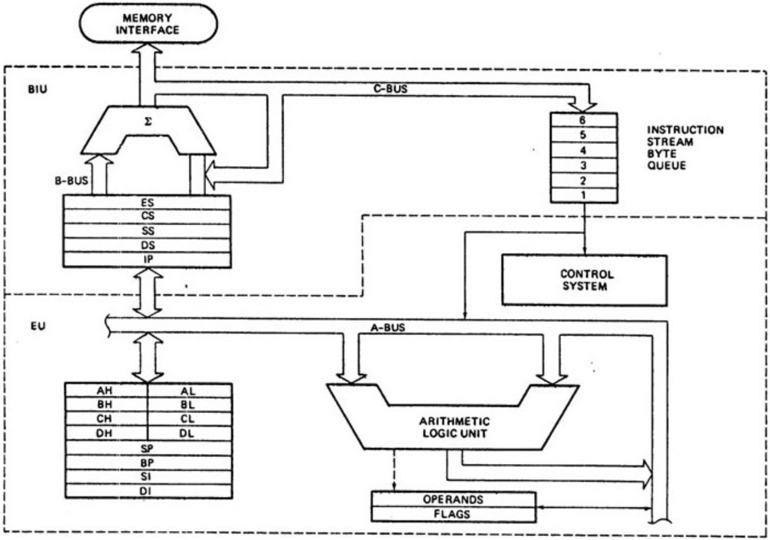


SP	Stack Pointer
BP	Base Pointer
SI	Source Index
DI	Destination Index
IP	Instruction Pointer

- all 16-bit registers
- high and low bytes not accessible

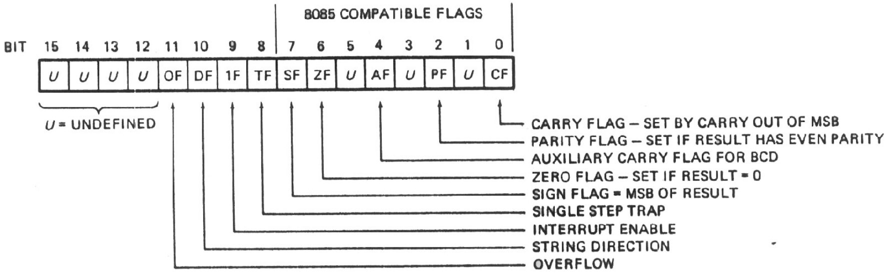
we will be studying about these in detail after a few slides

8086 INTERNAL ARCHITECTURE



EXECUTION UNIT : FLAG register

- 16-bit flag register
- 9 bits are active for flag


8085 COMPATIBLE FLAGS
BIT 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
U U U U OF DF 1F TF SF ZF U AF U PF U CF
U = UNDEFINED
CARRY FLAG – SET BY CARRY OUT OF MSB
PARITY FLAG – SET IF RESULT HAS EVEN PARITY
AUXILIARY CARRY FLAG FOR BCD
ZERO FLAG – SET IF RESULT = 0
SIGN FLAG = MSB OF RESULT
SINGLE STEP TRAP
INTERRUPT ENABLE
STRING DIRECTION
OVERFLOW

EXECUTION UNIT : FLAG register

- These 9 Flags are divided into two types
    - Conditional Flags
    - Control Flags

EXECUTION UNIT : FLAG register

CONDITIONAL FLAGS : represent result of last arithmetic or logical instruction executed.

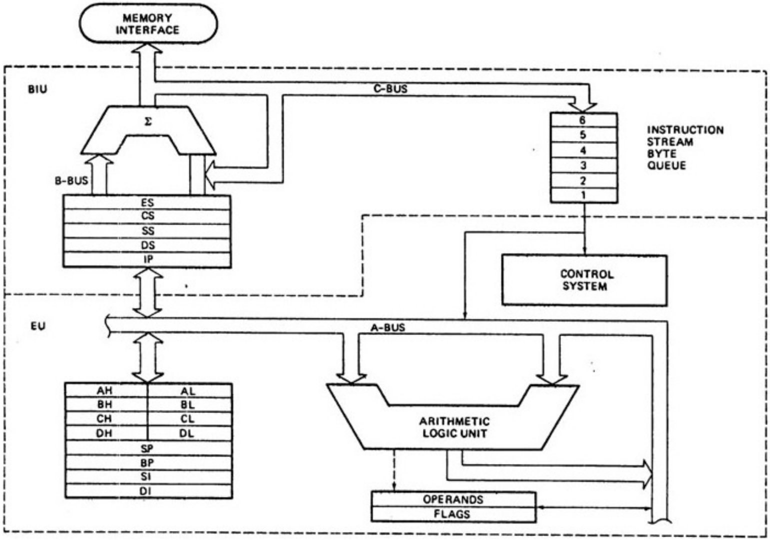
1. CF (Carry Flag): indicates overflow condition, for e.g. addition of two numbers produces a carry out of the MSB position, then CF$= 1$
2. PF (Parity Flag): indicate parity of result. If lower order 8 bits of the result contains even number of 1's; PF $= 1$ else 0.
3. AF (Auxiliary Carry): If an operation performed in ALU generates a carry / borrow from lower nibble (i.e. D0-D3) to upper nibble (i.e. D4-D7), AF$=$1 i.e. carry given by D3 bit to D4 is auxiliary carry.
4. ZF (Zero Flag): Is set if result of arithmetic or logical operation is zero.
5. SF (Sign Flag): In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative , SF $= 1$
6. OF (Overflow Flag): This indicates that result of any operation has exceeded capacity of the machine.

EXECUTION UNIT : FLAG register

CONTROL FLAGS : set or reset by user to control operations of execution unit.

1. TF(Trap Flag): Used for single step control. It allows user to execute one instruction of a program at a time for debugging.
2. IF (Interrupt Flag): Interrupt enable/disable flag. Allows or prohibit the interruption of a program.
3. DF (Direction Flag): Used with string instructions. If DF $= 1$; string bytes are accessed from higher memory address to lower memory address.

8086 INTERNAL ARCHITECTURE



BUS INTERFACE UNIT

Contains:

- 6- byte Instruction Queue
- Segment Registers (CS, DS, ES, SS)
- Instruction Pointer
- Address Summing Block

BUS INTERFACE UNIT

QUEUE

- The BIU uses a mechanism known as an **instruction stream queue** to implement a **pipeline architecture**.
- Both units (EU and BIU) operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as **Pipelining**. This results in efficient use of the system bus and system performance.
- This queue permits pre-fetch of up to 6 bytes of instruction code.
- Whenever the queue of the BIU is not full and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by pre-fetching the next sequential instruction.

BUS INTERFACE UNIT

QUEUE

- These pre-fetching instructions are held in its FIFO queue
- The EU accesses the queue from the output end. It reads one instruction byte after the other from the output of the queue
- The intervals of no bus activity, which may occur between bus cycles are known as Idle state

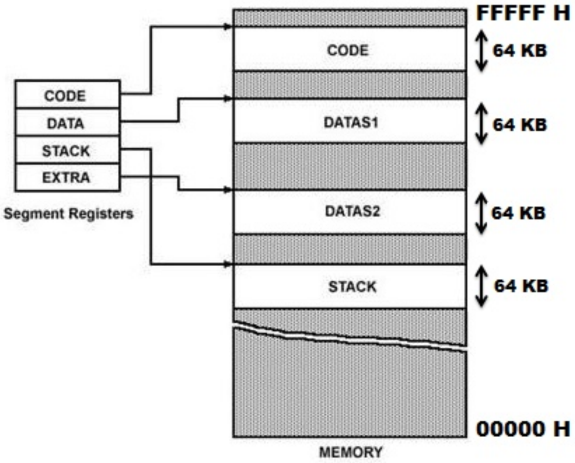
BUS INTERFACE UNIT

MEMORY SEGMENTATION

- 8086 is able to address 1Mbyte of memory (00000H - FFFFFH).The Complete physically available memory may be divided into a number of logical segments.
- The memory in an 8086 based system is organized as segmented memory.
- It is divided into 4 segments of 64KB each
- The segments are :
  - Code Segment
  - Data Segment
  - Stack Segment
  - Extra Segment

BUS INTERFACE UNIT

MEMORY SEGMENTATION



BUS INTERFACE UNIT

MEMORY SEGMENTATION

- A segment may be located any where in the memory
- Each of these segments can be used for a specific function.
  - Code segment is used for storing the instructions
  - The stack segment is used as a stack and it is used to store the return addresses
  - The data and extra segments are used for storing data byte.
- Segments may be overlapped or non-overlapped.
- In assembly language programming more than one Code/Data/Stack segments can be defined. But only one segment of each type can be accessed at a time.

BUS INTERFACE UNIT

SEGMENT REGISTERS

BIU has FOUR 16-bit segment registers. They are:

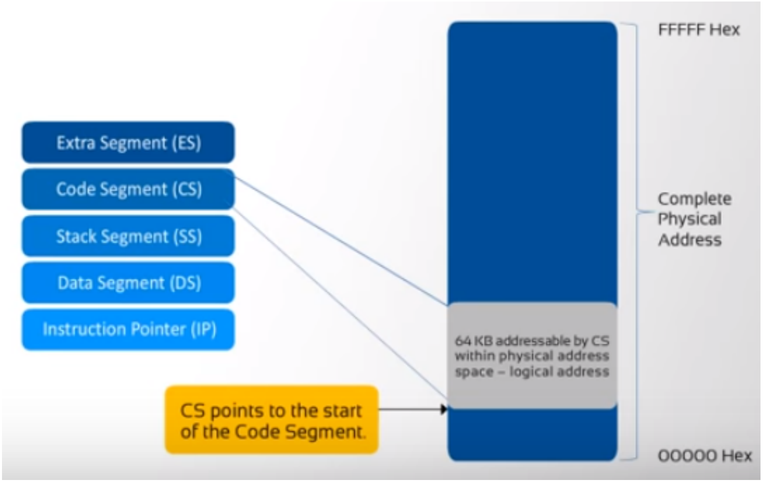
1. Code Segment Register (CS)
2. Data Segment Register (DS)
3. Stack Segment Register (SS)
4. Extra Segment Register (ES)

Segment registers hold upper 16 bits of starting address of four corresponding memory segments.

Each segment register is 16-bit long, so can access $2^{16} =$ *64 KBytes of memory.*

BUS INTERFACE UNIT

MEMORY SEGMENTATION


Extra Segment (ES)
Code Segment (CS)
Stack Segment (SS)
Data Segment (DS)
Instruction Pointer (IP)
CS points to the start of the Code Segment.
64 KB addressable by CS within physical address space – logical address
FFFFF Hex
Complete Physical Address
00000 Hex

INSTRUCTION POINTER (IP)

- Code segment Register (CS) holds what ???
- The IP is a register that holds 16-bit address of the next code byte within this code segment.
- It is incremented automatically, depending on the no. of bytes the instruction has
- Contents of the Instruction Pointer are used for 20 bit memory address calculation.
- It is also known as Offset Address. Offset is the displacement of the memory location from the starting location of the segment.

IP contains distance or offset from base address to the next instruction byte to be fetched.

CALCULATION OF ADDRESS

How is a 20-bit address obtained if there are only 16-bit registers?

- The 20-bit address of a byte is called its Physical Address.
- But, it is specified as a Logical Address
- Logical address is in the form of: **Base Address : Offset**

What is Base Address?

- It is 20-bit starting address of any segment
- To get 20- bit Base address of a segment, 0 H (or 0000) is appended at the lower bit.
- For eg: if CS = 1234 H, To convert this 16-bit address into 20-bit, the BIU appends 0H to the LSBs of the address.
- After appending, the starting address of the Code Segment becomes 12340H.

CALCULATION OF PHYSICAL/EFFECTIVE ADDRESS

How is a 20-bit address obtained if there are only 16-bit registers?

- To calculate the physical address of the memory, BIU uses the following formula
- Physical Address = Starting Address of Segment (Base Address) + Offset (content of IP)
    - To find the starting address of the segment, BIU appends the contents of Segment Register with 0H.
    - Then, it adds offset to it.

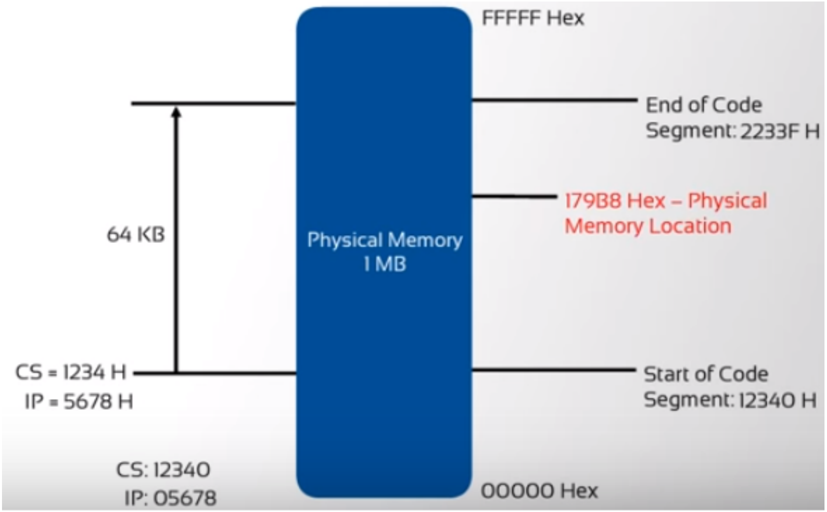
CALCULATION OF PHYSICAL/EFFECTIVE ADDRESS

EXAMPLE:

- If the data at any location has a logical address specified as; **1234 H : 5678 H**
- Then, the number 5678 H is the offset.
- 1234 H is the value of CS Therefore; Physical address =

```
 12340 H
+5678 H
 ______
 179B8 H
```

BUS INTERFACE UNIT


FFFFF Hex
End of Code Segment: 2233F H
179B8 Hex – Physical Memory Location
64 KB
Physical Memory 1 MB
CS = 1234 H
IP = 5678 H
Start of Code Segment: 12340 H
CS: 12340
IP: 05678
00000 Hex

Physical address = 179B8 H

STACK

STACK SEGMENT : Section of 64 Kbyte memory to store addresses and data while a subprogram is executing.

STACK SEGMENT REGISTER : Upper 16-bits of the stack segment are stored in Stack Segment Register (SS).

STACK POINTER :

- Is in the Execution Unit
- 16-bit offset from start of the segment to the memory location where a word was last stored.
- Same as Instruction Pointer

OTHER POINTER AND INDEX REGISTERS

EU has four 16-bit pointer and index registers:

- Stack Pointer
- Base Pointer
- Source Index
- Destination Index

They hold 16-bit offset of a data word in one of the segment.
For example: SI store offset of data word for Data segment.

This was all about the Internal Architecture of 8086 .

Let's have a look at the addressing modes of 8086 .

ADDRESSING MODES

The addressing modes describe the types of operands and the way they are accessed for executing an instruction.

IMMEDIATE ADDRESSING:

- In this type of addressing, immediate data is a part of instruction, and appears in the form of successive byte or bytes.
- Example: MOV AX, 0005H
- In the above example, 0005H is the immediate data. The immediate data may be 8-bit or 16-bit in size

ADDRESSING MODES

DIRECT ADDRESSING:

- In the direct addressing mode, a 16-bit memory address (offset) is directly specified in the instruction as a part of it.
- Example: MOV AX, [5000H]
- Here, data resides in a memory location in the data segment, whose effective address may be computed using 5000H as the offset address and content of DS as segment address.

ADDRESSING MODES

REGISTER ADDRESSING:

- In register addressing mode, the data is stored in a register and it is referred using the particular register. All the registers, except IP, may be used in this mode.
- Example: MOV BX, AX.

ADDRESSING MODES

REGISTER INDIRECT ADDRESSING:

- the address of the memory location, which contains data or operand, is determined in an indirect way, using the offset registers.
- the offset address of data is in either BX or SI or DI registers. The default segment is either DS or ES.
- The data is supposed to be available at the address pointed to by the content of any of the above registers in the default data segment.
- Example: MOV AX, [BX]
- Here, data is present in a memory location in DS whose offset address is in BX.

ADDRESSING MODES

REGISTER RELATIVE ADDRESSING:

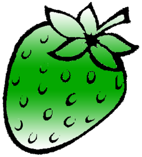
- the data is available at an effective address formed by adding an 8-bit or 16-bit displacement with the content of any one of the registers BX, BP, SI and DI in the default (either DS or ES) segment.
- Example: MOV Ax, 50H [BX]
- Here, effective address is given as 10H x DS + 50H + [BX]

ASSIGNMENT - 1

Study of Instruction set
Prepare a document of all the Instructions of 8086 along with their short description.
NOTE:

- This can be a hand written document or a soft copy.
- Refer any book, internet , anything
- To be submitted on 11-01-2016, Monday

STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com