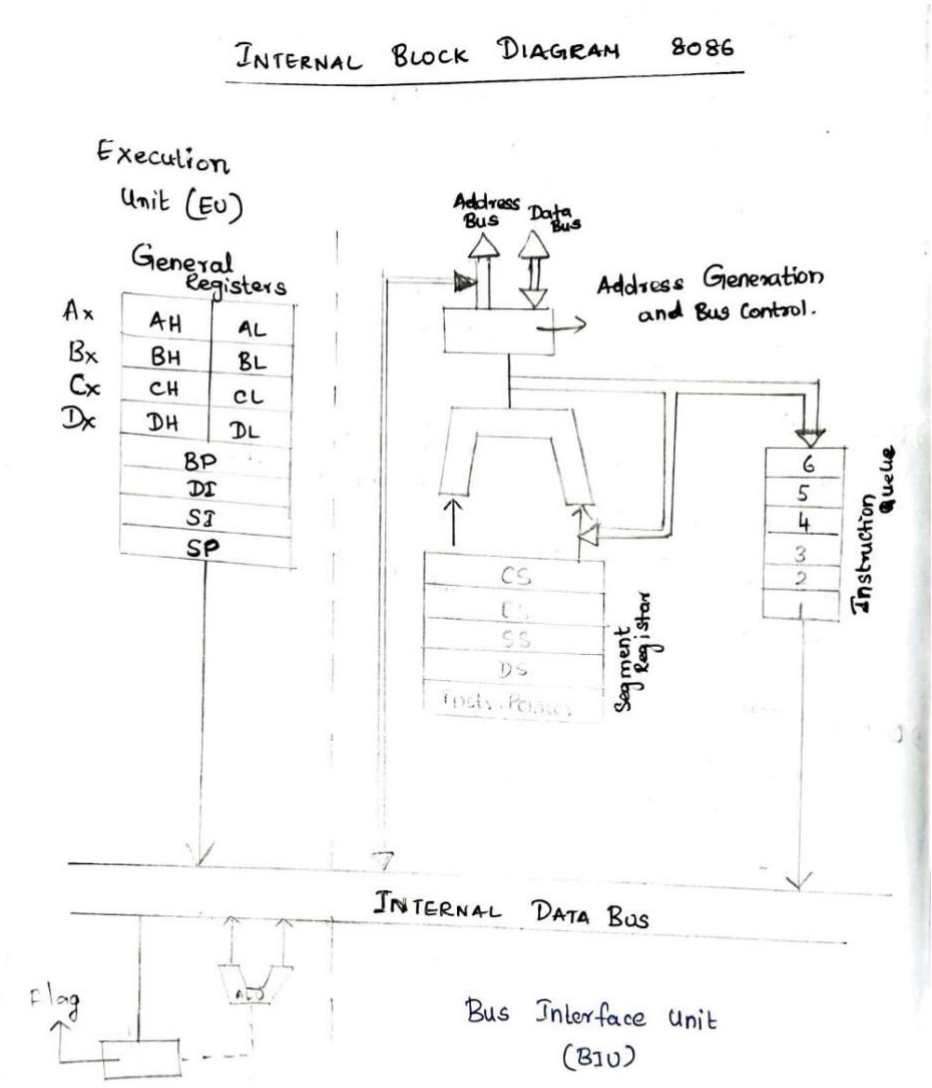
Assignment-I

19MID0020 Prashanth.S

8086 Internal block structure and it's description



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 instructions, the BIU proceeds to fetch new instruction.

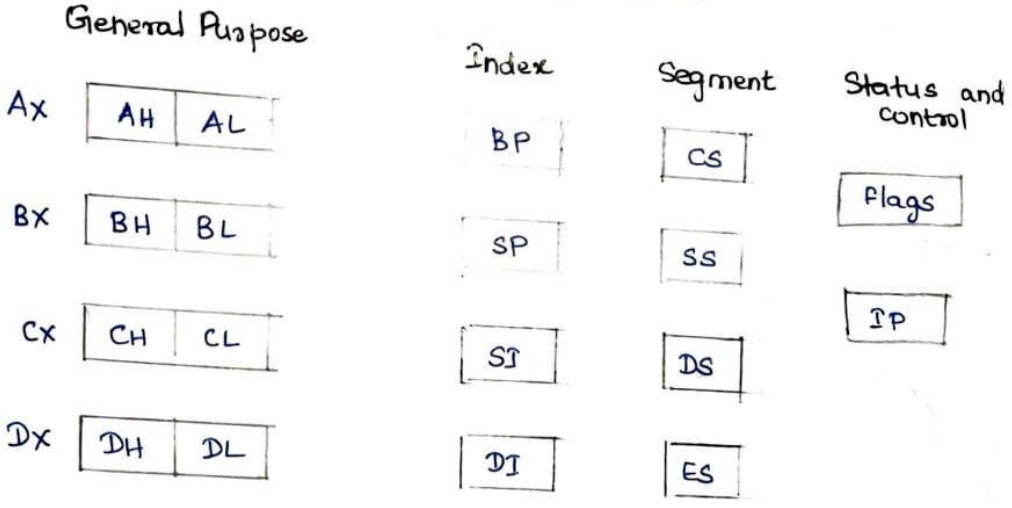
THREE CONDITIONS THAT WILL CAUSE THE 'EU' TO ENTER A 'WAIT MODE'

1. When an instruction requires access to a memory location not in queue.
2. Jump Instruction Executed.
3. Slow to execute.

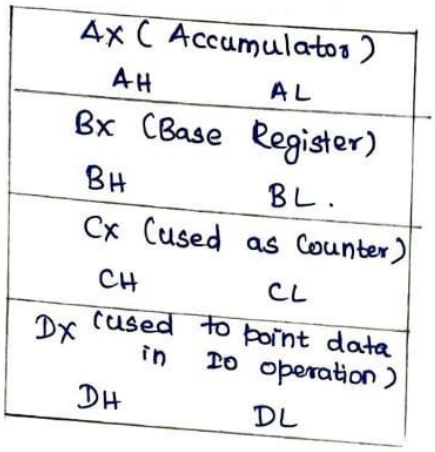
e.g

AAM → AASCJ Adjust Multiplication
→ requires 83 clock cycles to complete

8086 REGISTERS



General Purpose Registers :-



* 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:-

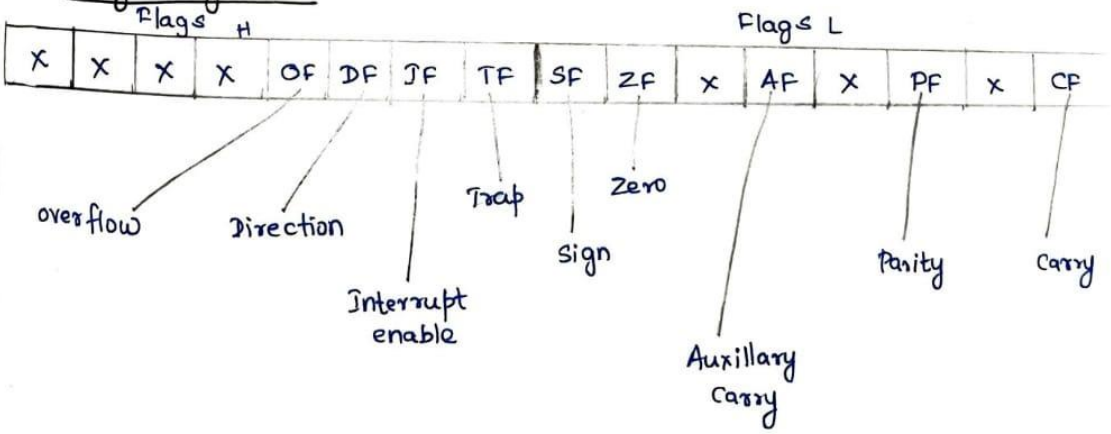
MOV AH [SI]

→ 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	Source Index.
DI	Destination Index.
IP	Instruction Pointer

Flag Register :-



6 status flags and 3 control flags

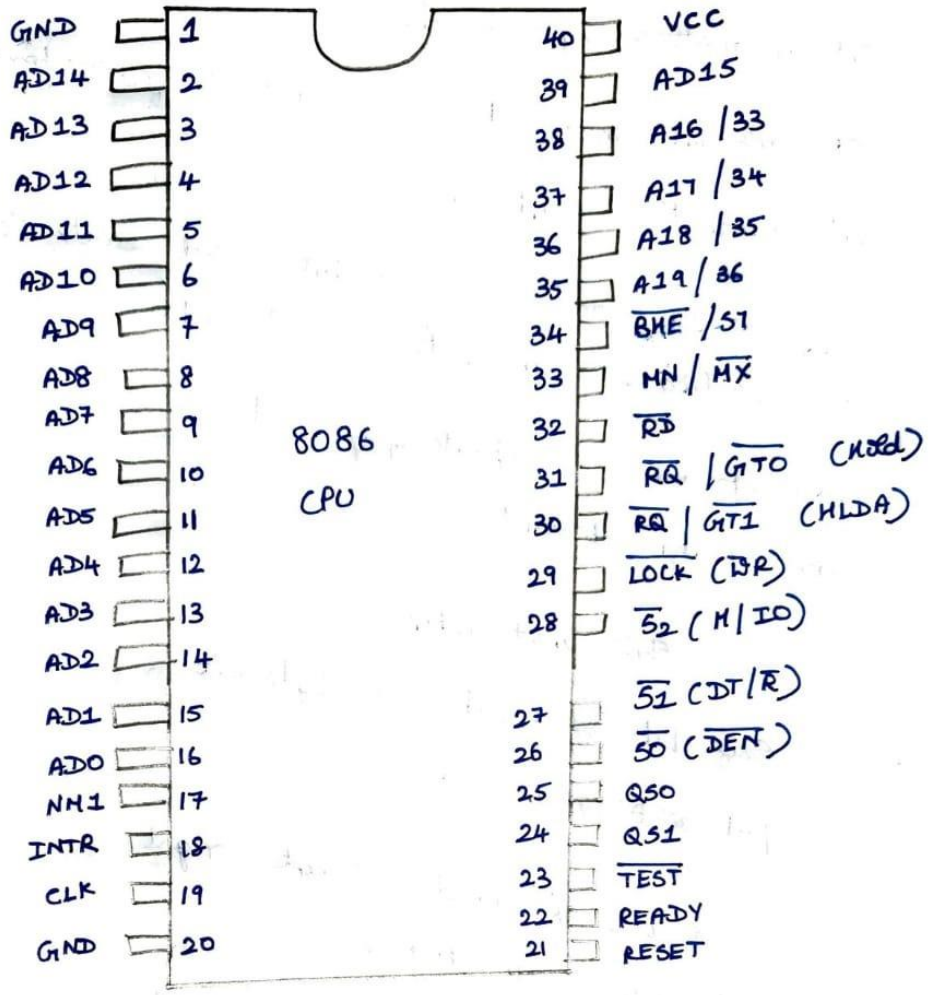
8086 Programmer's Model :-

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 Register
	DI		Destination Index Reg
	FLAGS		

8086 Pin configuration and it's signal pin description



*) Ground : PIN 1 and PIN 20 (GND)

*) Clock : PIN 19, Duty cycle : 33% (CLK)

*) power cycles : PIN 40, 5V ± 10% (VCC)

*) Reset :) PIN 21 (RESET)

•) Registers, segments, flags

•) CS : FFFFH

IP : 0000H

*) Address ... :) PIN 25 (ALE)

) When high, multiplexed address/data bus contains address information.

*) Address/Data Bus :) PIN-2 to PIN-16 and PIN-39 (AD0 - AD15)

) Contains address bits A_{15} to A_0 when ALE is 1

) Contains Data bits D_{15} to D_0 when ALE is 0.

*) Interrupt:) Non Maskable Interrupt (NMI) ⇒ PIN 17

) Interrupt Request (INTR) ⇒ PIN 18

) Interrupt Acknowledge (INTA) ⇒ PIN 24

*) HOLD Direct Memory Access

Hold (HOLD) - PIN 31

Hold acknowledge (HLDA) - PIN 30

*) Address/Status Bus:

) PIN 35 to 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.

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 | $S7'$

- PIN 34
- Enables most significant data bits D_{15} to D_8 during read (or) write operation.
- S_{17} always 1
- BHE#, A0

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 / MX)

Maximum Mode

Minimum Mode

Minimum Mode Pins ⇒ Pin 24 to 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 to PIN 28 ($\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$)

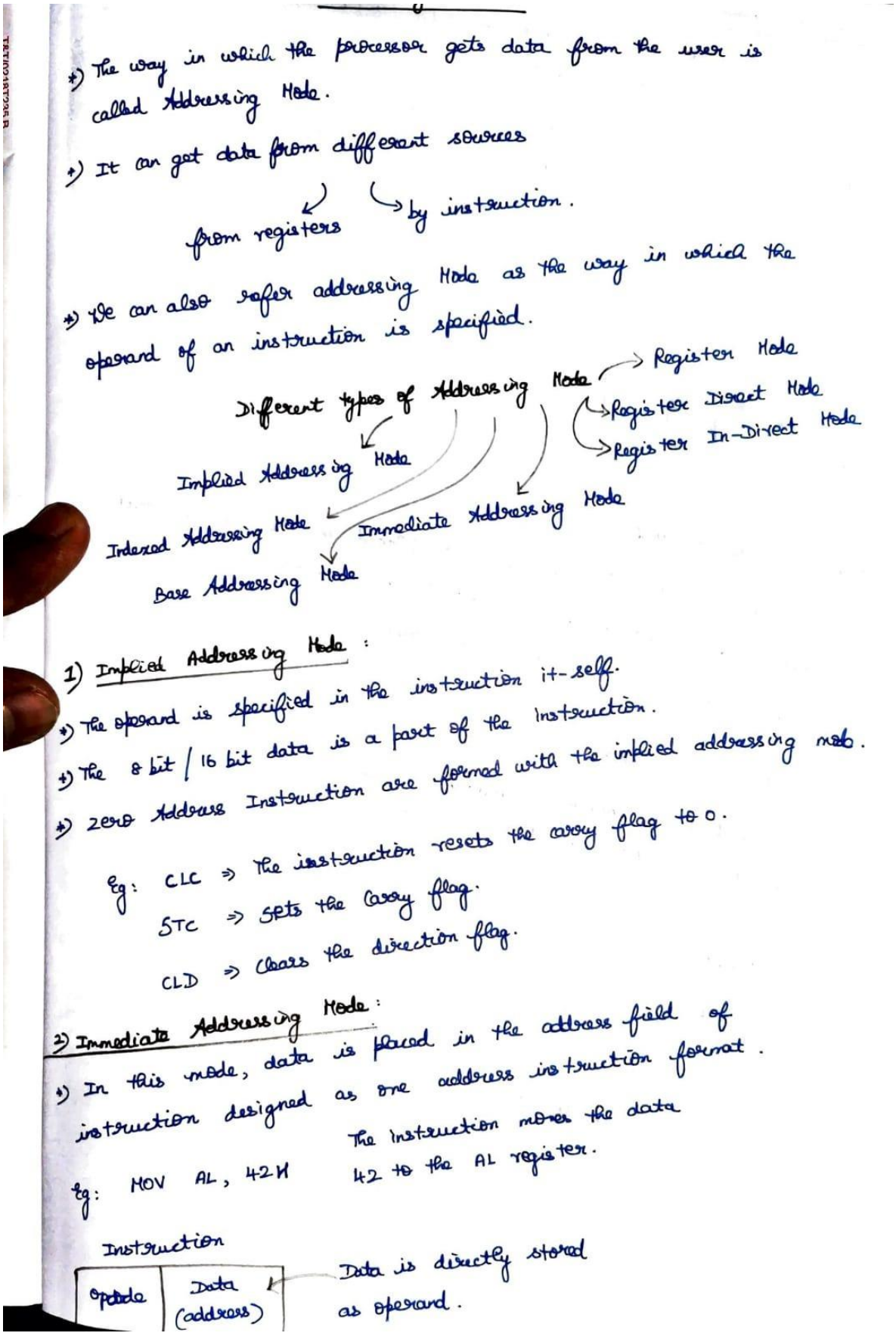
Inputs to 8288 to generate eliminated signals due to max mode.

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

*) Lock output:
  *) Used to lock peripherals of the system.
  *) Activated by using LOCK : Prefix on any instruction.
  *) LOCK output ($\overline{LOCK}$) => PIN 29
  *) DMA Request / GRANT => PIN 30 and PIN 31

*) Queue status:
  *) PIN-24 and PIN-25
  *) Used by numeric co-processor (8087)
  *) QS1 - QS0

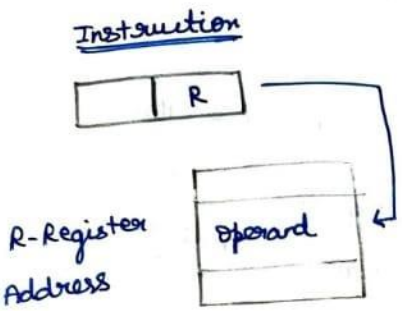
Addressing mode with examples



3) Register Mode

*) In this mode, (operand) data is placed in 8 bit / 16 bit register.
*) The register is specified in the instruction.

eg:
MOV AX, CX } This instruction moves the contents of CX register to AX register.

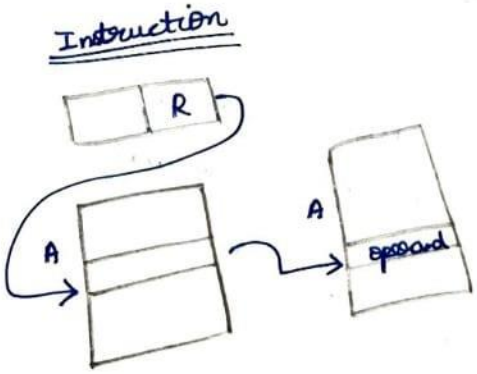


4) Register Indirect Mode

*) In this mode, data (operand) will be placed in the memory location.
*) The address of memory will be placed in register.
*) The register will be the part of instruction

eg: MOV AX, [BX]

Moves 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 this mode, 16 bit address of data is a part of instruction.

Eg: ADD AL, [0301]

Adds the contents of offset address to the contents of AL

MOV CL, [4321 H]

Moves the data from the location 4321 H in the segment into CL.

Physical address is calculated as DS * 10_H + 4321

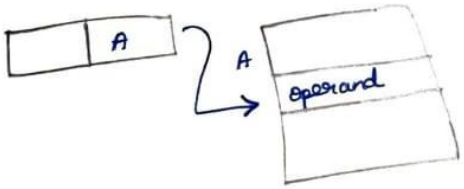
Assume DS = 5000 H

Physical Address ⇒ 50000 + 4321

CL ⇒ [54321 H]

```
  50000
+  4321
-------
  54321
```

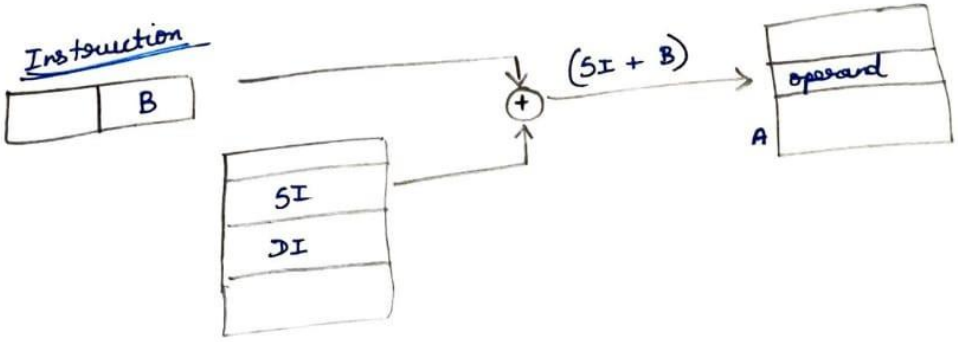
Instruction



b) Indexed Addressing Mode:

*) Operands address is the sum of the content in the index register SI (or) DI and 8 bit (or) 16 bit displacement.

Instruction

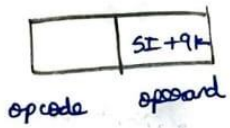


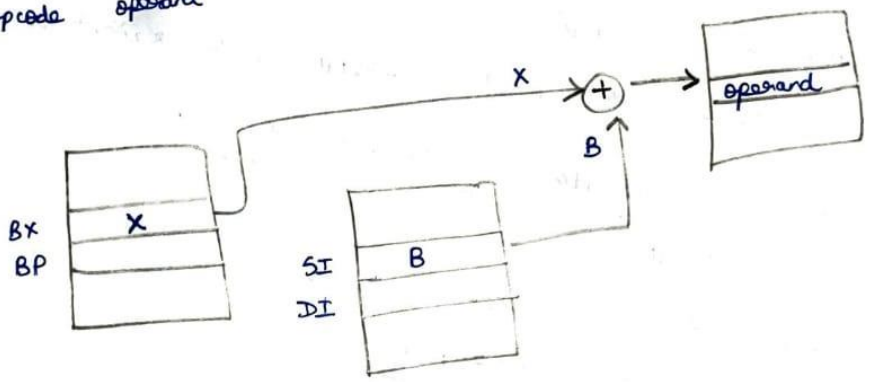
#) Base Addressing Mode

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

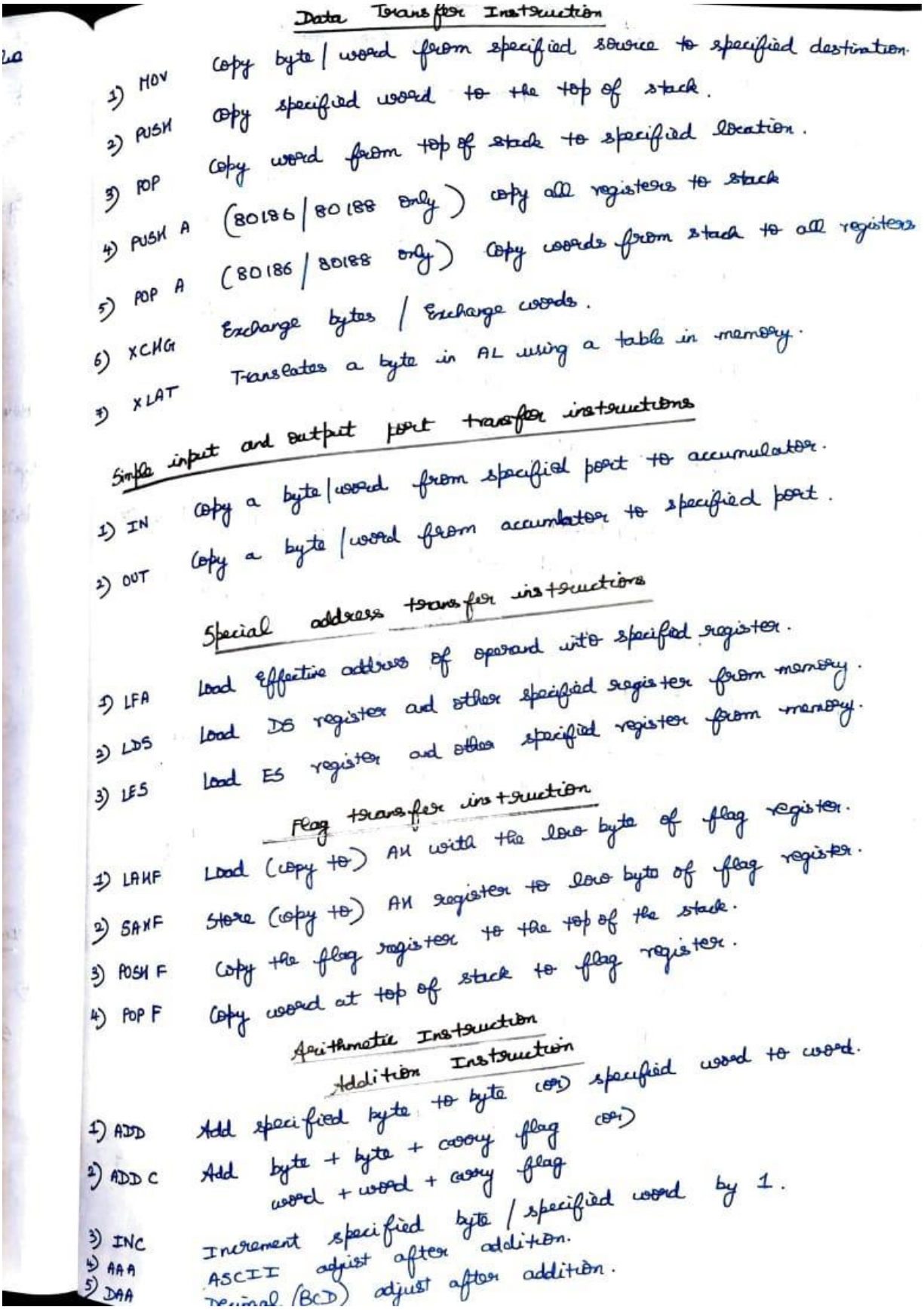
eg: ADD AX, [SI + BX]

Instruction





Instruction set summary



Data Transfer Instruction

1) MOV — Copy byte / word from specified source to specified destination.
2) PUSH — Copy specified word to the top of stack.
3) POP — Copy word from top of stack to specified location.
4) PUSH A — (80186 / 80188 only) copy all registers to stack
5) POP A — (80186 / 80188 only) Copy words from stack to all registers
6) XCHG — Exchange bytes / Exchange words.
7) XLAT — Translates a byte in AL using a table in memory.

Simple input and output port transfer instructions

1) IN — Copy a byte / word from specified port to accumulator.
2) OUT — Copy a byte / word from accumulator to specified port.

Special address transfer instructions

1) LEA — Load effective address of operand into specified register.
2) LDS — Load DS register and other specified register from memory.
3) LES — Load ES register and other specified register from memory.

Flag transfer instruction

1) LAHF — Load (copy to) AH with the low byte of flag register.
2) SAHF — Store (copy to) AH register to low byte of flag register.
3) PUSH F — Copy the flag register to the top of the stack.
4) POP F — Copy word at top of stack to flag register.

Arithmetic Instruction

Addition Instruction

1) ADD — Add specified byte to byte (or) specified word to word.
2) ADD C — Add byte + byte + carry flag (or) word + word + carry flag
3) INC — Increment specified byte / specified word by 1.
4) AAA — ASCII adjust after addition.
5) DAA — Decimal (BCD) adjust after addition.

1) SUB — Subtract byte from byte / word from word.

2) SBB — Subtract byte and carry flag from byte/word and carry flag from word.

3) DEC — Decrement specified byte / specified word by 1

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

5) CMP — Compare two specified bytes / two specified words.

6) AAS — ASCII adjust after subtraction.

7) DAS — Decimal (BCD) adjust after subtraction.

Multiplication Instruction

1) MUL — Multiply un-signed byte by byte / unsigned word by word

2) IMUL — Multiply signed byte by byte / signed word by word.

3) AAM — ASCII adjust after multiplication.

Division Instruction

1) DIV — Divide unsigned word by byte (or) unsigned double word by word.

2) IDIV — Divide signed word by byte (or) signed double word by word.

3) AAD — ASCII adjust before division.

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

5) CWD — Fill upper word of double word with sign bit of lower word.

Bit Manipulation Instructions

Logical Instructions

1) NOT — Invert each bit of a byte (or) word.

2) AND — AND each bit in a byte | word with the corresponding bit in another byte | word.

3) OR — OR each bit in a byte | word with the corresponding bit in another byte | word.

4) XOR — Exclusive OR each bit in a byte | word with the corresponding bit in another byte | word.

5) TEST — AND operands to update flags but don't change operands.

Shift instructions

1) SHL | SAL — Shift bits of word (or) byte left ; put zero(s) in LSB(s)

2) SHR — Shift bits of word (or) byte right ; put zero(s) in MSB(s)

3) SAR — Shift bits of word (or) byte right , copy old MSB into new MSB.

Rotate instructions

1) ROL — Rotate bits of byte (or) word left , MSB to LSB and to CF.

2) ROR — Rotate bits of byte (or) word right , LSB to MSB and to CF.

3) RCL — rotate bits of byte (or) word left , MSB to CF and CF to LSB.

4) RCR — Rotate bits of byte (or) word right , LSB to CF and CF to MSB.

String Instructions

1) REP — An instruction prefix. Repeat the following instruction until CX=0

2) REPE / REPZ — An instruction prefix. Repeat the instruction until CX=0 (or) zero flag ZF ≠ 1

3) REPNE / REPNZ — An instruction prefix. Repeat until CX=0 (or) ZF = 1

4) STDS / STOSB / STOSD — Stores the byte from AL (or) word from AX to string

Program Execution Transfer Instructions

Unconditional transfer instruction

- CALL — Call a procedure (sub-program) save return address on stack.

RET — Return from procedure to calling program.

JMP — Go to specified address to get next instruction

Conditional Transfer Instruction

JA / JNBE — Jump if above / Jump is not below / equal.

JAE / JNB — Jump if above (or) equal / Jump if not below.

JB / JNAE — Jump if below / Jump if not above (or) equal

JL / JNGE — Jump if less then / Jump if not greater (or) equal.

JLE / JNG — Jump if no carry (CF = 0)

INC — Jump if NO carry (CF = 0)

2-907

Iteration Control Statement

1) LOOP Loop through a sequence of instructions until CX=0

2) LOOPE/LOOPZ Loop through a sequence of instructions while ZF=1 & CX≠0

3) LOOPNE/LOOPNZ Loop through a sequence of instructions while ZF=0 & CX≠0.

4) JCXZ Jump to specified address if CX=0.

Interrupt Instructions

1) INT Interrupt program execution call service procedure.

2) INTO Interrupt program execution if OF=1

3) IRET Return from interrupt service procedure to main program

High-Level language interface instruction

1) ENTER (80186/80188 only) Enter procedure

2) LEAVE (80186/80188 only) Leave procedure

3) BOUND (80186/80188 only)
Check if the effective address within specified array bounds.

I-901

Process Control Instructions

Flag set/clear Instructions

1) STC — Set carry flag CF to 1
2) CLC — Clear carry flag CF to 0
3) CMC — Complement the state of the carry flag CF
4) STD — Set direction flag DF to 1
5) CLD — Clear direction flag DF to 0.
6) STI — Set interrupt enable flag to 1 (enable INTR input)
7) CLI — Clear interrupt enable flag to 0.

External Hardware synchronization instructions.

1) HLT — Halt until interrupt / rest.
2) WAIT — Wait until signal on the TEST pin low.
3) ESC — Escape to External Co-processor.
4) LOCK — Prevents another processor from taking the bus while the adjacent instruction executes

No operation instruction

1) NOP — No action except fetch and decode.