

Microprocessor Fundamentals Course



The International Congress for global Science and Technology
www.icgst.com

Instructor: Ashraf Aboshosha, Dr. rer. nat.
Engineering Dept., Atomic Energy Authority,
8th Section, Nasr City, Cairo, P. O. Box. 29
E-mail: aboshosha@icgst.com, Tel.: 012-1804952

Lecture Notes

Lecture (4) Microprocessor Architecture and Operation

Course Syllabus:

- Computer Architecture
- Microprocessor structure and Design
- Microprocessor programming and operation
- Microprocessor based Interfacing

This is free educational material.

Microprocessors have two major components

- The Execution unit (EU)
- The Bus interface unit (BIU)

The Execution unit

(EU) is used mainly to execute instructions. It contains a circuit called the ***arithmetic and logic unit (ALU)***. The ALU performs arithmetic (+ , - , * , /) and logic (AND, OR, NOT) operations. The data for operations are stored in circuit called ***Registers***. A register is like a memory location except that we normally refer to it by a name rather than address. The EU has eight registers for storing data; their names are AX, BX, CX, DX, SI, DI, BP, SP and FLAGS register.

Bus interface unit (BIU)

facilitates communication between the EU and memory or I/O circuits. It is responsible for transmitting address, data, and control signals on the buses. Its registers are named CS, DS, ES, SS, IP; they hold addresses of memory locations. The IP contains the address of next instruction to be executed by the EU.

The EU and the BIU

are connected via an internal bus and they work together. While the EU is executing an instruction, the BIU fetches up to six bytes of the next instruction and places them in the instruction queue. This operation is called ***Instruction prefetch***. The purpose is to speed up the processor.

I/O Ports :

I/O devices are connected to the computer through I/O circuits. Each of these circuits contains several register called ***I/O Ports***. Some are used for data while others are used control commands. Like memory locations, the I/O ports have address and they are connected to the bus system. These addresses are known as I/O address and can only be use in input (***IN***) or output (***OUT***) instructions.

Serial and Parallel Interface:

The data transfer between two digital systems can be achieved using single bit at a time (***serial***), or using 8 bit, 16 bit, etc. at a time (***parallel***). The parallel interface requires more wiring connections, while serial port tends to be slower. Slow devices, like the keyboard, or remote interface like internet are always connected via serial interface, while fast devices and directly interfaced devices like the disk drive, always connect via parallel port. But some devices like printers can be connected via either serial or parallel interface.

Registers:

- Data register
- Address register
- Status register
- Index register
- Segment register

Instruction Execution:

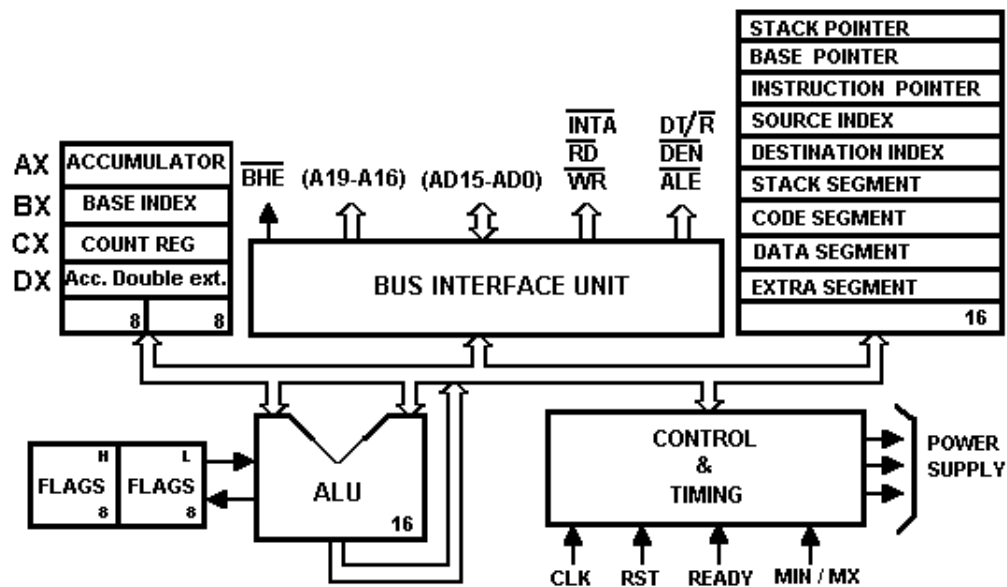
To understand how the CPU operates, let's look at how an instruction is executed. The machine instruction has two major parts: an ***opcode*** and ***operands***. The opcode specifies the type of operation and operands are often given as memory address to the data to be manipulated. The two operation cycles of the microprocessor are: ***(Fetch - Execute cycle)***

Fetch:

1. Fetch an instruction from memory
2. Decode the instruction to determine the operation
3. Fetch data from memory if necessary.

Execute:

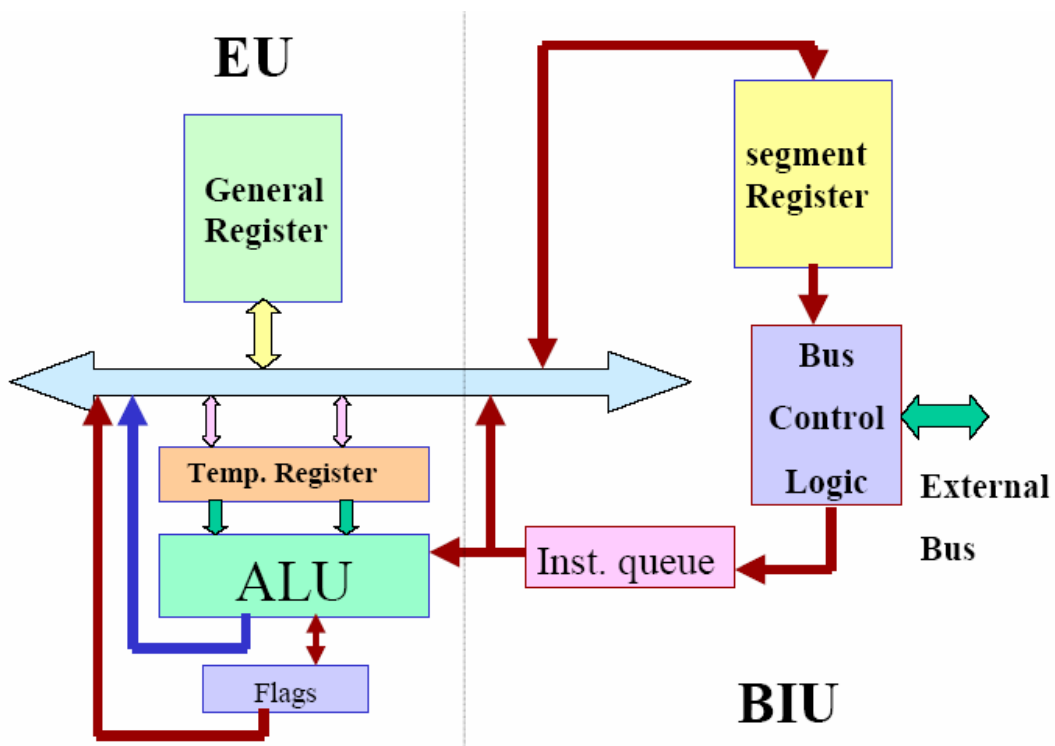
1. Perform the operation on the data
2. Store the result in memory if needed.



8086 BLOCK DIAGRAM

GND	1	40	VCC
AD14	2	39	AD15
AD13	3	38	A16/S3
AD12	4	37	A17/S4
AD11	5	36	A18/S5
AD10	6	35	A19/S6
AD9	7	34	BHE
AD8	8	33	MIN/MAX
AD7	9	32	\overline{RD}
AD6	10	31	HOLD
AD5	11	30	HLDA
AD4	12	29	\overline{WR}
AD3	13	28	$\overline{M/\overline{IO}}$
AD2	14	27	$\overline{DT/\overline{R}}$
AD1	15	26	\overline{DEN}
AD0	16	25	ALE
NMI	17	24	\overline{INTA}
INTR	18	23	\overline{TEST}
CLK	19	22	READY
GND	20	21	RESET

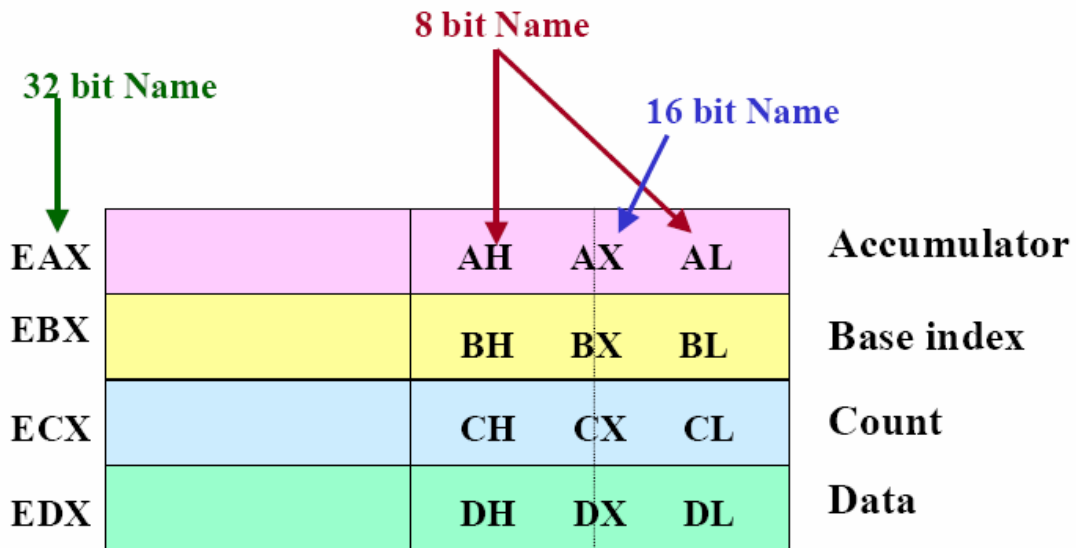
8086 MICROPROCESSOR PIN-OUT MINIMUM MODE



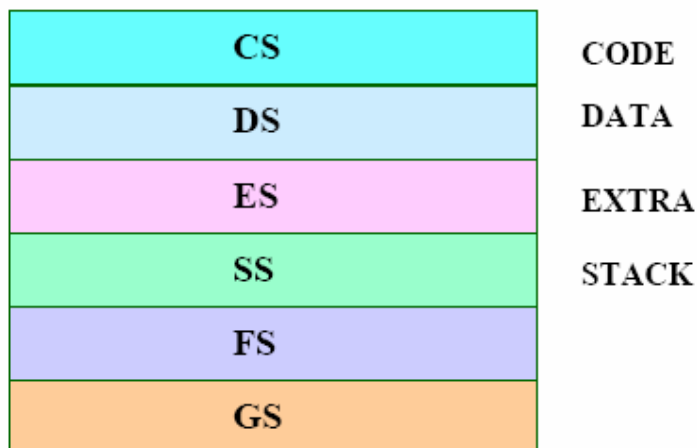
8086 Microprocessor simplified structure

	Internal Bus	External Bus	Memory
MP 8086	16	16	1 MB
MP 8088	16	8	1 MB
MP 80286	16	16	16 MB
MP 80386DX	32	32	4 GB
MP 80386 SX	16	24	16 MB
MP 80486 DX	32	32	4 GB
Pentium	32/64	32/64	4 GB

Intel microprocessors family



8, 16, 32 bit general register structure



Segment Registers

ESP		SP	Stack pointer
EBP		BP	Base pointer
EDI		DI	Destination index
ESI		SI	Source index
EIP		IP	Instruction pointer
FLAGS		FLAGS	FLAGS

32 bit Base & Index Registers

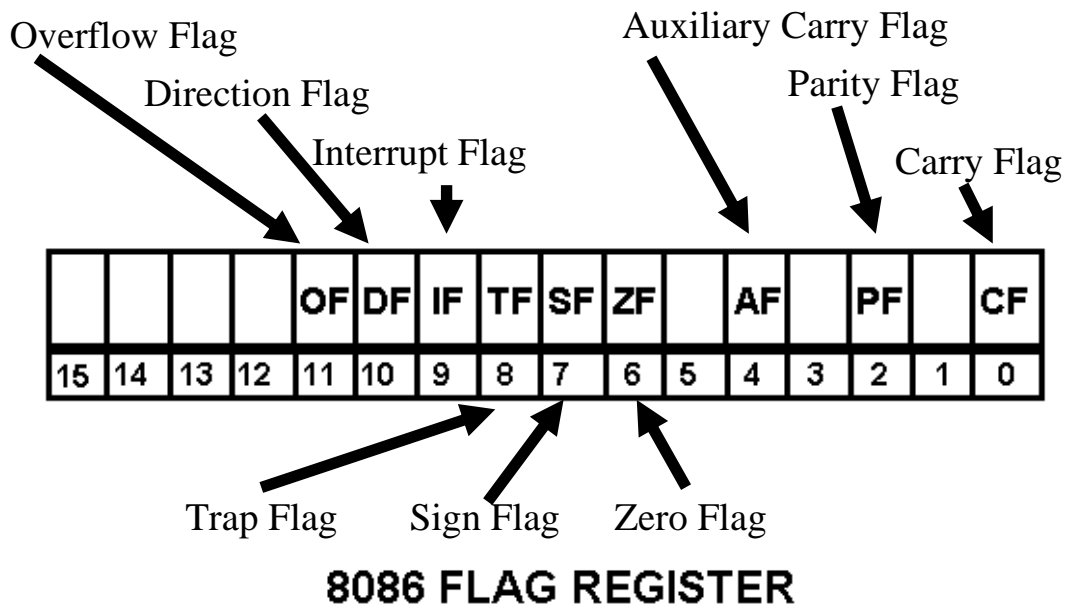
EAX	AH	AX	AL	Accumulator
EBX	BH	BX	BL	Base index
ECX	CH	CX	CL	Count
EDX	DH	DX	DL	Data
ESP		SP		Stack pointer
EBP		BP		Base pointer
EDI		DI		Destination index
ESI		SI		Source index
EIP		IP		Instruction pointer
FLAGS		FLAGS		FLAGS

EU

32 bit EU

HIGH BYTE (8 bit)	GP REGISTERS (16 bit)	LOW BYTE (8 bit)
AH	AX	AL
BH	BX	BL
CH	CX	CL
DH	DX	DL

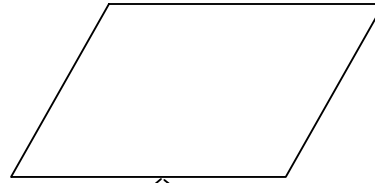
Basic general purpose registers A, B, C and D



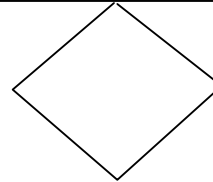
Components of Flowcharts

The first step in writing a program is to draw its flowchart which expresses the logic sequence of this program. The components of flowcharts are:

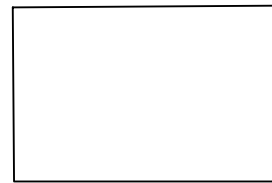
1- Input/Output units



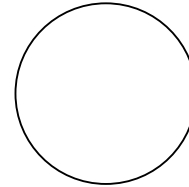
2- Decision



3- Process



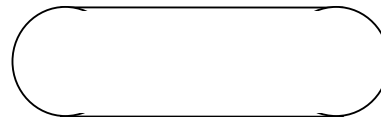
4- Start



5- Subroutines

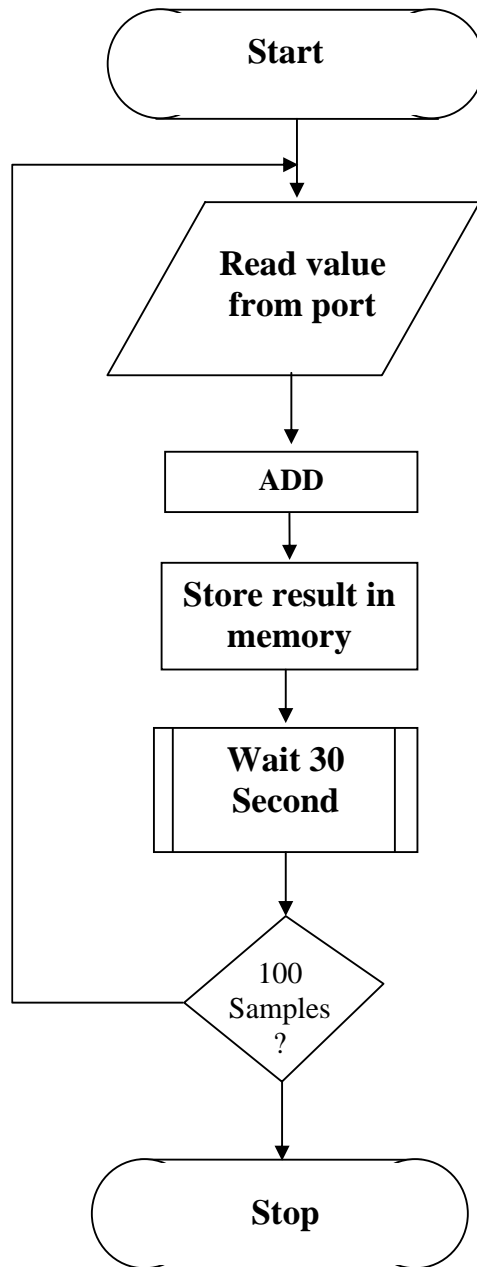


6- Terminator



Ex.:

The following flowchart represents a program that reads 100 value from a port



Finding the Right Instruction

After you get the structure of a program worked out and written down, the next step is to determine the instruction statements required to do each part of the program. Since the examples in this book are based on the 8088 family of microprocessors, now is a good time to give you an overview of the instructions the 8086 has for you to use. First, however, is a hint about how to approach these instructions. You do not usually learn a new language by memorizing an entire dictionary of the language. A better way is to learn a few useful words and practice putting these words together in simple sentences. You can then learn more words as you need them to express more complex thoughts. Likewise, you should not try to memorize all the instructions for a microprocessor at once. As an introduction, however, the few pages here contain a list of all the 8086 instructions with a short explanation of each. Skim through the list and pick out a dozen or so instructions that seem useful and understandable. As a start, look for move, input, output, logical, and arithmetic instructions. Then look through the list again to see if you can write a simple program. You can use the instruction reference as you write programs. Here we simply list the 8086 instructions in functional groups with single-sentence descriptions so that you can see the types of instructions that are available to you. As you read through this section, do not expect to understand all the instructions. When you start writing programs, you will probably use this section to determine the type of instruction and the instruction reference to get the instruction details you need them. After you have written a few programs, you will remember most of the basic instruction types and will be able to simply look up an instruction in the instruction reference to get any additional details you need.

I. Data transfer instructions

PUSH	Copy specified word to top of stack.
POP	Copy word from top of stack to specific location.
PUSHA	(80186/80188 only) Copy all registers stack.
POPA	(80186/80188 only) Copy words f stack to all registers.
XCHG	Exchange bytes or exchange words.
XLAT	Translate a byte in AL using a table in memory.

Simple input and output port transfer instructions:

IN	Copy a byte or word from specified port to accumulator.
OUT	Copy a byte or word from accumulator specified port.

Special address transfer instructions:

LEA	Load effective address of operand in specified register.
LDS	Load DS register and other specified register from memory.
LES	Load ES register and other specified register from memory.

Flag transfer instructions:

LAHF	Load (copy to) AH with the low byte the flag register.
SAHF	Store (copy) AH register to low byte of register.
PUSHF	Copy flag register to top of stack.
POPF	Copy word at top of stack to flag register

II. Arithmetic instructions

ADD	Add specified byte to byte or specified word to word.
ADC	Add byte + byte + carry flag or word + word + carry flag.
INC	Increment specified byte or specified by 1.
AAA	ASCII adjust after addition.

DAA	Decimal (BCD) adjust after addition.
SUB	Subtract byte from byte or word from word.
SBB	Subtract byte and carry flag from byte word and carry flag from word.
DEC	Decrement specified byte or specified word by 1.
NEG	Negate – invert each bit of a specified byte or word and add 1 (form 2's complement).
CMP	Compare two specified bytes or two specified word
AAS	ASCII adjust after subtraction.
DAS	Decimal (BCD) adjust after subtraction.

Multiplication Instructions

MUL	Multiply unsigned byte by byte or unsigned word by word
IMUL	Multiply signed byte by byte or signed word by word
AAM	ASCII adjust after multiplication

Division Instructions

DIV	Divide unsigned word by byte or unsigned double word by word
IDIV	Divide signed word by byte or signed double word by word
AAD	ASCII adjust before division
CBW	Fill upper byte of word with copies of sign bit of lower byte
CWD	Fill upper word of double word with copies of sign bit of lower word

III. Logic and Bit manipulation instructions (AND, OR, XOR)

Logic instructions:

NOT	Invert each bit in a byte or word
AND	AND the content of a byte or a word with another byte or word
OR	OR the content of a byte or a word with another byte or word
XOR	Exclusive OR the content of a byte or a word with another byte or word

Shift instructions:

SHL/SAL	Shift bits of word or byte left, put zero(s) in LSB(s)
SHR	Shift bits of word or byte right, put zero(s) in MSB(s)
SAR	Shift bits of word or byte right, copy old MSB into new MSB

Rotate instructions:

ROL	Rotate bits of byte or word left, MSB to LSB and to CF
ROR	Rotate bits of byte or word right, LSB to MSB and to CF
RCL	Rotate bits of byte or word left, MSB to CF and CF to LSB
RCR	Rotate bits of byte or word right, LSB to CF and CF to MSB

IV. String instructions

A string is a series of bytes or a series of words in sequential memory locations. A string often consists of ASCII character codes. In the list, a "/" is used to separate different mnemonics for the same instruction. Use the mnemonic which most clearly describes the function of the instruction in a specific application. A "B" in a mnemonic is used to specifically indicate that a string of bytes is to be acted upon. A "W" in the mnemonic is used to indicate that a string of words is to be acted upon.

REP	An instruction prefix. Repeat following instruction until CX =0
REPE/REPZ	An instruction prefix. Repeat instruction until CX = 0 or zero Flag ZF!=1
REPNE/REPNZ	An instruction prefix. Repeat until CX = 0 or ZF = 1
MOVS/MOVS/MOVSW	Move byte or word from one string to another
COMPS/COMPSB/COMPSSW	Compare two string bytes or two string words
INS/INSB/INSW	(80186/80188) Input string byte or word from port
OUTS/OUTSB/OUTSW	(80186/80188) output string byte or word to port
SCAS/SCASB/SCASW	Scan a string. Compare a string byte with a byte in AL or a string word with a word in AX
LODS/LODSB/LODSW	load string byte into AL or string word into AX
STOS/STOSB/STOSW	Store byte from AL or word from AX into string

V. Program execution transfer instructions

Instructions are used to tell the 8086 to start fetching instructions from some new address, rather than continuing in sequence.

Unconditional transfer instructions:

CALL	Call a procedure (subprogram), save return address on stack
RET	Return from procedure to calling program
JMP	Go to specified address to get next instruction

Conditional transfer instructions:

A "/" is used to separate two mnemonics which represent the same instruction. Use the mnemonic which most clearly describes the decision condition in a specific program. These instructions are often used after a compare instruction. The terms below and above refer to unsigned binary numbers. Above means larger in magnitude. The terms greater than or less than refer to signed binary numbers. Greater than means more positive.

JA/JNBE	Jump if above/Jump if not below or equal
JAE/JNB	Jump if above or equal/Jump if not below
JB/JNAE	Jump if below/Jump if not above or equal
JBE/JNA	Jump if below or equal/Jump if not above
JC	Jump if carry flag CF = 1
JE/JZ	Jump if equal/Jump if zero flag ZF = 1
JG/JNLE	Jump if greater/Jump if not less than or equal
JGE/JNL	Jump if greater than or equal/Jump if not less than
JL/JNGE	Jump if less than/Jump if not greater than or equal
JLE/JNG	Jump if less than or equal/Jump if not greater than
JNC	Jump if no carry (CF = 0)
JNE/JNZ	Jump if not equal/Jump if not zero (ZF = 0)
JNO	Jump if no overflow (overflow flag OF = 0)
JNP/JPO	Jump if not parity/Jump if parity odd (PF = 0)
JNS	Jump if not sign (sign flag SF=0)
JO	Jump if overflow flag OF=1
JP/JPE	Jump if parity/Jump if parity even (PF = 1)
JS	Jump if sign (SF = 1)

Iteration control instructions:

These instructions can be used to execute a series of instructions some number of times. Here mnemonics separated by a "/" represent the same instruction. Use the one that best fits the specific application.

LOOP Loop through a sequence of instructions until CX= 0
 LOOPE/LOOPZ Loop through a sequence instructions while ZF= 1 and CX != 0
 LOOPNE/LOOPNZ Loop through a sequence instructions while ZF=0 and CX != 0
 JCXZ Jump to specified address if CX=0

Interrupt instructions:

INT Interrupt program execution call service procedure
 INTO Interrupt program execution OF =1
 IRET Return from interrupt se procedure to main program

High-level language interface instructions:

ENTER (80186/80188 only) Enter procedure
 LEAVE (80186/80188 only) Leave procedure
 BOUND (80186/80188 only) Check effective address within specified array bounds

VI. Processor control instructions**Flag set/clear instructions:**

STC Set carry flag CF to 1
 CLC Clear carry flag CF to 0
 CMC Complement the state of the carry flag CF
 STD Set direction flag DF to 1 (decrement string pointers)
 CLD Clear direction flag DF to 0
 STI Set interrupt enable flag to 1 (enable INTR input)
 CLI Clear interrupt enable flag to 0 (disable INTR input)

Execution control instructions:

HLT Halt (do nothing) until interrupt or reset
 WAIT Wait (do nothing) until signal on the test pin is low
 ESC Escape to external coprocessor such as 8087 or 8089
 LOCK An instruction prefix. Prevents another processor from taking the bus while the adjacent instruction executes
 NOP No action except fetch and decode

The assembly instruction format includes Label: which defines the program line, Opcode or mnemonic which describe the required operation, the operand which includes the data and the comment a typical instruction is shown down

Label:	Opcode	Operand1, Operand 2;	comment
--------	--------	----------------------	---------

Assembly instruction format

Assignments:

1. Draw the diagram of 8086 microprocessor structure
2. Draw the basic components of flowcharts; Input/Outputs, Decision, Process, Subroutines and Termination
3. Draw a flowchart that reads the temperature from a sensor and sets the heater voltage in a control system for 8 hours.
4. Describe the operations of the Following instructions MOV, ADD, AND, ROR, CALL, RET, HLT
5. Draw and describe the flag register