ClassRoom
Page No.
Date / / /

Sign/ Remarks

Microprocessor 8086

→ The three buses in a processor are:
ADDRESS Bus - selects the memory or the i/o (input / output)
DATA Bus - Transfer data b/w memory or i/o and the microprocessor
CONTROL Bus - carries synchronization signals b/w both.

→ FETCH - DECODE - EXECUTE. A microprocessor conducts this cycle for every instruction until it comes across the stop (HLT / RST 75) instruction

→ 8086 - first 16-bit microprocessor
- compatible with present-day microprocessors

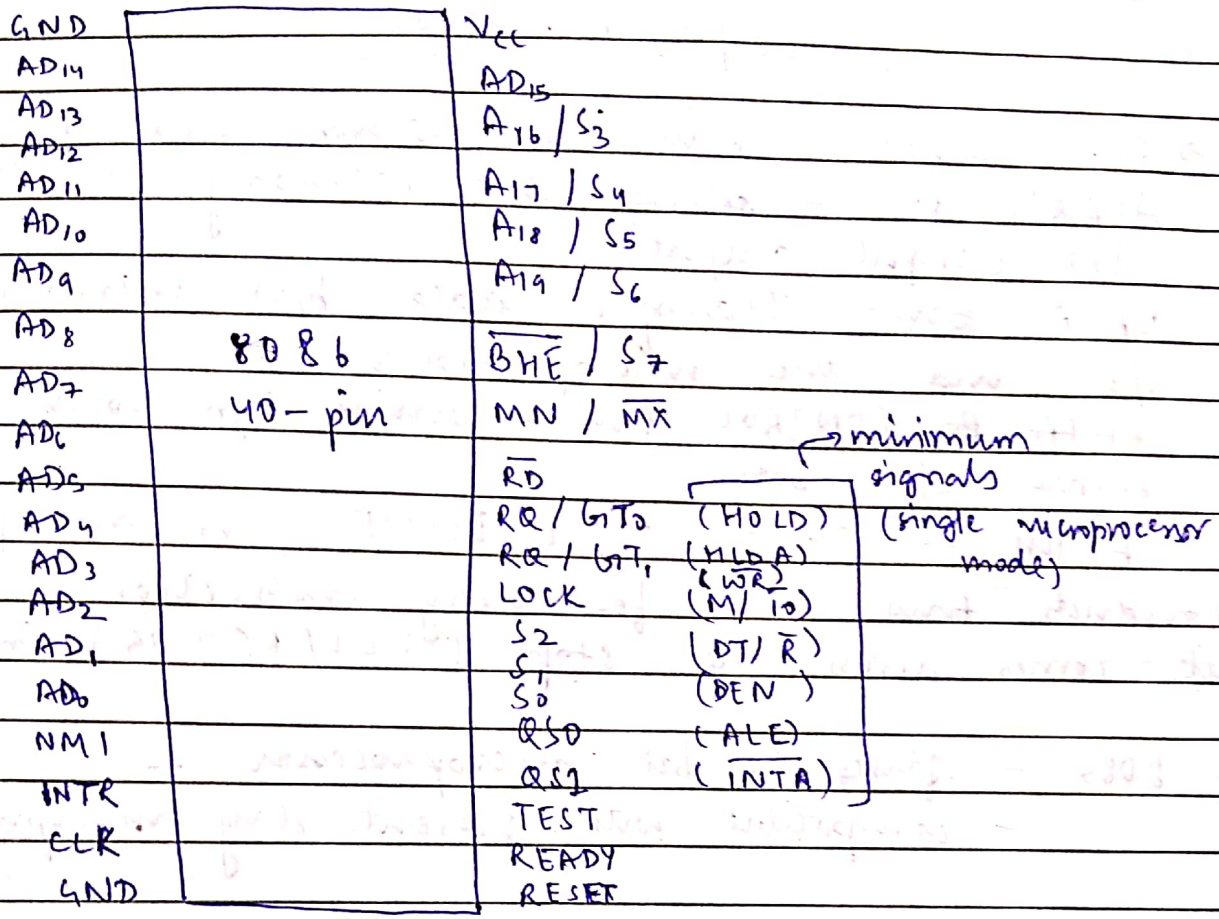
→ For 8086
Address Bus - unidirectional
- 20-bit
Data Bus - bidirectional
- 16-bit

→ 8086 works on an internal clock frequency of 5 MHz
Other variants 8086-2 ⇒ 8 MHz
8086-4 → 4 MHz

→ 16-bit microprocessor means storage capacity of 16 bit; data line is 16-bit.

→ 8086 has a capacity to address upto 1 MB of memory, ie, 2^{20} → width of address bus

→ 8086 is a 64-pin IC.



AD_0 - AD_{15} → multiplexed address - data lines (bus)

A_{16} - A_{19} → address lines

S - S_7 → status signals (perform synchronization signals)

NMI → non-maskable interrupt. (highest priority operation)

INTR → interrupt pin (interfacing applications)

RESET → reset the processor

ALE → Address Latch Enable signal

If active, then AD_0 - AD_5 is address

If not active, multiplexed bus is data

Processor completes the current process and then checks this operation. NMI stops the current operation and executes the one on priority basis.

ClassRoom
Page No.
Date / / /

CLK → 8086 has an internal clock generator 8284 which divides the external frequency on CLK by 3.

READY → synchronise the slower peripherals with microprocessor.

TEST → synchronise the operations of multiple processors.

LOCK → prevent the other processor from gaining control

DEN → Data Enable

QS1 → Queue Status 1] 6-bit instruction queue in 8086

QS0 → Queue status 2]

RQ / GT → Request - grant to enable other processor to gain control

BHE → Bank High Enable signal (address location memory odd → high even → low)

MN / $\overline{MX}$ → min or max. mode (multiprocessor mode)

$\overline{RD}$ → Read

$\overline{INTA}$ → interrupt acknowledge signal

DT / $\overline{R}$ → Data transferred or received

M / io → memory & i/o

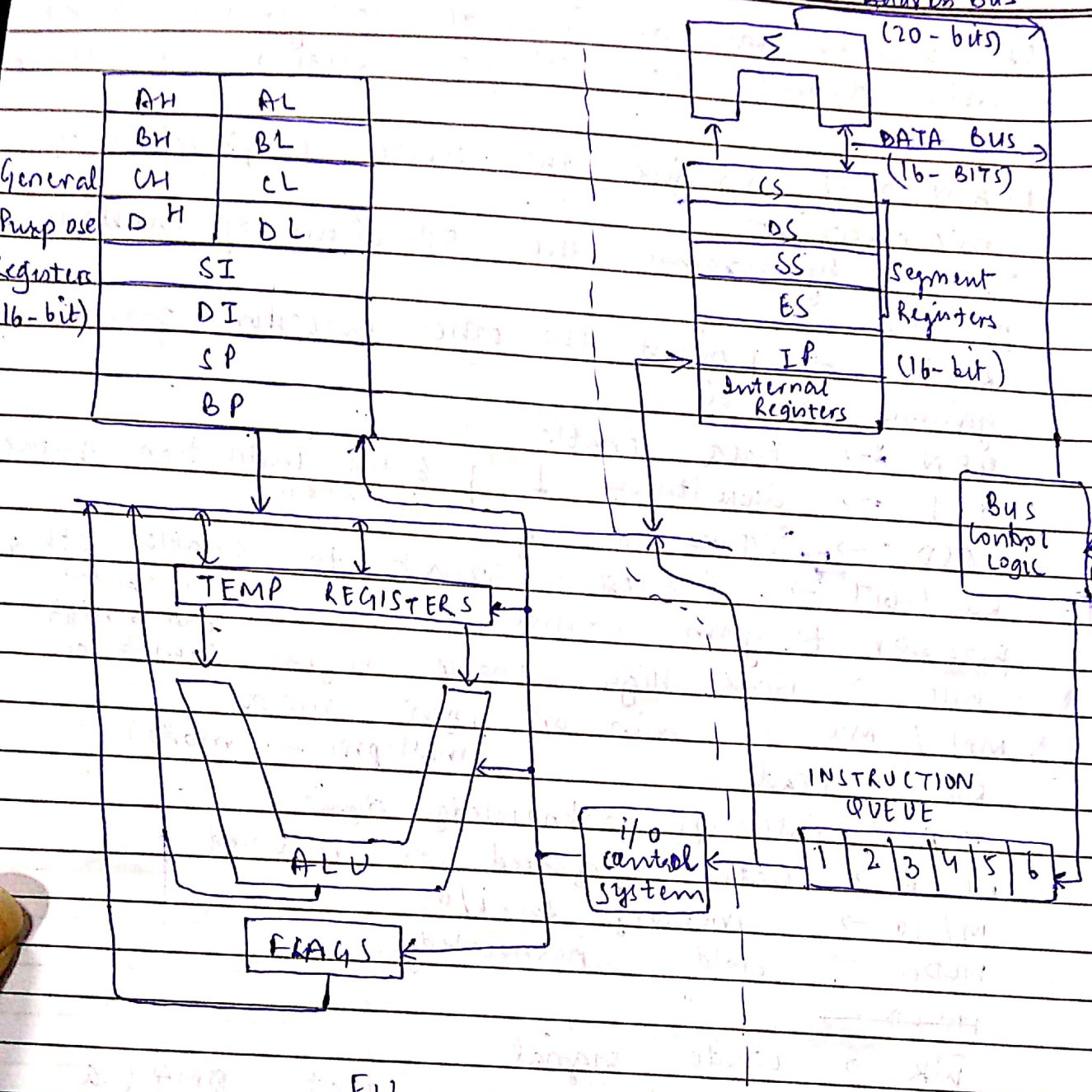
HLDA → Hold Acknowledge

~~HOLD →~~

$\overline{WR}$ → write signal

HOLD → tells the processor that DMA (a peripheral) is requesting access and hence it bypasses the microprocessor.

6-bit instruction queue means it can fetch 6-~~bit~~ instructions ahead of time (when the decode-execute is going on), called as pipeline. QS tells its status.



EU
(Execution Unit)

BIU
(Bus Interface Unit)
↓
fetch

AX → 16-bit general purpose register which can also be used as 2 8-bit register
AH - high order register of AX
AL - low order

ClassRoom
Page No.
Date / / /

SP → Stack pointer BP → Base pointer
FLAGS → 16-bit register
CS → code segment – stores the code (instructions)
DS → Data " – stores the data in the instructions
SS → Stack "
ES → Extra " – extension of data segment
IP → instruction pointer
Σ → adder, which generates a 16-bit general address

⇒ Real Mode Addressing:
Allows microprocessor to address the first 1 MB of the memory capacity.
Eg: 8086 – 1 MB (Memory capacity)
So, protected mode addressing is not possible

Protected Mode Addressing:
Rest of the memory

⇒ Memory of 8086 is divided into 4 segments of 64 kB each. The rest of the memory is reserved for other applications.
These 4 segments are CS, DS, SS, ES.

⇒ 1000 : 2000 ← 16-bit address
segment address — offset address ⇒ location within a specific segment

Let's suppose the starting address for CS is 1000.
CS – 1000 : 0000 ⇒ first address of CS
DS – 2000 : 0000

have an address (starting point)

Pointer to segment register — Pointers to memory location

ClassRoom
Page No.
Date / / /

SS - SP, BP ; SP & BP act as offset address for SS.

DS - SI, DI → Decimal Index Register

~~CS - SI~~

SI → Source index register

16-bit register

↳ provides the address from where the next instruction needs to be placed.

CS - IP (instruction pointer)

ES - DI

eg:

1000 (CS)	: 0000	MOV	AX,	1234
	: 0003	MOV	BX,	1111
	: 0006	ADD	AX,	BX

↳ Instruction pointer tells the address for next instruction.

⇒ General Purpose Registers:

AX (Accumulator) - AH AL

↳ default register ~~- BH BL~~

BX (Base register) - BH BL

CX (Counter register) - CH CL

DX (Data Register) - DH DL

↳ ~~general~~.

If the result exceeds 16-bit, then the overflow data is stored in DX by default.

DX (overflow) AX (16 bit)

⇒ The final 20-bit Physical / Effective address is obtained by shifting segment address by one place to the left and adding the offset address.

Eg: SS - 1234H, BP - 1000(H) → hexadecimal no.

20-bit : 12340

+ 1000

13340

ClassRoom
Page No.
Date / / /

⇒ 1 MB → 2^{20} 1 KB → 2^{10}

64 KB → $2^{10} \times 2^6 = 2^{16}$

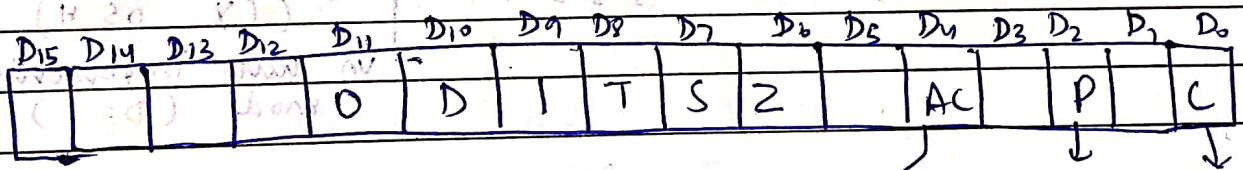
So, address range of each segment register is 2^4 → 0000 to FFFF (hexadecimal)

1 MB → 2^{20} 00000 to FFFFF

⇒ If we enter 8-bit data in AX, it will store it in AL and consider the junk value of AH

Eg: MOV AX, 00 H
AX = 7300 73 : junk value of AH
MOV AX, 0000 H
AX = 0000
MOV AH, 00 H
MOV AL, 00 H
] specific values entered into AH, AL.

⇒ 16-bit FLAG Register (8086)



1000 + 0111 = 1010 (carry)
1011

Auxiliary carry (in case of nibbles 4Bit data)

Parity Flag (no. of 1's) → even/odd

Carry

2: zero flag - when any operation results in zero, it gets high

S: ~~sign~~ flag - polarity / sine of your no.

MSB is high ⇐ -ve no. → sine flag sets

MSB is low ⇐ +ve no. → resets

(2's comp. form)

T: trap flag - single stepping / debugging the applications

Result of each instruction is ~~the~~ checked.

ClassRoom
Page No.
Date / / /

I: Interrupt flag - Request on INTR pin will be serviced only when interrupt flag is high.

D: Direction flag - used in string operations
- by default low

O: Overflow flag - overflow of data from registers storing it in DX

↳ Eg:
```
MOV CX, 05H
MOV SI, 1111
MOV DI, 2222
MOV DI, SI
```

If D is low, work in auto incrementing mode
D is high, auto decrementing mode.

```
1000 : 1111 ——— 2222
       1112 ——— 2223
       1113 ——— 2224
       .....
```
Values are stored for 5 values ((X, 05H) in auto incrementing mode (D=0)

→ Memory to memory operations are not allowed in 8086
Eg: ADD DI, [SI]

```
MOV AX, 1234
MOV BX, 1111
ADD AX, BX
```
AX ↓ destination register; BX ↳ source register

INT A5 ↓ stops the execution!
(A5 is the address where the stop command is stored)

```
MOV SI, 1200
MOV DI, 1300
MOV AX, [SI]
ADD [DI], AX
INT A5
```
↳ Data register stores 8-bit data

ClassRoom
Page No.
Date / / /

MOV AX, [SI]
16-bit ↳ 8-bit
1200 location → data stored in AL
1201 location → data stored in AH
MOV AL, [SI] or MOV AH, [SI]
1200 location → data stored in AL | AH.

* INSTRUCTION SET

– Data Transfer Instructions

① MOV → copies content from source to destination

Eg: MOV AX, CX
MOV [SI], CX ; 1200 = CL, 1201 = CH
MOV BX, [SI] → brackets means the data of the address

② XCHG → exchanges the instructions :
Eg: XCHG AX, DX
XCHG [1200], BX

③ LEA → Load Effective Address
Eg: LEA SI, [0200] → offset address
DS = 1200
Physical Address ← PA = 12200 (SAX10 + offset) ← segment address.
SI ⇐ 12200 = AB
12201 = CD
So, SI = CDAB
MOV AX, [SI]

⇒ this data changes the value of SI pointer (address)

④ LDS → Load Data segment
Eg: LDS SI, [0200]
12200 ⇒ SI
12201
12202 ⇒ DS
12203

(5) LES - Load Extra segment

Eg: LES SI, [0200]

12200 → SI
12201

12202 → ES
12203

(6) XLAT - Translate

- only instruction that adds 8-bit no. to 16 bit.

Operation

PA = DS + BX + AL
(16-bit) (8-bit)

[PA] → AL.

Eg: PA = 12000 DS x 10
0200 BX
+ AB AL
PA ⇒ Address location

Data stored at address [PA] is stored in AL.
location of PA

- Addition Instructions

(1) ADD AX, BX → destination register (added value is stored in AX)
ADD AX, [1200]

(2) ADC BX, CX
BX = BX + CX + CF (carry flag)
↓
Addition with carry.

(3) INC AX
Increments the value by 1

(4) DAA - Decimal Adjust Addition
- BCD addition
- performs operation on accumulator, so no parameter is passed.

Eg: ADD AX, BX
DAA.

ClassRoom
Page No.
Date / / /

- Subtraction

① SUB BX, CX

② SBB BX, CX
BX = BX − CX − CF

③ DEC AX
Decrements the value by 1

④ DAS - Decimal Adjust Subtraction
- BCD Subtraction -

⑤ NEG AX → Negate AX
→ 2's complement of AX is stored in AX after negating AX.

- Multiplication & Division

① MUL N
AL . N8 ⟶ AX
AX . N16 ⟶ (DX) (AX)
Higher order, Lower order

② DIV N
Q (AX / N8) → AL
R (AX / N8) → AH
Q ((DX)(AX) / N16) → AX
R ((DX)(AX) / N16) → DX

③ IMUL N
④ IDIV N
} Integer operation

⑤ AAM
AAD
} Adjust AX for multiplication and division. (for BCD)

ClassRoom
Page No.
Date / / /

Q. Determine the effect of the instruction DIV CH assuming the data prior to execution as CX = 0300H, AX = 0091 H

Ans: AL → 30
AH → 01

- Logical Instructions

① AND AX, BX
OR BX, 05 (05: 0000 0101)
XOR CX, AX
NOT AX

Result saved in destination register.
In logical instructions, no flags are affected.

② TEST CL, 05
Performs AND operation but the result is not saved, and the flags are affected.

③ CBW - Convert byte to word
CWD - Convert word to double word.
- Work on accumulator as no argument is passed.
- Byte → 8 bit Word → 16 bit
Double word → 32 bit
- These are used on signed operations
CBW : MSB of AL is extended into AH.

Eg: AL = 7F = 0111 1111
AX = 007F
AL = 8F = 1000 1111
AX = FF8F

ClassRoom
Page No.
Date / / /

CWD – MSB of AX is extended into DX.
Eg: AX = 1234
⇒ DX = 0000 — 0 000 1234

Q Determine the effect of CBW & CWD assuming the following data prior to execution: CX = 0300
AX = 0091

Ans CBW
AL = 91 H = 1 001 0001
AX = FF 91 H

CWD
AX = 0091 H
DX = 0000 H

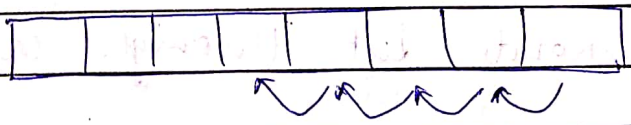
– Shift and Rotate Instructions

Shift
- Arithmetic – used with signed nos (MSB)
- Logical – used in unsigned nos.

① SHL – Shift Logical Left
SHR – Shift Logical Right

② SAL – Shift Arithmetic Left
SAR – Shift Arithmetic Right

SHL (or SHR)



Bits are shifted one place to the left and the vacated positions ~~at~~ at the right are filled with 0.
SAL also performs the same task.

SAR → one bit shifted to the right and vacated spots to the left are filled with the signed bits.
Eg: 7f → 0's are filled
8f → 1's are filled

ClassRoom
Page No.
Date

→ Left shift performs multiplication by 2 and right shift performs division by 2.
→ The outgoing bit is stored in the carry flag.

Q. Assume that CL contains 02H and AX contains 091A H. Determine the new contents of AX and the carry flag after the instruction SAR AX, CL is executed.

Ans: AX = 0000 1001 0001 1010

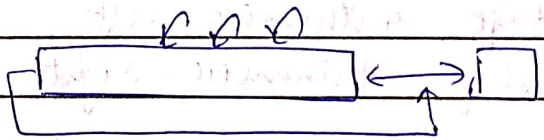
After execution
AX = 0000 0010 0100 0110
= 0246 H

Carry flag : High (1)

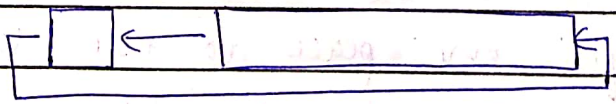
(3) ROR – Rotate right

The outgoing bit is filled into carry & MSB.

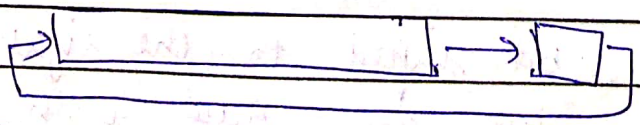
ROL – Rotate left



RCL – Rotate left through carry



RCR – Rotate Right through carry



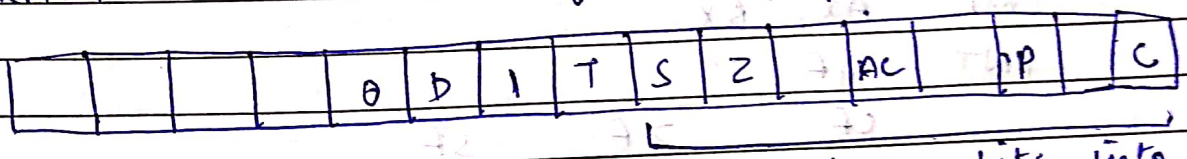
ClassRoom
Page No.
Date / / /

Q Write a single instruction that will load AX from the address 4136 and DS from the address 4138.

Ans: LDS AX, [4136]

Flag Control Instructions

① LAHF – Load AH from flag



Load lower bits into AH (8-bit)

So, these flags can be checked using this instruction.

② SAHF – store AH into flags
– stores value of bits of AH into corresponding flag.

AH = 1 0 1 0 0 1 1 0
S Z Ac P C → values stored

③ CLC – clear carry flag
STC – set carry flag
CMC – complement carry
CLI – clear interrupt
STI – set interrupt

– O, D, I, & T can be checked only using jump instruction.

- CMP AX, BX (AX - BX)
-compare AX and BX ; tells which is greater / equal
- result will not be stored, only flags are affected.

```
MOV   AX , 1234
MOV   BX , 3456
CMP   AX , BX
INT   Af
```

	Cf	Zf	Sf
AX = BX	0	1	0
AX > BX	0	0	0
AX < BX	1	0	1

- Jump Instructions

Eg: Conditional
JNZ 020A ← I.P.
- jumps acc. to the conditions specified in previous instruction

Unconditional
JMP 020A
- jumps no matter what.

31 conditional jump instructions:

unsigned nos.

- JA - jump on above (takes place when CF=0, ZF=0)
- JAE - jump on above or equal (CF=0 ZF=1)
- JB - jump on below (Cf = 1)
- JBE - (CF=1 or ZF=1)
- JC - jump on carry (Cf = 1)
- JCXZ - jump when CX=0
  ↓
  counter register

Page No.
Date

takes into account signed nos.

- JE - equal, checks ZF
- JG - greater
- JGE
- JL - less
- JLE

JNA - jump on not above
JNAE - not above nor equal
JNB -
JNBE
JNC - jump on no carry (CF = 0)
JNE - not equal
JNG - not greater
JNGE
JNL
JNLE
JNO - jump on not overflow (OF = 0)
JNP - jump on not parity (PF = 0)
JNS - no sign (SF = 0)
JNZ - no zero (ZF = 0)
JO - jump on overflow
JP - parity
JPE - jump on ~~odd~~ even parity
JPO - odd parity
JS - sign flag
JZ - zero flag (ZF = 1)

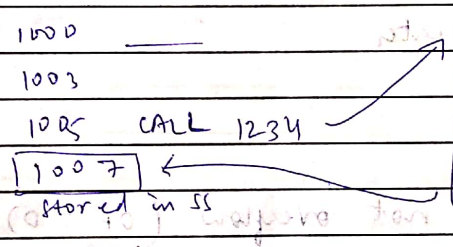
→
```
MOV SI, 1200
MOV DI, 1300
MOV CX, 0A
* MOV AH, [SI]
MOV [DI], AH
INC SI
INC DI
DEC CX
JNZ *
INT A5
```
(Transfer of values)

Date / / /

- Subroutine Handling Instructions

(1) CALL & RET - IP is altered
- The address of instruction next to CALL is stored in STACK segment, so it can be returned to after executing the subroutine

CALL 1234 → offset address
CALL BX → added with segment address to give
CALL [BX] → [CS*10 + BX] address location



(2) PUSH AX - stores contents of AX into stack
POP AX - retrieving contents from stack to AX.

```
PUSH  AX
PUSH  BX
 :
POP   BX
POP   AX
RET
```

SP - stack pointer
- offset address in stack segment
- always points at the top of SS.

SP - 2
PUSH AX
AH → SP-1
AL → SP-2

POP AX
AL ← SP+2 (SP)
AH ← SP-1 (SP+1)
SP + 2 (SP+2)

IP → increases
SP → decreases

1000 : 1000
1001
1002
2000 : 100A
1009
1008

- Loop Handling Instructions

LOOP ≈ DEC CX → by default decrements CX.
JNZ *

LOOP *
LOOPE / LOOPZ * ; checks CX ≠ 0 & ZF = 1
LOOPNE / LOOPNZ * ; ZF = 0 & CX ≠ 0

- String Handling Instructions - operate with SI & DI.

SB : string byte SW : string word

① MOV SB / MOV SW → SI to DI
② CMP SB / CMPSW - Compares SB ; compares two elements in 2 strings (SI & DI), i.e, subtracts destination operand from source operand and result of subtraction is not stored.
③ SCA SB / SCASW - Scan SB / SW - similar to compare instruction, but it compares the byte/word of destination string to the contents of AL / AX. (ES & DI)
④ LODSB / LODSW - Load SB/SW ; Loads the byte/word from string in source index into AL / AX. (DS & SI)
SI → incremented by 1 for LODSB (AL)
→ incremented by 2 for LODSW (AX)
⑤ STOSB / STOSW - stores the content of AL / AX into the memory location specified by DI. (ES & DI)
DI → +1 for STOSB (AL)
→ +2 for STOSW (AX)

↓
segments considered with SI & DI

ClassRoom
Page No.
Date / / /

Example :

```
MOV  SI, 1300   ⟶ offset addr. of Data segment
MOV  DI, 1400   ⟶ offset of extra segment
MOV  CX, 0BH
CLD             ; clear direction flag
NEXT : MOVSB    [DS+1300] moves into [ES+1400]
       LOOP NEXT
       INT  A5
```

[DS+1300] ⟶ [ES+140A]
increment locations automatically

- Repeat String Handling Instructions

Prefix	Used with	Function
REP	MOVS STOS	Repeat while not end of string CX≠0
REPE/REPZ	CMPS SCAS	Repeat while not end of string & strings are equal CX≠0 ; ZF=1
REPNE/REPNZ	CMPS SCAS	Repeat while not end of string & strings are not equal CX≠0 ; ZF=0

Example :

```
MOV  AL, 0F H
MOV  DI, A000 H
MOV  CX, 0A H
CLD
REP STOS B
INT  A5
```

A000 - 0f
: - 0f
: - 0f
A009 - 0f

ClassRoom
Page No.
Date / / /

Q1 What would be the effect of the following instruction sequence on AX.

PUSH	AX		ABCD
ADD	AX, 4		ABCD+4
POP	BX		ABCD
MOV	CX, AX		ABCD+4
PUSH	BX		ABCD
POP	AX		ABCD

Q2 Which flags are tested in instruction JAE 1000.

Q3 Develop a sequence of instructions that set the rightmost 4 bits of AX, clears the leftmost 3 bits of AX and invert bit 7, 8, 9 of Ax.

Q4 Assuming DS = 1200H, BX = 0100H, SI = 0250H. Determine the address accessed by MOV [100H], DL assuming real mode operation.

A1 AX is popped as it is.

A2 CF & ZF

A4 12100 100H; offset address

A3