

## Contents:

- |            |           |  |
|------------|-----------|--|
| Intel      | Chapter 1 |  |
| (Barry B.) | Chapter 2 |  |
| Mirin      | Chapter 3 |  |
| ASOSOIS    | Chapter 4 |  |
|            | Chapter 5 |  |
|            | Chapter 6 |  |

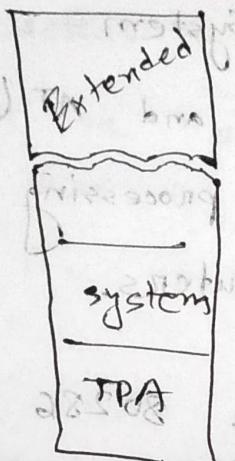
Intel Barry B. Bruey-ch 1

2102024 - sharza fat

## Chapter - 1

- 1) Charles Babbage  
2) Joseph Jacquard, a Frenchman who used it as input to a weaving machine.  
3) Hollerith  
4) Monroe  
5) Military applications  
6) ENIAC, eniakitent to 2000M = 291M (PS)  
7) Intel 8008, te mibiteni xelma = 3219 (PS)  
8) Countess of Lovelace worked with Charles Babbage in the development of software for his analytic engine.  
9) Grace Hopper  
10) Von Neuman machine is a type of system where instructions are also stored in memory.  
11) 8008  
12) 100 million

- 13) 8086  
14) 16M byte  
15) 4G  
16) 1993  
17) 1995  
18) Late 2000  
19) Pentium4  
20) MIPS = Millions of Instructions per second  
21) CISC = Complex instruction set computer  
22) 1 ratio between clocked to executed  
23)  $10^3$  bytes to transplant soft in standard  
chips without  
24)  $2^{10}$   
25)  $2^{20}$   
26)  $2^{30}$   
27) Depends on the per page size  
28) TPA and system area  
29) 2008  
30) million 001



bitmask = 209 (88)  
 (extended) bitshift = 111 (88)  
 out = 270 (extended)  
 1M (weak memory) 2208

instant 2G  
 29) 2G  
 30) 640K

31) 2G

register and JV are used here A23V (88)

32) 2M

register and JV are used here A23V (88)

33) 1T

34) 80386

register and JV are used here A23V (88)

35) extended

memory register and JV are used here A23V (88)

36) The system BIOS is a collection of

programs stored in either a read-only

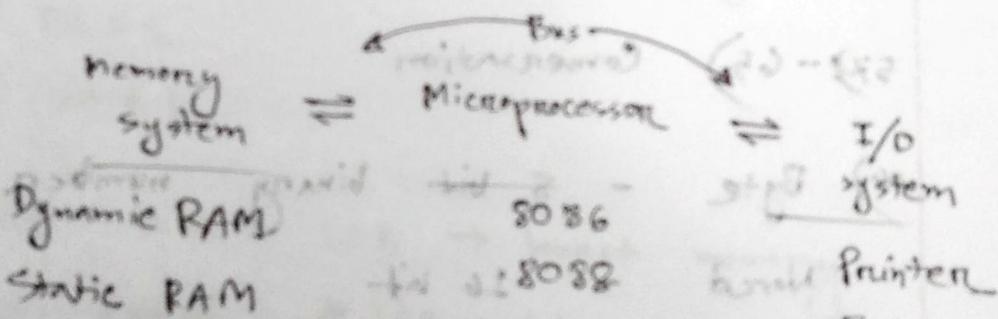
memory (ROM) or flash memory that operates

many of the I/O devices connected

to our computer system.

- 37) DOS = Disk Operating System
- 38) XT (Extended technology) and AT (Advanced Technology) are two processing power measurement of computers.
- 8086 used XT whereas 80286 introduced AT. AT was significantly faster, could handle twice as much data as XT.
- 39) VESA local bus or VL bus interfaces disk and video to the microprocessor.
- 40) 16
- 41) Universal Serial Bus (USB) is a serial data path with twisted pair of cables which is intended to connect peripheral devices such as keyboard, mouse, etc.
- 42) AGP or Advanced Graphics port is a type of bus that transfers data between video card and microprocessor.

- 43) XMS refers to extended memory system.
- 44) SATA or serial ATA interface is a bus that is used for hard disk drives and the PCI Express bus for video card.
- 45) TPA
- 46) G9 K
- 47) [same as 36] Pentium 400 - half  
[same as 36] Pentium 400 - full
- 48)



- 49) Doing the computations...
- 50) Address, data, control
- 51) Address of zeros not 256 at (6)
- 52) MRD<sub>0</sub> going to A<sub>0</sub> no result
- 53) IO Read control
- 54) Memory read

- 55) DB - Define Bytes or define MAX (8)  
DD - Define Quadword  
DW - Define word  
DD - Define of Doubleword
- 56) char - 8 bit character  
short - 16 bit integer  
int - 32 bit integer  
float - 32 bit floating point  
double - 64 bit floating point

- 57) - 65) Conversation
- Byte - 8 bit binary number  
Word - 16 bit  
Doubleword - 32 bit
- 67) conversation
- 68) The ASCII for enter key is carriage return or CR. It generally inserts a new line to the document.

69) Unicode : Unicode is a system used by windows based applications to store alphanumeric data.

70) LINE

DB

"what

time it is?"

71-72) Conversation

73) DATA DB - 34

74) 

- (a)  $\overrightarrow{3412}$  High to low memory
- (b) 22A1
- (c) 00B1

75) Little endian = LSB  $\rightarrow$  lowest memory  
Big endian = MSB  $\rightarrow$  highest memory

76) DATA DW 123AH

77) @ 102

Packed - 0102H

Unpacked - 01 00 02 H

78-79) Convention

## Chapter - 02

- 27) Registers that are visible to programmers and can be accessed by programming, is known as program-visible registers.
- 1) 16  $11111111 - 000000$  (21)
  - 2) 80386  $11111111 - 000000$  (4)
  - 3) EBX  $111 + 000000$  (41)
  - 4) CX  $11111111$  (21)
  - 5) Instruction pointer (IP/EIP) holds the offset of the next instruction to be run.
  - 6) Increment/ Decrement
  - 7)  $\begin{array}{r} \text{FFH} \\ \text{01H} \\ \hline 100H \end{array}$  (21)
- No overflow will occur if it is 8 bit addition. In result we will get 1 as carry flag.
- Hence FFH will be considered as -1.

9) odd

so - notqual

10) I (Interrupt) flag

11) 3086 - 8038 - core2

12) it holds the base address

13) a 10000H - 1FFFFH

b 12340H - 2233FH

14) CSX10H + IP

15) IM

16) DI

17) EAX, EBX, ECX, EDX, ESI, EDI

18) SS + SP / ESP

19) SS

H77

H40

H300

20) segment X10H + offset

21) a DS = 2000H

$$EAX = 0000$$

b 3000H

$$\begin{array}{r} 20000 \\ + 3000 \\ \hline 23000H \end{array}$$

22) all area

23) any location

24) The segment register works as a selector to select a descriptor from the descriptor table

25)  $2^{13} / 8192$

26) starting  $\rightarrow 00000H$   
ending  $\rightarrow 01000H$

27) starting  $\rightarrow 01000000H$

ending  $\rightarrow 01000000 + 0FFFFH$

28) starting  $\rightarrow 00280000H$

ending  $\rightarrow 00280000H + 00010FFFH$

29)

0000	0000	0010	0000
0	0	1	0
selector = 4			
TF RPL			

30-31) same

32) 64k

33)

80286

0000H	FEH	T 21
0000H	001FH	

Access rights

P	DPL	S	E	ED C	R/W	A
1	11	1	1	1	1	0

FEH

03	G <sub>1</sub> 1	D 1	1	AV 10	F2	00
0000					8FFF	

34)

Access rights

P	DPL	S	E	ED C	R/W	A
1	11	1	0	0000 0100	0000 0000	1 0

- 35) GDTR (Global Descriptor Table Locator)
- 36) Through a descriptor at Global table
- 37) The microprocessor access the descriptor table using selector. Its base address, limit, and access rights are loaded into the program invisible segment register cache.
- 38) Registers that programmers can't access.
- 39) GDTR contains the base address of global descriptor table
- 40) 4K
- 41) CPO
- 42) 1024
- 43) 4K times
- 44) 16 bytes to compare of memory
- 45) Flat mode: Only segment + no offset
- 46) 1T

## Chapter 03

1) a) `Mov AX, BX`

It moves the content of BX  
into AX

b) moves 32 bit content

c) moves 64 bit content

2) AH, AL, BH, BL, CH, CL, DH, DL

3) AX, BX, CX, DX, SP, BP, SI, DI

4) EAX, EBX, ECX, EDX, ESP, EBP, ESI, EDI

5) RAX, RBX, RCX, RDX, RSP, RBP, RSI, RDI

6) CS, DS, ES, SS, FS, GS

7) Types doesn't match

8) Segment to segment not allowed directly

9) a) `Mov EDX, EBX`

b) `Mov CL, BL`

c) `Mov BX, SI`

d) `Mov AX, DS`

⑤ MOV AH, AL H0000 + H00X00 (cs)

⑥ MOV BX, BX H0000 + H00X00 (cs)

10) ⑦ MOV AL, 12H H0000 + H00X00 (cs)

⑧ MOV AX, 123AH H0000 + H00X00 (cs)

.... H0000 + H00X00 (cs)

11) Older assembler uses # of address (cs)  
Modern assembler uses like B/H (suffix)

12) Selects single segment model

13) .CODE directive

14) symbolic memory → AT&T X86 VDM (cs)  
bytes signs - AT&T T32770 X86 VDM

15) opcode H0010 + H0000 + (H00XH0000) (cs)

16) letter/some special characters (cs)

17) exit to DOS (cs)

18) Tiny model creates .COM program  
while small model creates .EXE program

19) STARTUP is used to load to data segment  
register with the segment address of DS

- 20) DS X 10H + 2000H      03 JA H A VOM (e)
- 21) memory address      23 CD H VOM (1)
- 22) (a) DS X 10H + 1234H      1E2F JA VOM (g) (a)  
(b) " + 0300H      1A85F X H VOM (d)  
(c) " + 0400H      ...
- 23) Memory-to-memory transfer not allowed.
- 24) MOV BYTE PTR [2000H]
- 25-27) same
- 28) MOV BX, DATA - copies the word  
MOV BX, OFFSET DATA - copies offset
- 29) (a) (2000H X 10H) + 1000H + 0100H  
(b) DS register is large enough (10)  
(c) SS
- 30) MOV AL, [BX][SI]  
is equivalent to B  
MOV AL, [BX+SI]  
So it will work (e)

31) (a)  $(12000 + 0109H)$

23168 2 (EE)

(b)  $12000 + 250 + 200 = 12350H$

32) (a)  $DS \times 10H + LIST + BX$

(b)  $DS \times 10H + LIST + BX + SI$  (or)

33) (a)  $SS \times 10H + BP + 200H$

(b)  $SS \times 10H + BP + SI - 200H$

34) BP / SP  
Set down no bid as beginning (IP)

35) same

36) FIELDS STRUCT

F1 WORD ?

-18 (SP)

F2 WORD ?

24 (IP)

F3 WORD ? (expands, goes to) (IP)

F4 WORD ?

(BL) goes to (IP)

F5 WORD ?

FIELDS ENDS

→ ends @ (IP)

37) MOV AX, FIELDS.F3

→ AX @ (IP)

38) 1 - Direct

→ AX @ (IP)

2 - Relative

→ AX @ (IP)

3 - Indirect

30) 5 bytes

opcode + offset + offset + segment + segment  
(low) (high) (low) (high)

40) Intersegment  $\rightarrow$  new segment

(new  $\rightarrow$  CS + IP)

Intrasegment  $\rightarrow$  same segment

(only IP)

41) signed 16 bit can reach  $\pm 32k$

which means it can access the entire memory

42) 32

43) far jump changes both CS and IP

44) short jump (1B)

45) a) short

b) Near

c) short (diff 2h)

d) Far

46) JMP BX

- 47) JMP WORD PTR [TABLE] easy (1)
- 48) 2B medium (2)
- 49) Pushes the content to the stack medium (3)
- 50) AX, CX, BX, DX, SP, BP, SI, DI medium (3)
- 51) Just like 30, but for 32 medium (3)
- 52) PUSHFD - Flag medium (3)
- 53) No [EA + X8], 20 (6)

22 - 92 ② (5)

22 - X87 ④

22 - 10 ⑤

22 - 983 ⑥

22 - 12 ⑦

X8 - 1A VOM (8)

[H0094 + E8] X8 VOM (9)

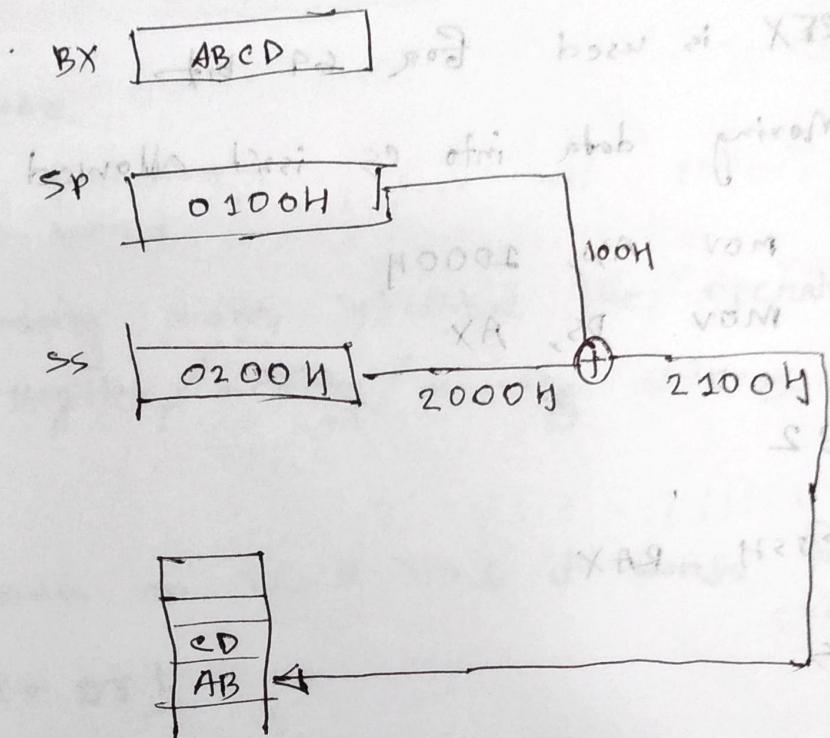
- 80 85 87 (10)

# Chapter 4

- 1) Opcode
- 2) D = Direction  
W = Word
- 3) Addressing mode, whether the operator is register/memory/memory address
- 4) DL
- 5) Depends on W, if  $W=1$  it works for 32 bit - DS
- 6) PS:[BX + DI]
- 7)
  - (a) SP - SS
  - (b) EBX - DS
  - (c) DI - DS
  - (d) EBP - SS
  - (e) SI - DS
- 8) MOV AL, BX
- 9) MOV BX, [SI + 4C00H]
- 10) 8B 77 02

- 11) Same
- 12) REX is used for 64 bit
- 13) Moving data into CS isn't allowed
- 14) MOV AX, 1000H  
MOV DS, AX
- 15) 32
- 16) PUSH RAX
- 17) CS
- 18) AX, CX, DX, BX, SP, BP, SI, DI
- 19) 32 bit (like 18)
- 20) ① AX to stack  
② stack to ESI  
③ [BX] to stack  
④ Flags to stack  
⑤ F stack to DS  
⑥ A to stack

21)



22) same

23) 2

24) Any possible combination of XA

25) not contents

LEA → offset

26) same

27) mov with offset

- 28) segment  $\rightarrow$  DS : to find offset [RE] offset .  $\rightarrow$  BX I<sub>EF</sub> BX D<sub>HDX</sub> (OP)
- 29) segment location DX (OP)
- 30) so easy, just mov operation = THRS (SP)
- 31) Direction flag: For string operation which the DI and SI increases/decreases NOC MARK.
- 32) STD  $\rightarrow$  set direction APAC.  
CLD  $\rightarrow$  clear direction APAC.
- 33) movs, emps
- 34) LODSB loads DS:SI into AL/AM
- 35) uses RAX
- 36) [DS:SI] to output (DX port)
- 37) SOSW = AX  $\rightarrow$  [FS:DI]
- 38) MOV CX, 12  
MOV LEA SI, SOURCE  
LEA DI, DEST ; good known
- CLD
- REP MOVSB

39) Iterates until cx is 0 ← loops  
 40) XCHG EBX, ESI      X8 ← . fact  
 41) DX                    pointer transfer  
 42) LAHF = Load AH with flag → flags or  
 SAHF = Store " " " "      Digits with  
 43) -MADE SMALL      IP from EC set  
 .STACK 100h  
 • DATA                    pointer to ← AT  
 Table DB 030h, ..., 39h ← C10  
 Res DB 10 DUP (?)      29h, 20h, 00h ← C10  
 .CODE  
 MAIN PROC      tried      error      error  
 mov ax, @data      XA9 ← 220h  
 mov ds, ax  
 (mov, SC) offset of C10C7  
 mov bx, offset Table      XA = W202  
 mov di, offset Res      10 ← 0000  
 mov cx, 10      10 ← 0000  
 mov al, 0      IP A31 ← 0000

ConvertLoop:

XLATB  
 mov [di], ah      | inc  
 inc di      | inc  
 loop convertloop

- 44)  $AL \leftarrow DS : [BX + AL]$  (3)  
45) Input port to AL (3)  
46) Output port (outputs one bit) (3)  
47) Using non-default segment (3)  
48)  $MOV AH, ES:[BX]$  (3)  
49) XCHG (3)  
50) Statement for assembler (3)  
51) conditional move (3)  
52)
  - (a) DB - Byte
  - (b) DW - Word
  - (c) DD - Doubleword  
53) LIST 1 DB 30 DUP(?)  
54) Constant  
55) enable 686 instruction  
56) memory model  
57) DS

- 58) depends on segment name ] : 2A → 1A (P)
- 59) terminates and return code to AL register (P)
- 60) PROC main } no stack } no error (P)  
main ENDP } no function pointer (P)
- 62) Store Proc } EX : 2A → 1A (P)  
MOV [DI], AL PDX (P)  
MOV [DI+1], AL  
MOV [DI+2], AL not segmentate (P)  
MOV [DI+3], AL non (P)
- Store ENDP
- ① DE - 89 (P)  
② BM - 99 (P)  
③ DD - 00 (P)  
④ T211 (P)  
⑤ 00 (P)  
⑥ 00 (P)  
⑦ 00 (P)  
⑧ 00 (P)  
⑨ 00 (P)  
⑩ 00 (P)  
⑪ 00 (P)  
⑫ 00 (P)  
⑬ 00 (P)  
⑭ 00 (P)  
⑮ 00 (P)  
⑯ 00 (P)  
⑰ 00 (P)  
⑱ 00 (P)  
⑲ 00 (P)  
⑳ 00 (P)

## Chapter 5

- 1) ① ADD AX, BX      1A H 9 00H (?)  
     ② ADD AL 12H      1B H 9 00H  
     ③ ADD EBP, EDI      1C H 9 00H  
     ④ ADD CX, 22H      1D H 9 00H  
     ⑤ ADD AL, [SI]      1E H 9 00H  
     ⑥ ADD ~~EFOG~~, CX      1F H 9 00H (?)  
     ⑦ ADD RCX, 239H      20 H 9 00H (?)
- 2) ~~Same size required - bottleneck for size~~ (?)
- 2) Different size
- 3) Segment registers are not allowing for arithmetic result stored directly (?) (?)
- 4) C - carry - 0  
    A - auxiliary - 0  
    S - sign - 0  
    Z - zero - 0  
    O - overflow - 0

- 5) MOV DH, AL  
ADD DH, BL  
ADD DH, CL  
ADD DH, DL  
ADD DH, AH
- 6) same
- 7-8) same
- 9) ADC DX, BX
- 10) INC SP
- 11) Size not defined. Correct one will be  
INC BYTE PTR [BX]
- 12) @ SUB CX, BX  
⑥ SUB DH, DEEH
- 13) C = 0  
A = 1  
Z = ①  
O = 0  
S = ②

- 14) MOV BX, AX      *bangfenz = JUM* (2)
- SUB BX, DI      *bangfie = JUM*
- SUB BX, SI      *1000XXA = XA* (2)
- SUB BX, BP
- 15) DEC EBX      *XA (2)*
- 16) Subtract with borrow      *XAX (2)*
- 17) SUB - stores result      *and no zero ①* (85)  
CMP - ~~only~~ changes flag *offnow ②*
- 18) AX      *bangfenz = VCD* (2)
- 19) DX:AX      *Bangfie = VIDE*
- 20) If upper half exists,  
then C=1, D=1      *HA (2)*
- 21) RD<sub>X</sub>:RAX      *XAX (2)*
- 22) MOV DL, 5      *O, HA VDM (2)*  
MOV AL, DL      *JP JA VDM*  
MUL DL      *JA, JA QDA*  
MOV BL, AL      *JR, JR VDM*  
MUL BL      *O, HA VDM*

- 14) MOV BX, AX      *bangianu = JUM* (es)  
SUB BX, DI      *bangia - JUM*  
SUB BX, SI      *000XX8 - X8* (es)  
SUB BX, BP      *X8* (es)
- 15) DEC EBX      *X8* (es)
- 16) Subtract with borrow      *XAS* (es)
- 17) SUB - stores result  
CMP - ~~only~~ changes flag *of noisivib ①* (es)  
*of noisivib ②* (es)
- 18) AX      *DX, AX*      *DX = V1D* (es)
- 19) DX:AX      *DX = V1D*
- 20) If upper half exists,  
then C=1, D=1      *HR* (es)  
*XAS* (es)
- 21) RDX:RAX      *O.HA VOM* (es)
- 22) MOV DL, 5      *JP JR VON*  
MOV AL, DL      *JL V1G*  
MUL DL      *JA JR GGA*  
MOV BL, AL      *JR JS VGM*  
MUL BL      *O.HG VGM*

- 23) MUL - Unsigned       $\times A \quad \times B \quad \text{volt}$  (A)  
IMUL - signed  
 $\times A \quad \times B \quad \text{volt}$   
 $\times A \quad \times B \quad \text{volt}$   
 $\times A \quad \times B \quad \text{volt}$
- 24) BX = DX X 100 n
- 25) AX
- 26) AX
- 27) RAX      warning after writing
- 28) ① division by zero      divisor register = 803 (1)  
② overflow of registers      divisor = 900  
B
- 29) DIV = Unsigned       $\times A \quad (8)$   
IDIV = signed       $\times A : \times D \quad (8)$
- 30) AH      divisor then move to (c)  
 $A = 0 \quad D = 9 \quad \text{volt}$
- 31) RAX
- 32) MOV AH, 0       $\times A : \times D \quad (1)$   
MOV AL, BL  
DIV CL  
ADD AL, AL  
MOV DL, AL  
MOV DH, 0

33) DAA  
DAS  
AAA  
AAS  
AAM  
AAD

34) AAM converts into packed BCD digits

35) AAA, AAS, AAM, AAD

36) MOV CX, 5  
MOV SI, 10000

Next-digit:

XOR DX, DX

DIV SI

ADD AL, 30H

MOV [BX], AL

INC BX

MOV AX, DX

MOV DX, 0

MOV SI, SI/10

LOOP Next-digit

37) ADD DX, BX  
DAA  
ADD CX, AX  
DAA

38) NO

AAA

AAQ

AAA

39) AND BX, DX

AAA

AAA

AAA

40) ⑥ AND DH, DEAH

AAA

⑦ AND DI, BP

AAA

⑧ AND EAX, 1122H ORI ZEROS MMX

⑨ AND [BP], CX AAA, MAA, ZAA, AAB

⑩ AND DX, [SI-8] A, X, VOM

⑪ AND WHAT, AL 0000D, IR VOM

40)

MOV BH, DH

X, X, FOX

AND BH, 1FH

IE VIB

41) ① OR AH, BL

HOB, IA QCA

② OR ECX, 88H

-1B, [EX-1] VOM

③ OR SI, DX

KET SNC

④ OR BP, 112H

X, X, VOM

⑤ OR [RBX], RCX

Q172, IE VOM

⑥ OR AL, [BP+40]

X, X, QCA @ (S)

⑦ OR WHEN, AH

AAQ

X, X, QCA

AAQ

- 42) MOV SI, DI      AND TEST (EP)  
OR SI, 001FH      AND TEST & CF ← TOH (EP)
- 43) ① XOR AH, BH  
② XOR CL, 99H      E, JA, JNE ③ (EP)  
③ XOR DX, DL      E, JA, JNE ④  
④ XOR RSP, 1A23H      E, JA, JNE ⑤  
⑤ XOR [EBX], DX      E, JA, JNE ⑥  
⑥ XOR DI, [BP + 60]      E, JA, JNE ⑦  
⑦ XOR DI, WELL      E, JA, JNE ⑧  
23 (EP)
- 44) ① BTS AX, 0  
BTS AX, 1      loops slides things (EP)  
BTS AX, 2      loops slides things (EP)  
BTS AX, 3
- ② AND AX, 1FFH  
③ XOR AX, 0380H      loops from RL (EP)
- 45) AND stores result  
but TRGT doesn't  
[place new ERG]

- 46) TEST CH, 4      IC, IS - NOX (2)
- 47) NOT  $\rightarrow$  1's complement  
NEG  $\rightarrow$  2's complement      AT 00 IS 90
- 48) (a) SHR DI, 3      HEE 40 90X (2)  
(b) SHL AL, 2      LD, XA 90X (2)  
(c) ROL AL, 3      HEDAC 90X 90X (2)  
(d) RCR EDX, 1      XD, [XBX] 90X (2)  
(e) SAR DH, 2      [D + 98], IS 90X (2)
- 49) Accumulator vs memory      IC 90X (2)
- 50) ES      XA 278 (2)
- 51) D=0  $\rightarrow$  increment      XA 278  
D=L  $\rightarrow$  decrement      XA 278  
E, XA 278
- 52) Repeat while equal      P777D, XA DMA (2)
- 53) If not equal ( $ZF = 1, CX = 0$ ) 90X (2)
- 54) compares string from es to ch (2)  
[DS:SI] with [ES:DI]

55) MOV CX, 300H      20 instruction  
MOV AL, 66H  
LEA DI, LIST  
CLD      set word aligned file of 256  
REPNZ SCASB      nothing - throws

56) 43H is displayed

## Chapter - 06

- 1) Short jump transfers control, within -128 to 127 bytes from the CS current position
- 2) near
- 3) far
- 4) far
- 5) For 32-bit mode  
it's around 2G
- 6) a) Near  
b) Short  
c) Far
- 7) symbolic name of an address
- 8) IP
- 9) CS + IP
- 10) near

- 11)  $\text{JMP DI}$  - register indirect
- $\text{JMP [DI]}$  - memory indirect
- 12)  $\text{JMP [DI]}$  - only offset (near jump)
- $\text{JMP FAR PTR [DI]}$  - far jump
- 13) C, A, O, S, Z
- 14) CF=0 and ZF=0
- 15) OF=0 overflow flag, OF=1
- 16) JG, JGE, JLE, JL, JO, JNO, JNS, JE/JZ, JN
- 17) JA, JAE, JB, JBE, JC, JNC, JE/JZ, JNE/JN
- 18) JA, JBE
- 19) HF CX=0
- 20) SETZ AL
- 21) CX
- 22) ECX
- 23) RCX

- 24) MOV AX, ES  
MOV DI, OFFSET DATAZ  
MOV AL, 00H  
MOV CX, 150H  
CLD  
STOSB  
LOOP \$-2
- 25) LOOPE only loops if the prev. operation generates equal flag
- 26) CMP AL, 3  
JNE NEXT  
ADD AL, 2
- 27) MOV CX, 100H  
MOV SI, OFFSET BLOCKSTART  
MOV AL, 42H  
MOV DI, 0 ; UP count  
MOV BX, 0 ; DOWN count

### COMPARE\_LOOP:

```
CMP [SI], AL
JA IS_UP
JB IS_DOWN
JMP NEXT_BYTE
```

IS-UP:

INC DI  
JMP NEXT-BYTE

IS-DOWN:

INC BX

NEXT-BYTE:

INC SI

LOOP COMPARE-LOOP

MOV UP, DI

MOV DOWN, BX

28)

MOV SI, OFFSET BLOCKA

MOV DI, OFFSET IN BLOCKB

REPEAT

LODSB

STOSB

UNTIL AL=00H

29) infinity loop

MOV SI, OFFSET BLOCKA

MOV DI, OFFSET BLOCKB

CLD

. WHILE AL != 12H

LODSB  
ADD AL, [DI]  
MOV [DI], AL  
INC DI  
. ENDW

31) Breaks a loop

32) Procedure: Procedure is a reusable section of the code. They generally performs one task. It is stored once in memory but can be used as much as necessary.

33) Near call: Transfers flow within the code segment.

Changes only IP

Far call: Transfers to a new block changes CS + IP.

- 34) RET (Return) (CP)
- 35) RET removes the return address from the stack so the program returns.
- 36) By using NEAR or FAR (XAT 9A)
- 37) PROC
- 38) CUBE PROC NEAR JA JNFZ?
- ```
    MOV AX, CX
    MUL CX
    MUL CX
    MOV CX, AX
    RET
CUBE ENDP
```
- 39) RET G is to clean up parameters that were pushed onto the stack by the caller before the procedure was called. (SP)
- 40) CALC PROC NEAR USES SI DI BX CX (CP)
- ```
    MOV AX, DI
    MUL SI
    MOV CX, 100H
    DIV CX
    RET
CALC ENDP
```

41) SUMS PROC NEAR (USES EBX ECX EDI EDX)  
MOV EDI, 30H ADD EAX, EBX ADD ECX ADD EDX  
SETNC AL MOV EDI, 0 RET  
SUMS ENDP

42) Interrupt is a hardware-initialized function call on a hardware-initiated procedure

43) INT, INT0, INT3  
44) 256 interrupt offset address  
45) Interrupt = 4 byte pointer  
↳ offset  
↳ segment

46) 0<sub>4</sub> refers to ~~in the divide errors~~ (P2)

47) RET: RET only restores the instruction pointer

~~when control returns to memory~~ IRET: Used as an exception handler.

~~int word~~ Also pops code segment and flag register

48) 32-bit return

Value  $\leftarrow$  92 (Q2)

49) 64-bit return

50) INTO only interrupt when overflow

51) Address :  $40H \times 4 = 100H$

Se location:

$100H - 103H$

52) STI  $\rightarrow$  Set Interrupt flag

CTI  $\rightarrow$  Clear interrupt flag

53) Wait instruction

54) Bound interrupt will interrupt  
a program if the value in the  
register or memory location under  
the test is above or below the  
boundaries.

55) 16

56) BP → stack