MICRO PROCESSOR AND INTERFACING TECHNIQUES

DIGITAL (LAB) ASSIGNMENT - 1

NAME : MOTHILSHWARAN.C

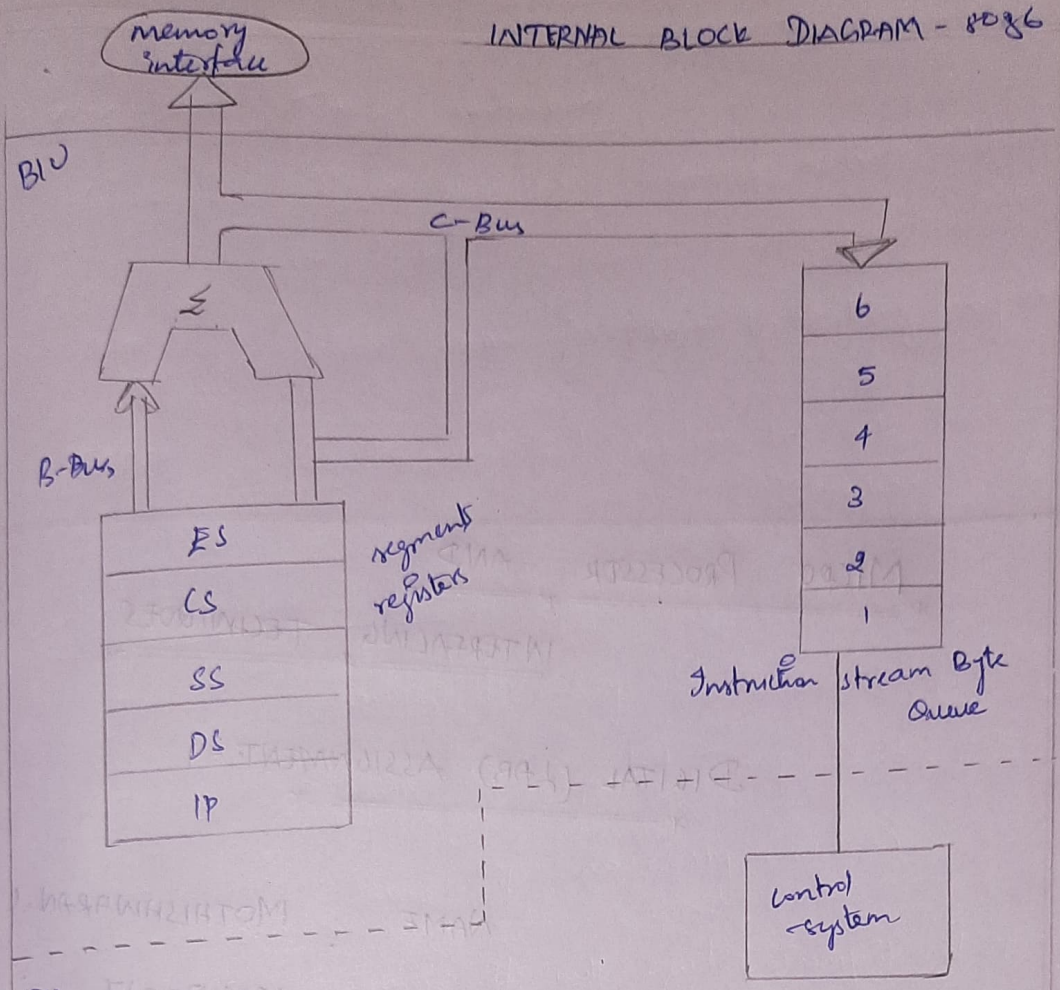
REG NO : 19MID0017

COURSE CODE : CSI2006

SLOT : L53+L54

FACULTY : Dr. NARESH.K

INTERNAL BLOCK DIAGRAM - 8086



ARCHITECTURE OF 8086

As 8086 does 2 stage pipelining its architecture is divided into two units

1) Bus Interface Unit (BIU)
2) Execution Unit (EU)

BUS INTERFACE UNIT:

* It provides the interface of 8086 to other devices
* It operates with respect to Bus cycle
* It performs various machine cycle such as Mem. Read, I/O write etc to transfer data with Memory and I/O devices.
* It performs the following functions:
  - → It generates the 20 bit physical address for memory address
  - → Fetches instruction from memory
  - → Transfer data to and from the memory or I/O.
  - → Supports pipelining using 6 byte instruction queue

Execution Unit:

* It fetches instruction from the queue in BIU decodes and executes them
* It performs arithmetic, logic and internal data transfer operations.

* It sends request signals to the BIU to access the external module

* It operates w.r.t to clock cycles (states)

WORKING FLOW OF 8086

* BIU outputs the content of instruction pointer onto the address Bus. This will cause selected byte or word to be read into BIU

* Instruction pointer (IP) is incremented by 1 to prepare for the next instruction fetch

* Once fetched inside BIU, it is passed to the Queue (6 instruction)

* Execution Unit (EU) draws this instruction from the Queue and begins execution

* While EU is executing the current instructions, the ~~BIU~~ BIU proceeds to fetch new instructions.

Three conditions that will cause the 'EU' to enter 'WAIT MODE'

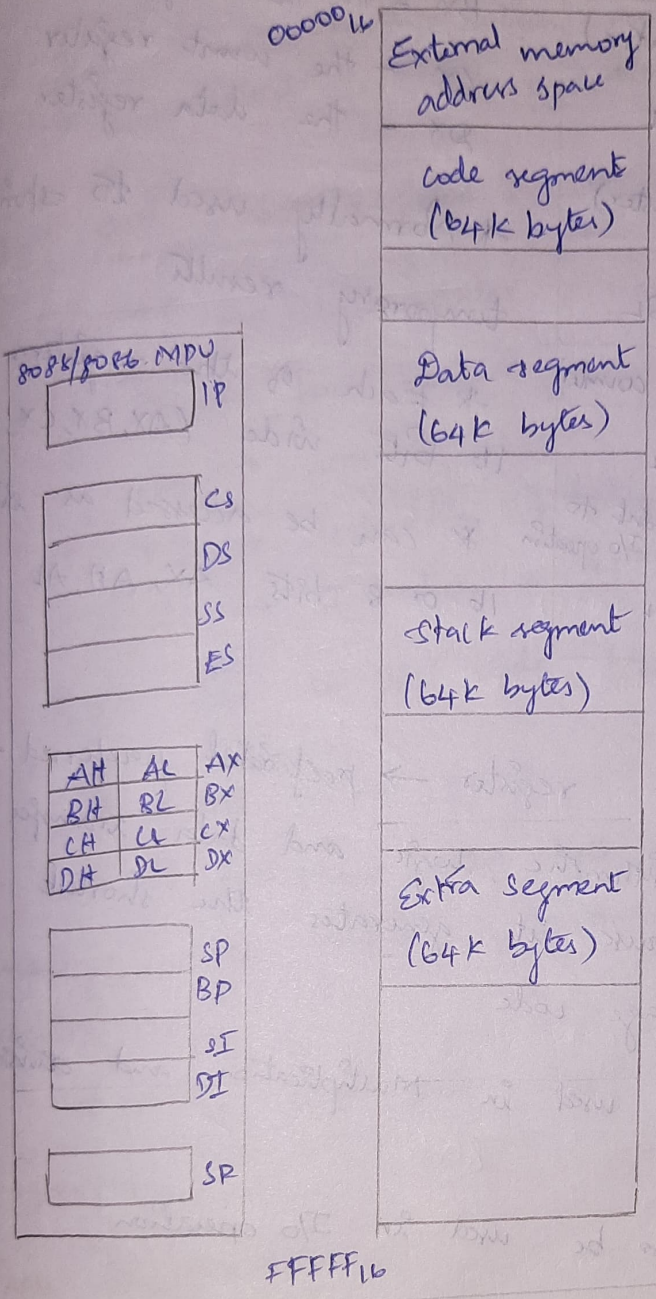
(1) When an instructions requires access to a memory location not in queue

(2) Jump instruction executed [Branching Instruction]

(3) Slow to execute

eg: AAM → AASCII Adjust multiplication
↳ requires 83 clock cycles to complete

Software Model of MPU 8086



8086 Registers

General purpose

- AX: AH | AL
- BX: BH | BL
- CX: CH | CL
- DX: DH | DL

Stack and control

- Flags
- IP

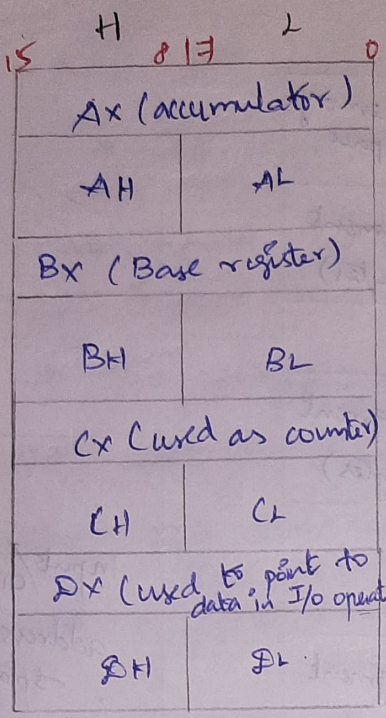
Index

- BP
- SP
- SI
- DI

segment

- CS
- SS
- DS
- ES

GENERAL PURPOSE REGISTER:



AX - the Accumulator
BX - the Base register
CX - the count register
DX - the data register

* Normally used to storing temporary result
* Each of the registers is 16 bits wide (AX, BX, CX, DX)
* can be accessed as either 16 or 8 bits AX, AH, AL

AX:

* Accumulator register → preferred register to use in arithmetic, logic and data transfer instructions because it generates the shortest machine language code
* Must be used in Multiplication and division operations
* Must also be used in I/o operation

BX:

* Base register
* Also serves as an address register

CX:

* count register
* used as loop counter
* used in shift and rotate operations

DX:

* Data register
* Used in multiplication and division
* Also used in I/O operations

Pointer and Index register:

SP → stack pointer
BP → base pointer
SI → source Index
DI → destination Index
IP → Instruction pointer

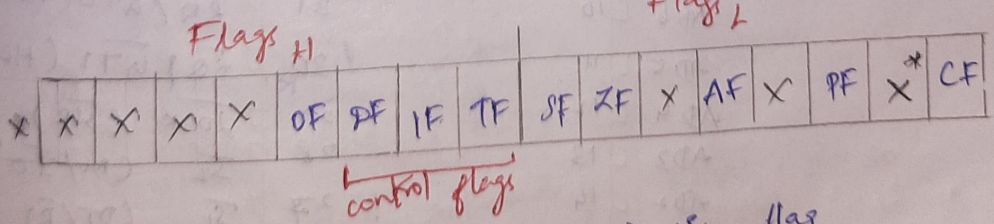
* All are 16 bits wide, L/H bytes are not accessible

eg: MOV AH, [SI]

→ Move the byte stored in memory location whose address is contained in register to register AH

* IP is not under direct control of the programmer

FLAG REGISTERS:



OF → overflow flag
DF → direction flag
IF → interrupt enable flag
TF → Trap flag

SF → sign flag
ZF → zero flag
AF → auxillary carry flag
PF → parity flag
CF → carry flag

* 6 are status flags and 3 are control flags

8086 Programmer's Model

BIU registers

Extra segment
Code segment
Stack segment
Data segment
Instruction pointer

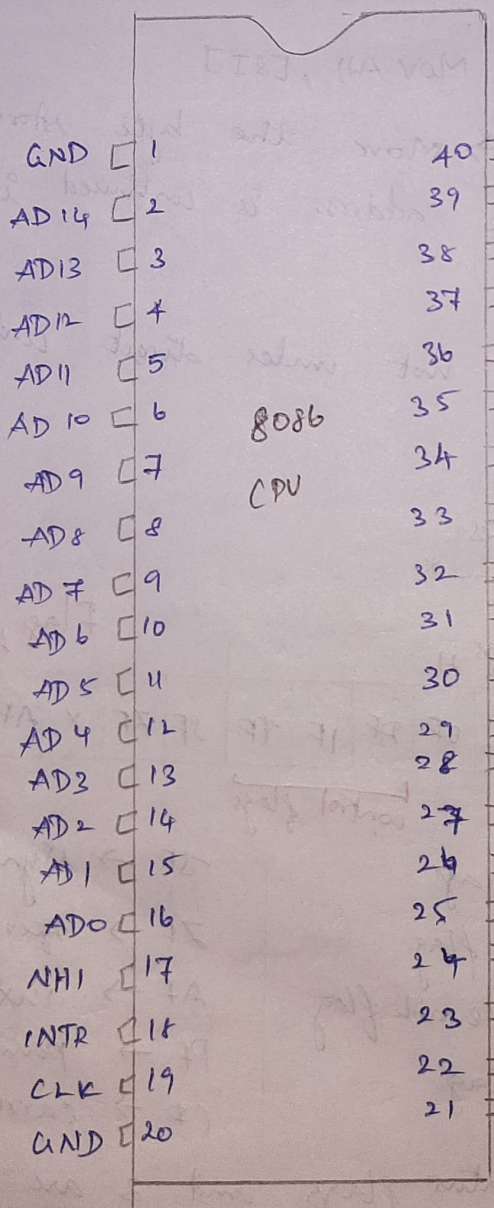
(20 bit address)

EU registers

AH	AL	Accumulator
BH	BL	Base register
CH	CL	Count register
DH	DL	Data register
SP		Stack pointer
BP		Base pointer
SI		Source index ptr
DI		Destination index ptr
FLAGS		

INTEL 8086 PIN DIAGRAM

Max MODE & MIN MODE.



* Ground : Pin 1 and Pin 20 (GND)

* Clock : Pin 19, duty cycle (33%) (CLK)

* Powercycle : Pin 40, 5V ± 10 %. (VCC)

* Reset :
  - Pin 21 (RESET)
  - Registers, segments, flags
  - CS : FFFFH
  - IP : 0000H

* Address latch Enable : Pin 25 (ALE)
  when high, multiplexed address/data bus contains address information

* Address/Data bus : Pin 2 - Pin 16 and Pin 39 (AD0 - AD15)
  → contains address bits A_{15} - A_0, when ALE is 1
  → contains data bits D_{15} - D_0, when ALE is 0.

* Interrupt : Non markable interrupt (NMI) → pin 17
  Interupt Request (INTR) → pin 18
  Interrupt Acknowledge ($\overline{INTA}$) → pin 24

* Hold : Direct memory access purpose
  Hold (HOLD) - pin 31
  Hold acknowledge (HLDA) - pin 30

* Address/status Bus : Pin 35 - Pin 38 (A16 - A19 and S_3 - S_6)
  Address bits A19 - A16
  status bits S_6 - S_3
  S_6 - logic 0

S_5 - indicates condition of IF flag bit

S_4, S_3 - indicates which segment is accessed during current Bus cycle

S_4	S_3	function
0	0	Extra segment
0	1	stack segment
1	0	Code (or) no segment
1	1	Data segment

* Bus High Enable/S_7 : • Pin 34 ($\overline{BHE}/S_7$)
• Enables most significant data bits $D_{15} - D_8$ during read (or) write operation
S_7 - always 1
• $\overline{BHE}$ #, A_0 :
0,0 : whole word (16-bits)
0,1 : high byte to/from odd address
1,0 : low byte to/from even address
1,1 : No selection.

* Min/Max mode : • Pin 33 (MN/$\overline{MX}$)
• Maximum mode : 0V
• Minimum mode : +5V
• Minimum mode Pins → Pin 24-31

* Read signal : Pin 32 ($\overline{RD}$)

* Write Signal : Pin 29 ($\overline{WR}$)

* Memory (or) I/O : Pin 28 ($M/\overline{IO}$)

* Data Transmit/Receive : Pin 27 ($DT/\overline{R}$)

* Data Bus enable : Pin 26 ($\overline{DEN}$)

* status signal : Pin 26 to Pin 24 ($\overline{S_0}, \overline{S_1}, \overline{S_2}$)

I/p to 8288 to generate eliminated signals due to max. mode

S_2	S_1	S_0	
0	0	0	- INTA
0	0	1	- read I/o port
0	1	0	- write I/o port
0	1	1	- halt
1	0	0	- code access
1	0	1	- read memory
1	1	0	- write memory
1	1	1	- none passive

* Used to Lock

* Lock output : Used to lock pheripherals of the system Activated by using the Lock: prefix on any instruction

Lock output ($\overline{LOCK}$) - pin 29

DMA request/Grant - Pin 30 & pin 31

* Queue stat/c : Pin 24 and Pin 25

used by numeric co-processor (8087)

QS_1	QS_0	
0	0	- Queue is idle
0	1	- first byte of opcode
1	0	- Queue is empty
1	1	- subsequent byte of opcode

ADDRESSING MODES:

* The way in which the processor gets data from the user is called Addressing mode
* It can be get data from different sources.
  - → registers
  - → by information
* We can also refer addressing mode as the way in which operand of an instruction is specified

DIFFERENT ADDRESSING MODE:

8086 provides different addressing modes for Data, program and stack memory.

Addressing modes for Data memory:

(i) Implied Addressing Mode:

* The operand is specified in the instruction field itself: The 8 bit (or) 16 bit data is a part of instruction
* zero address instruction are formed with implied addressing mode

eg:

* CLC - The instruction resets the carry flag to zero
* STC - sets the carry flag
* CLD - clears the direction flag

(ii) Immediate Addressing mode:

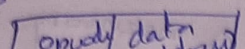
In this mode, data is placed in address field of instruction. Designed as one address instruction format

eg:

MOV AL, 35H ← moves data 35 to AL register

MOV BX, 1234H ← moves 1234 immediately to BX register

Instruction:



↑ data is directly stored as operand

(iii) Register mode:

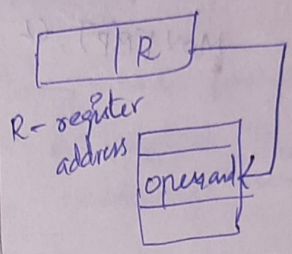
In this mode, operand is placed in 8 bit or 16 bit register. The register is specified in the instruction.

eg:

MOV AX, CX ← moves the data of CX register into AX register

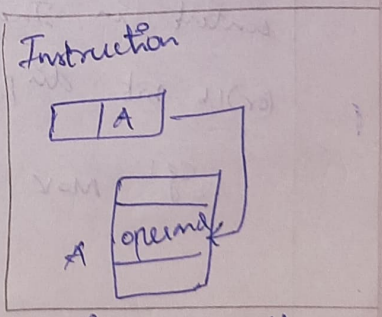
MOV CL, DL ← moves data of DL into CL register

Instruction:



(iv) Direct Addressing Mode: (Absolute)

Operands address is given in the instruction on as 8 bit. In the mode the 16 bit address of data is a part of instruction



Eg: ADD, AL, [0301] → Add the data of offset address 301 to the content of AL

MOV CL, [4321H]: Moves data from location 4321H in the segment into CL

The physical address is calculated as DS * 10_H + 4321

Assume DS = 5000H, PA = 5000 * 10 + 4321

= 54321

CL = [54321H]

(V) Indirect Addressing Mode:

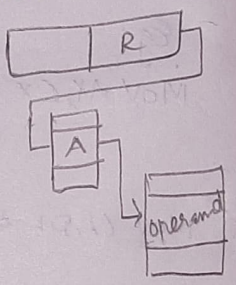
a) Register Indirect Addressing Mode:

The data (operand) will be placed in the memory location. The address of memory will be placed in register. The register will be part of instruction.

Eg: MOV AX, [BX] → Move the contents of memory location address placed is register BX to register AX.

MOV [BP], CL → Moves a byte from CL into the location pointed by BP in stack segment

Instruction:

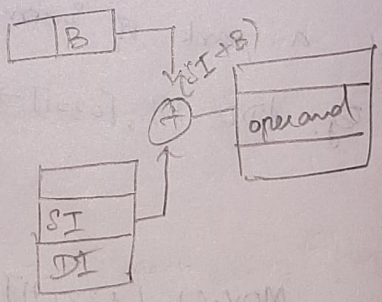


b) Indexed Addressing Mode

Operands address is the sum of the content in Index register SI or DI and an 8 bit (or) 16 bit displacement

Eg: MOV AX, [SI + 05]

Instruction:

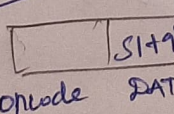


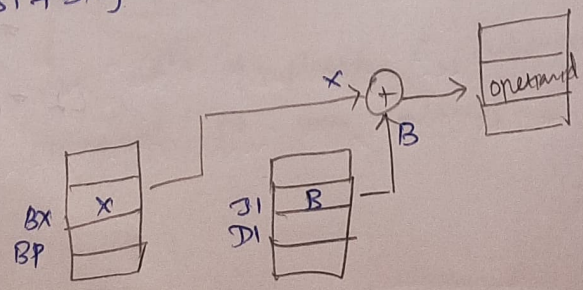
c) Base Addressing Mode:

Effective address is obtained by adding base register value to index register SI or DI (BX) or (BP)

Eg: ADD AX, [SI + BX]

Instruction:





DATA TRANSFER INSTRUCTIONS:

General purpose byte or word transfer instruction.

MNEMONIC	DESCRIPTION
MOV	copy byte or word from specified source to specified destination
PUSH	copy specified word to top of stack
POP	copy word from top of stack to specified location
PUSHA	(80186/80188 only) copy all registers to stack
POPA	(80186/80188 only) copy words from stack to all registers
XCHNG	Exchange bytes or exchange words
XLAT	Translate a byte into AL using a table in memory

Simple I/O and O/p port transfer instructions:

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

Special Address transfer Instruction:

LEA	Load effective address of operand into specified register
LDS	load DS register and other specified register from memory
LES	Load ES register and other specified register from memory
Flag	transfer instruction

LAHF — Load (copy to) AH with low byte of the flag register

SAHF — Share (copy) AH register to low byte of flag register

PUSHF — copy flag register to top of stack

POPF — copy word at top of the stack to flag register

ARITHMETIC INSTRUCTION:

Addition instruction:

ADD — Add specified byte or specified word to word

ADC — Add byte + byte + carry flag or word + word + carry flag.

INC — Increments specified byte or specified word by 1

AAA — ASCII adjust after addition

DAA — Decimal (BCD) adjust after addition.

Subtract instruction:

SUB — Subtract byte from byte or word from word

SBB — Subtract byte and carry flag from byte or word and carry flag word

DEC — Decrements specified byte or specified word by 1

NEG — (Negative) Negate or invert each of bit of a specified byte or word and add 1 (from 2's complement)

CMP — compare 2 specified bytes or 2 specified word

AAS — ASCII adjust after subtraction

DAS — Decimal (BCD) adjust after subtraction.

Multiplication instruction:

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 instruction:

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 after division

CBW — Fill upper byte of word with copies of sign bit of lower bit

CWD — Fill upper word by double word with sign bit of lower word

BIT MANIPULATION INSTRUCTION:

logical instruction:

NOT — invert each bit of byte or word

AND — AND each bit in a byte or word with the corresponding bit in another byte or word

OR — OR each bit in a byte or word with the corresponding bit in another byte or word

XOR — Exclusive OR each bit in a byte or word with the corresponding bit in another byte or word

TEST — AND operands to update flags, but doesn't change operands

Shift instruction:

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

Rotate instruction:

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

STRING INSTRUCTION:

REP	An instruction prefix, Repeat following instruction until CX=0
REPE/REPZ	An instruction prefix, repeat instruction until CX=0, zero flag ≠1
INS/INSB/INSW	(80186/80188) I/P string byte or word from port
OUTS/OUTSB/OUTSW	output string byte/word to port

PROGRAM EXECUTION TRANSFER INSTRUCTION:

Unconditional transfer instruction:

CALL	call a procedure save a return address on stack
RET	Return from procedure to calling program
JMP	Go to specified address to get next instruction

Control transfer Instruction:

JC	jump if carry CF = 1
JE/JZ	jump if equal / Jump if ZF = 1
JNC	Jump if no carry (CF = 0)
JNE/JNZ	jump if not equal / Jump if not ZF = 0
JNO	Jump if no overflow flag = 0

Iteration control instruction:-

Loop	Loop through a sequence of instruction with CF = 0
LoopNE/LoopNZ	while ZF = 0 & CX ≠ 0

Interrupt instruction:

INT	interrupt program instruction execution, call service procedure
INTO	interrupt program execution if OF = 1

PROCESSOR CONTROL INSTRUCTION:

Flag set/clear instruction:

STC	Set CF to 1

CLC	set CF = 0
CLD	Clear DF to 0
STD	set DF to 1

External Hardware Synchronization Instruction:

HLT	Halt until interrupt (or) reset
WAIT	wait until signal on the TEST pin low
ESC	Escape to external co-processor
Lock	prevents another processor from taking the bus while the adjacent instruction executes

No operation instruction:

NOP	no action except fetch & decode