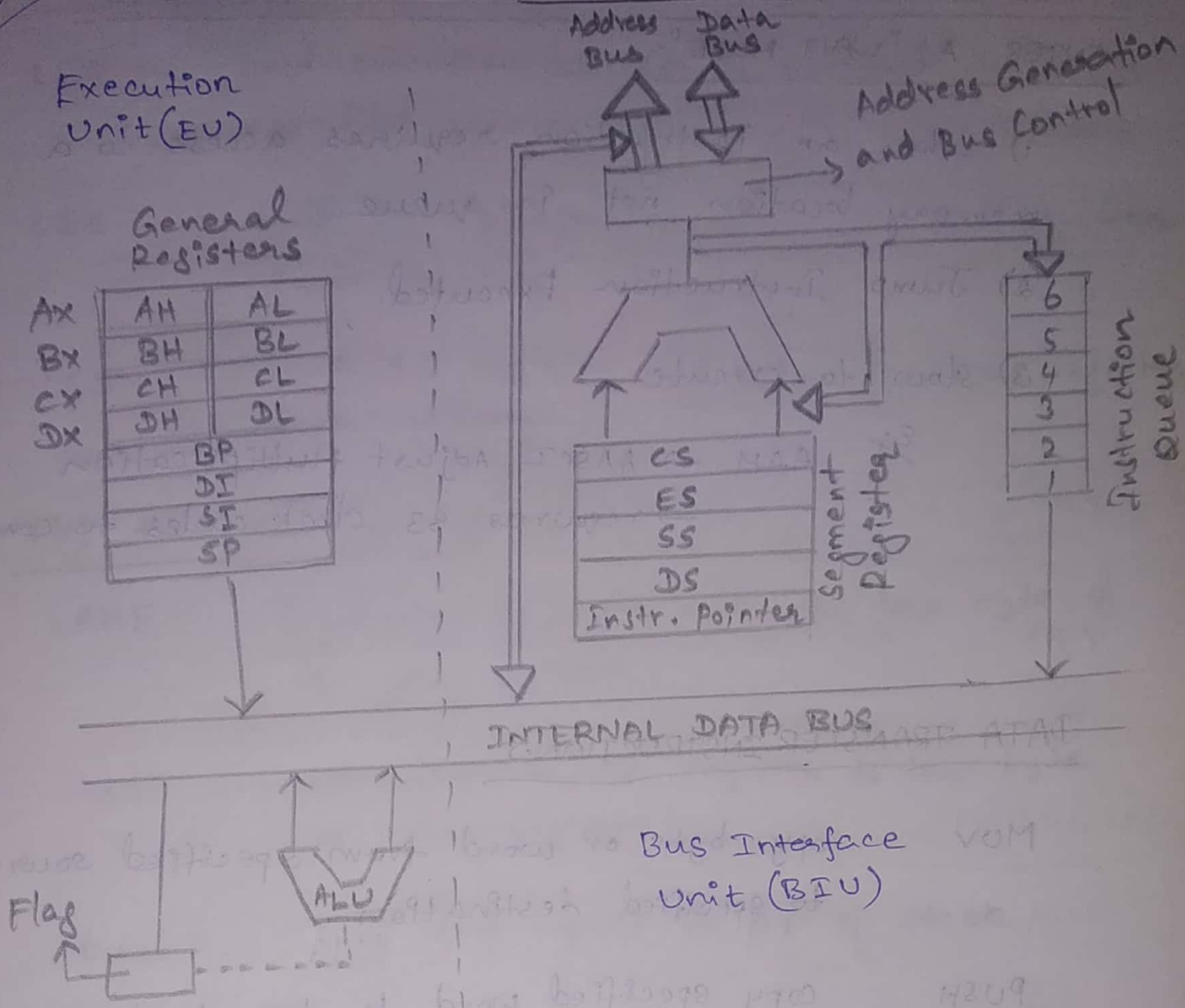
05/01/2022

INTERNAL BLOCK DIAGRAM - 8086



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

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

3) Once fetched inside BIU, it is passed to the Queue.

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

5) While EU is executing the current instruction, the BIU proceeds to fetch new instruction

THREE CONDITIONS THAT WILL CAUSE THE 'EU' TO ENTER A "WAIT MODE"

1) When an instruction requires access to a memory location not in queue.

2) Jump Instruction Executed.

3) slow to execute

Eg AAM → AASCI Adjust Multiplication
→ requires 83 clock cycles to complete

8086 REGISTERS

General purpose			Index	Segment	Status and control
AX	AH	AL	BP	CS	Flags
BX	BH	BL	SP	SS	IP
CX	CH	CL	SI	DS	
DX	DH	DL	DI	ES	

General Purpose Registers:

AX (Accumulator)	
AH	AL
BX (Base Register)	
BH	BL
CX (used as counter)	
CH	CL
DX (used to point data in IO operation)	
DH	DL

* Normally used for storing temporary results
* Each of the registers is 16 bit 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 instruction because it generates the shortest Machine Language Code.
  - Must be used in Multiplication and division operations
  - Must also be used in I/o operations

- BX
  - Base Register
  - Also serves as an address Register

- CX
  - count register
  - used as a 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 Registers

- All 16 bits wide, L/H bytes are not accessible
- Used as memory pointers

  Example

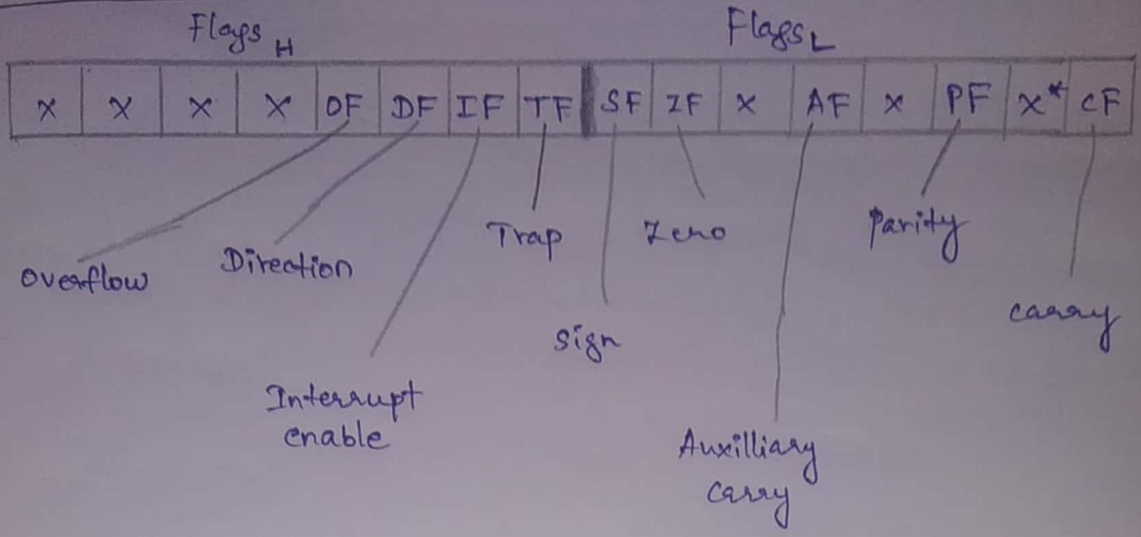
  MOV AH, [SI]

  ⇒ Move the bytes stored in memory location whose address is contained in register SI to register AH

- IP is not under direct control of the programmer

SP	Stack pointer
BP	Base pointer
SI	Sourse Index
DI	Destination Index
IP	Instruction Pointer

Flag Register



6 status flags and 3 control flags

8086 Programmer's Model:

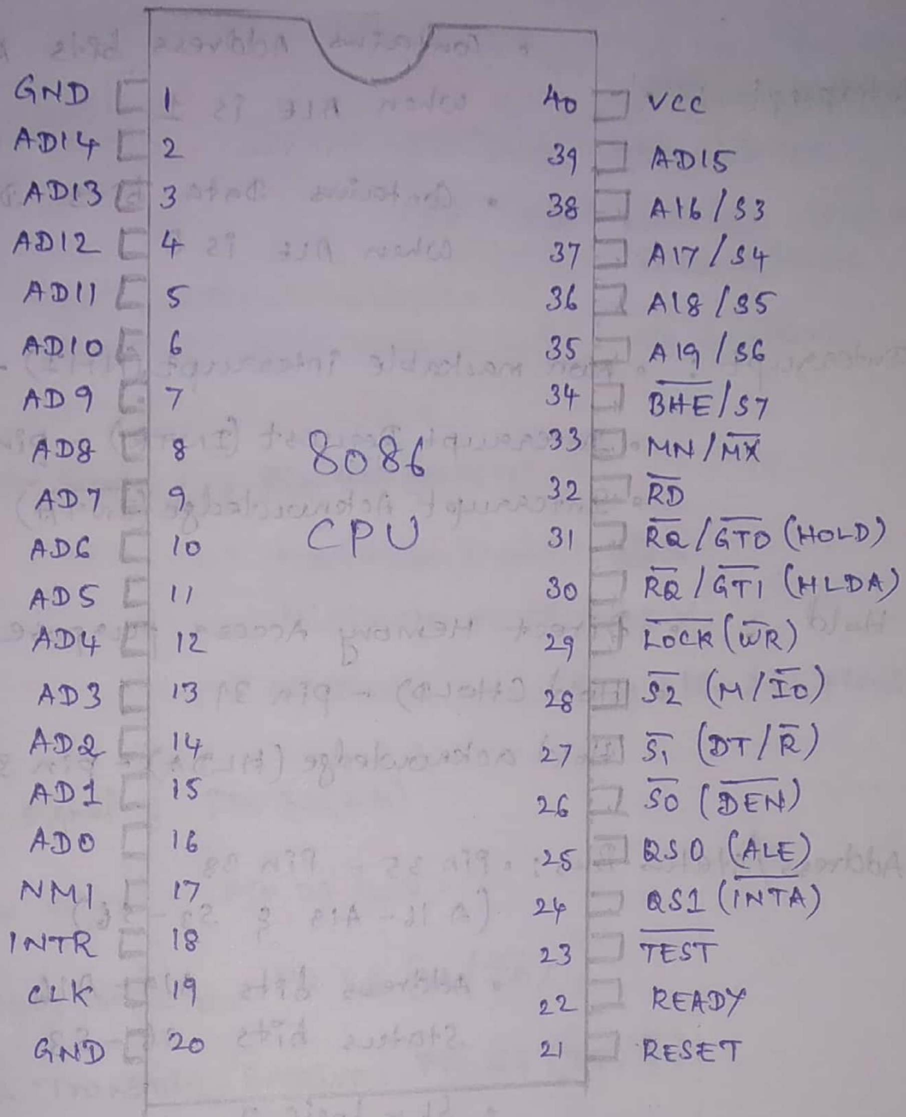
BIU registers

(20 bit adder)

ES	Extra segment
CS	Code segment
SS	stack segment
DS	Data Segment
IP	Instruction pointer

AX	AH	AL	Accumulator
BX	BH	BL	Base Register
CX	CH	CL	Count Register
DX	DH	DL	Data Register
	SP		stack pointer
	BP		Base pointer
	SI		Source Index Registers
	DI		Destination Index Register
	FLAGS		

INTEL 8086 - PIN DIAGRAM



* Ground : Pin 1 and Pin 20 (GND)

* clock : Pin 19, Duty cycle : 33% (CLK)

* power cycle : 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 : •~~

* Interrupt : • Non maskable interrupt (NMI) - pin 17
  • Interrupt 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 & S3 - S6)
  • Address bits A19 - A16
    Status bits S6 - S3
  • S6 - Logic 0
    S5 - Indicates condition of IF flag bits
    S4, S3 - Indicates which segment is accessed during current bus cycle
  •

S4	S3	Function
0	0	Extra segment
0	1	stack segment
1	0	code (or) no segment
1	1	Data segment

* Bus High Enable/S7 : • Pin 34 ($\overline{BHE}$/S7)
  - Enables most significant data bits $D_{15} - D_8$ during read (or) write operation
  - S7 — always 1
  - <u>BHE#, A0</u> :
    - 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 : 0 V
    Minimum mode : +5 V
  - Minimum mode Pins → Pin 24 - Pin 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 - Pin 28 ($\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$)
  - Inputs to 8288 to generate eliminated signals due to max mode
  - <u>S2 S1 S0</u>

000	- INTA
001	- read I/O port
010	- write I/O port
011	- halt
100	- Code access
101	- read memory
110	- write memory
111	- none - passive

* Lock Output : • Used to lock pheeripherals off 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 Status : • Pin 24 & Pin 25

• Used by numeric co-processor (8087)

• QS1 QS0

00 - Queue is idle

01 - First byte of opcode

10 - Queue is empty.

11 - Subsequent byte of opcode

Addressing Modes

* The way in which the processor gets data from the user is called Addressing mode
* It can get data from different sources
  → register
  → by Instruction
* we can also refer addressing mode as the way in which the operand of an Instruction is specified

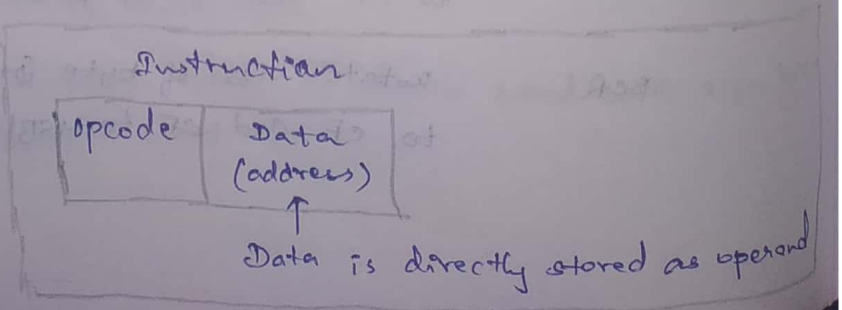
DIFFERENT ADDRESSING MODES

Implied Addressing Mode

* The operand is specified in the instruction itself
* The 8 bit (or) 16 bit data is a part of instruction
* Zero address instruction are formed with implied Addressing mode
* Example:
  CLC - The instruction resets the carry flag to zero

Immediate Addressing Mode

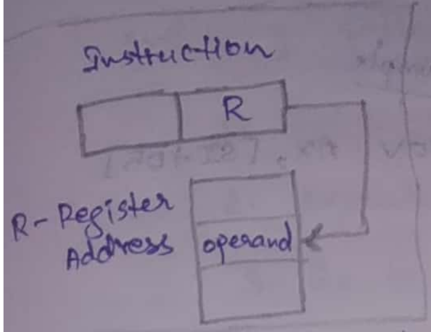
* In this mode, data is placed in address field of Instruction.
* Designed as one address instruction format
* Example
  MOV AL, 35H ⇒ The instruction moves data 35 to AL Register



Register Mode:

* In this mode, operand is placed in 8-bit (or) 16-bit register.

* the register is specified in the instruction



Example

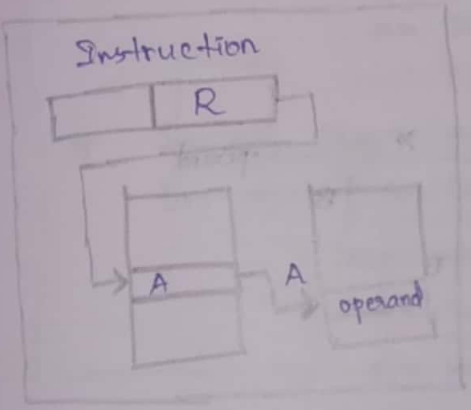
MOV AX, CX => The instruction moves the contents of CX register to AX register

Register Indirect 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



Example

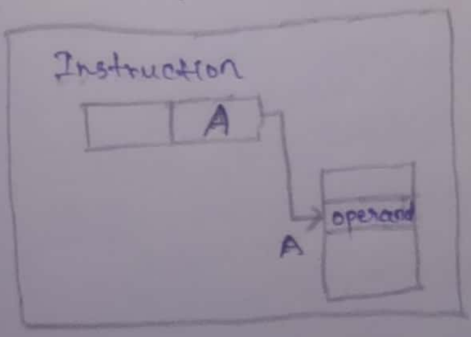
MOV AX, [BX]

=> Move the contents of memory location's address placed in Register BX to register AX

Direct Addressing Mode (Absolute)

* operands address is given in the instruction as an 8-bit

* In the mode the 16-bit address of data is a part of instruction



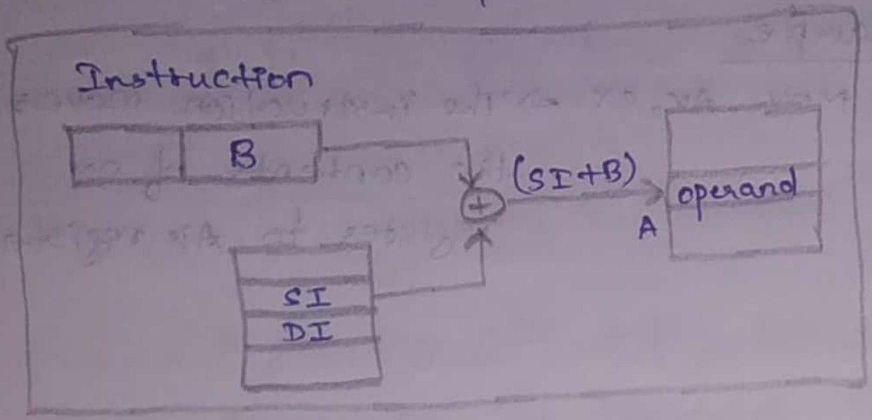
Example

ADD AL, [0301]

=> add the contents of offset address 30L to the contents of AL .

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



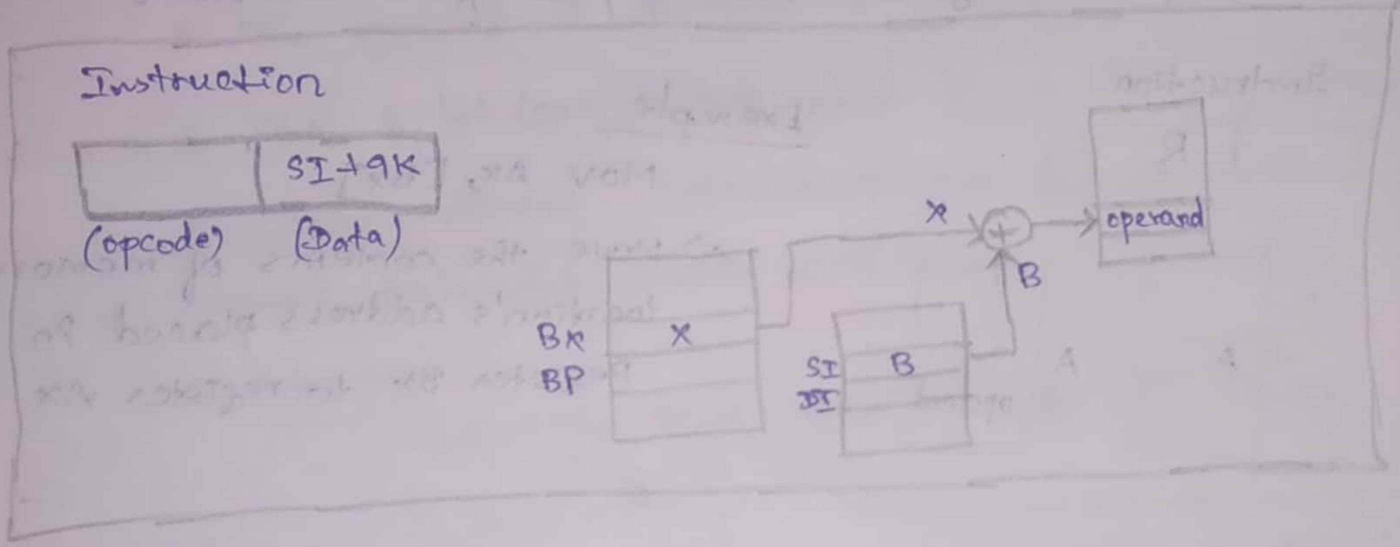
Example

MOV AX, [SI+05]

Base Addressing Mode

* Effective address is obtained by adding base register value to Index register SI (or) DI

Bx (or) BP



Example

ADD AX, [SI+Bx]

→ requires 83 clock cycles to compl

DATA TRANSFER INSTRUCTIONS

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 stac
POPA	(80186/80188 only) copy words from stack to all register.
XCHG	Exchange bytes or exchange words
XLAT	Translate a byte in AL using a table in memory

simple input & output 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 instructions

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 instructions

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

ARITHMETIC INSTRUCTIONS

Addition 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 word by 1
AAA	AASCII adjust after addition
DAA	Decimal (BCD) adjust after addition.

Subtraction instructions:

SUB	Subtract byte from byte (or) word from word
SBB	subtract byte and carry flag from byte (or) 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 (from 2's complement)
CMP	Compare two specified bytes (or) two specified words.
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 before division
CBW	Fill upper byte of word with copies of sign bit of lower byte
CWD	Fill upper word of double word with sign bit of lower word.

BIT MANIPULATION INSTRUCTIONS

Logical Instructions

NOT	Invert each bit of a 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 don't change operands

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

STRING INSTRUCTIONS :

REP	An instruction prefix. Repeat following 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/MOVSB/ MOVSW	Move byte (or) word from one string to another.
COMPS/COMPSB/ COMPSW	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

PROGRAM EXECUTION TRANSFER INSTRUCTIONS:

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:

JA / JNBE	Jump if above / Jump is 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 no carry (CF = 0)
JNC	Jump if no carry (CF = 0)

Scanned with CamScanner

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)

<u>Iteration Control Statement:</u>

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

<u>Interrupt Instructions:</u>

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

High-level language interface instructions:

ENTER	(80186/80188 only) Enter procedure
LEAVE	(80186/8088 only) Leave procedure
BOUND	(80186/80188 only) check if effective address within specified array bounds.

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
CLD	clear direction flag DF to 0
STI	Set interrupt enable flag to 1 (enable INTR input)
CLI	clear interrupt enable flag to 0

External Hardware synchronization instructions:

HLT	Halt until interrupt (or) rest
WAIT	wait until signal on the TEST pin low
ESC	Escape to external coprocessor
LOCK	Prevents another processor from taking the bus while the adjacent instruction executes

No operation instruction:

NOP	No action except fetch & decode