1(D) - Day

Date: 3/5/2017

Microprocessors and Assembly Language

Dr. Nazrul Islam Mondal

Ref. Book:

Charles Manut

BIOS → software
↘ ROM এ থাকে

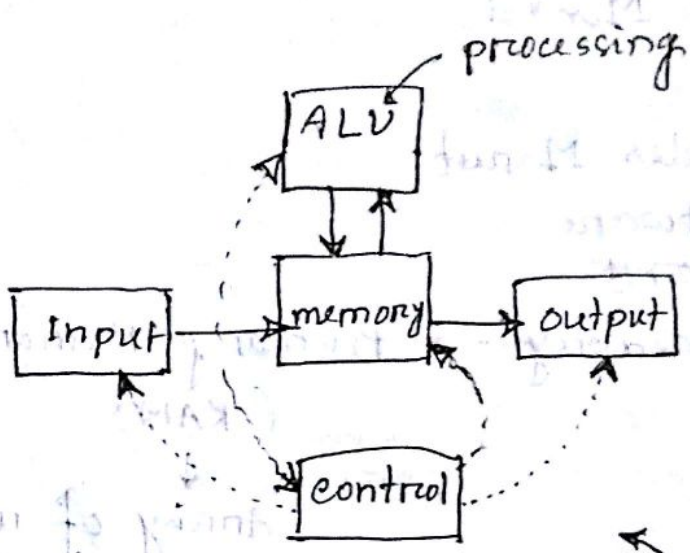
Execution memory → Primary Memory
(RAM)
↓
Array of resisters

Access → read + write

Intel 8086/8088

2(A)-Day Date: 7/6/2017

von Neumann Architecture:



→ Sequential operation (sequential machine)

MAR → Memory Address Register (Data ~~read~~ ~~write~~)

MDR → " " Data " "

Cache Memory → one type of register
(processor এর কাছে)

Basic Performance Equation

$$T = \frac{N \times S}{R}$$

Intel 4004 → 4 bit

⊞ Difference between multi-core and many-core processon

Intel 8086

Address bus – 20 bit → segment & offset → 32 bit এর result 20 bit এর সমান হলে relate & problem হইয়ে থাকে

Data bus – 16 bit

Control bus –

Flags-register

16-bit register এর ৭ টা flag

Opt code → operation কি হবে তা বলে দিবে

operands → যার উপর operation করতে হবে

- MODEL SMALL
- 100H
- Data

চেপ্টার 4th chapter → 4.9

- MODEL SMALL
- STACK 100H
- DATA

A DW 2

B DW 3

SUM DW ?

- CODE

AL → Accumulation Register

BL → Base Register

CL → counter

DL → Data

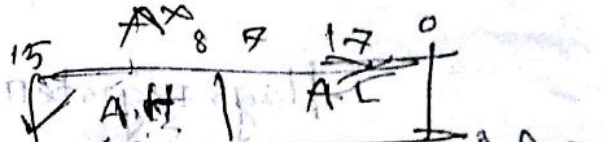
AH →

MAIN PROC

— — —

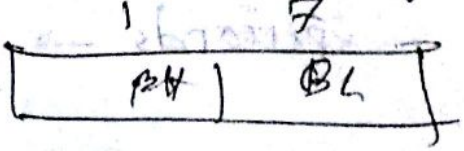
MOV DL, AL

INT 21H



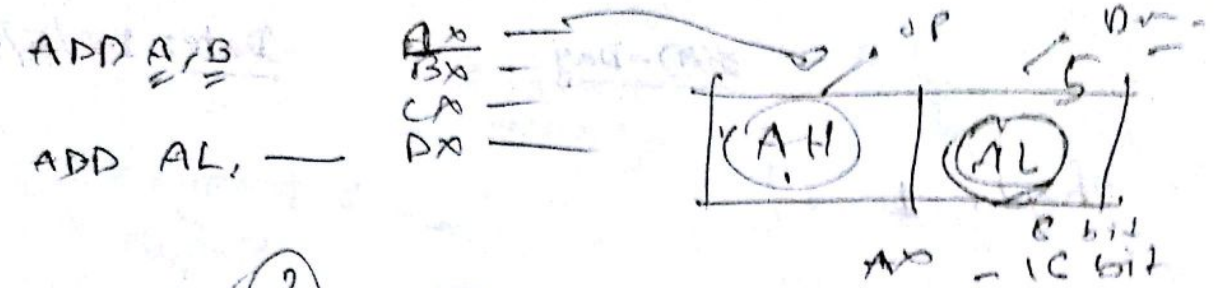
mov AH, 2 → o/p

1 → I/p



ADD A, B

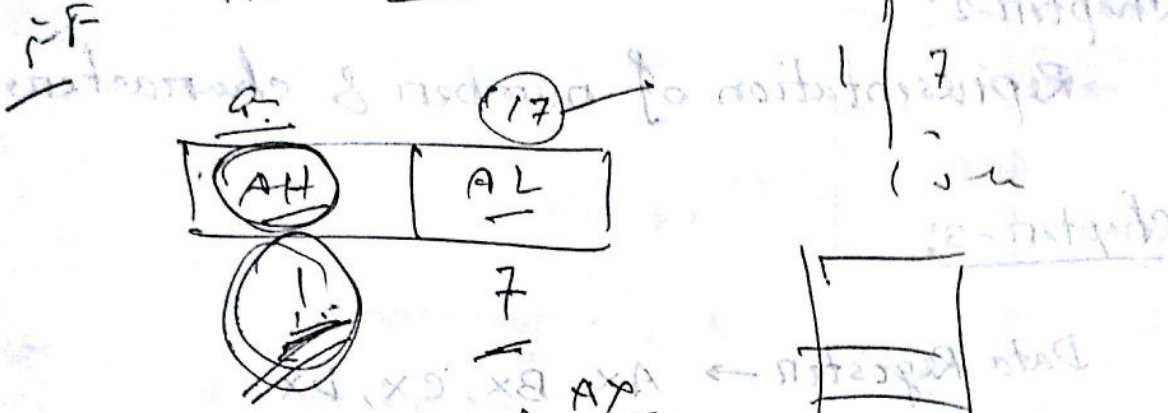
ADD AL, —



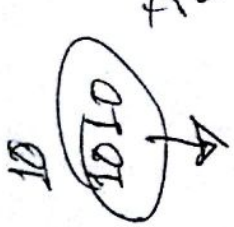
(2 3) (5) DL

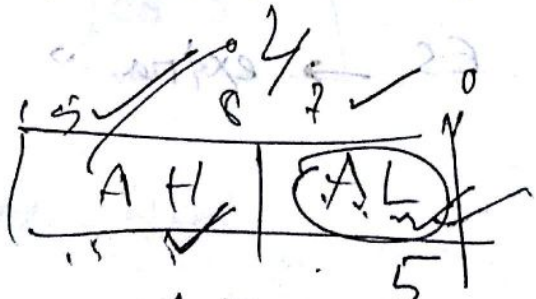
5

AL — 017



AL





3(A)-Day Date: 15/5/2017

Chapter-1

Chapter-2:

→ Representation of number & characters.

Chapter-3:

Data Register → AX, BX, CX, DX

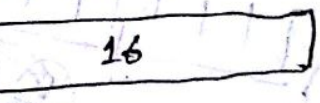
Segment Register → 4

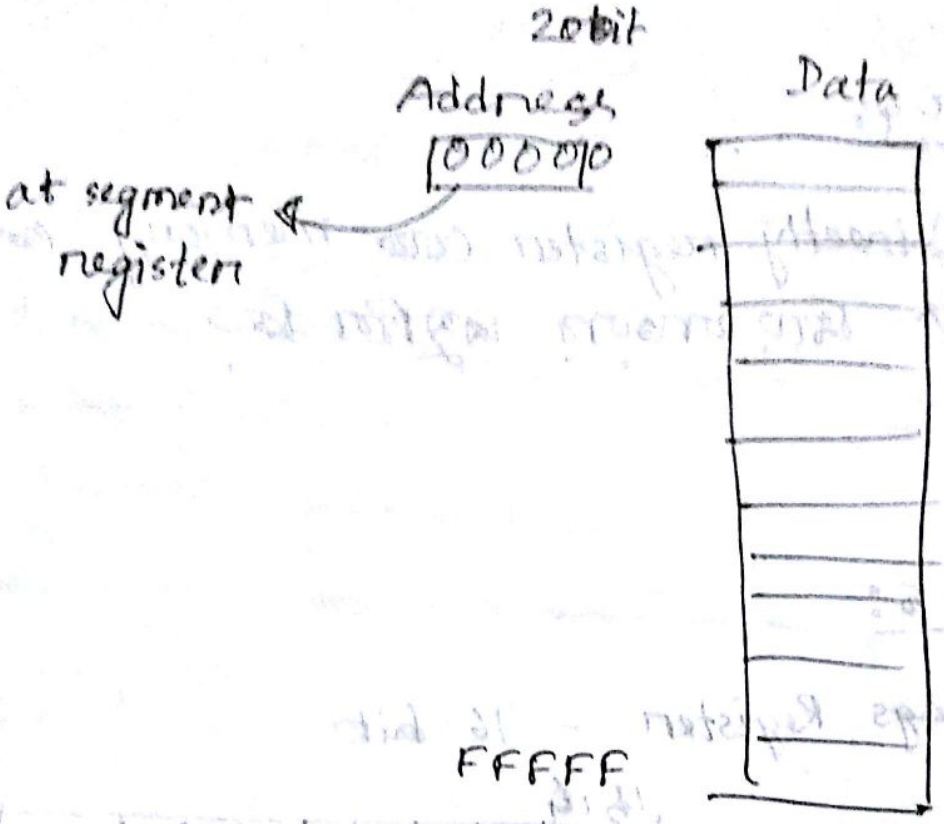
CS → core segment

DS → data "

SS → stack "

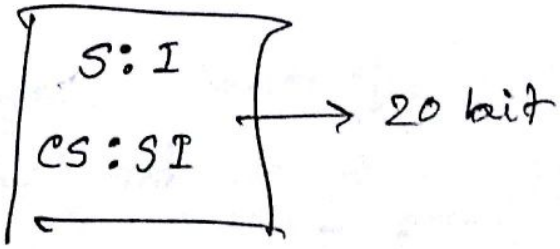
ES → extra "





→ Indexing register

SI → Source Index
DI → Destination "
SP →
DP →



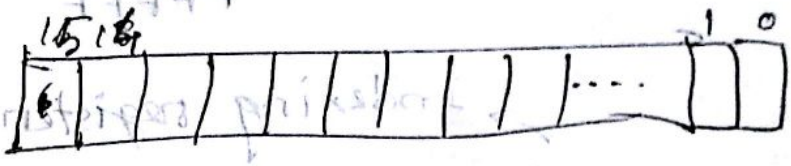
Startup operation

Chapter-4:

Directly register থেকে memory তে data নিয়ে যাওয়ার প্রক্রিয়া কি?

Chapter-5:

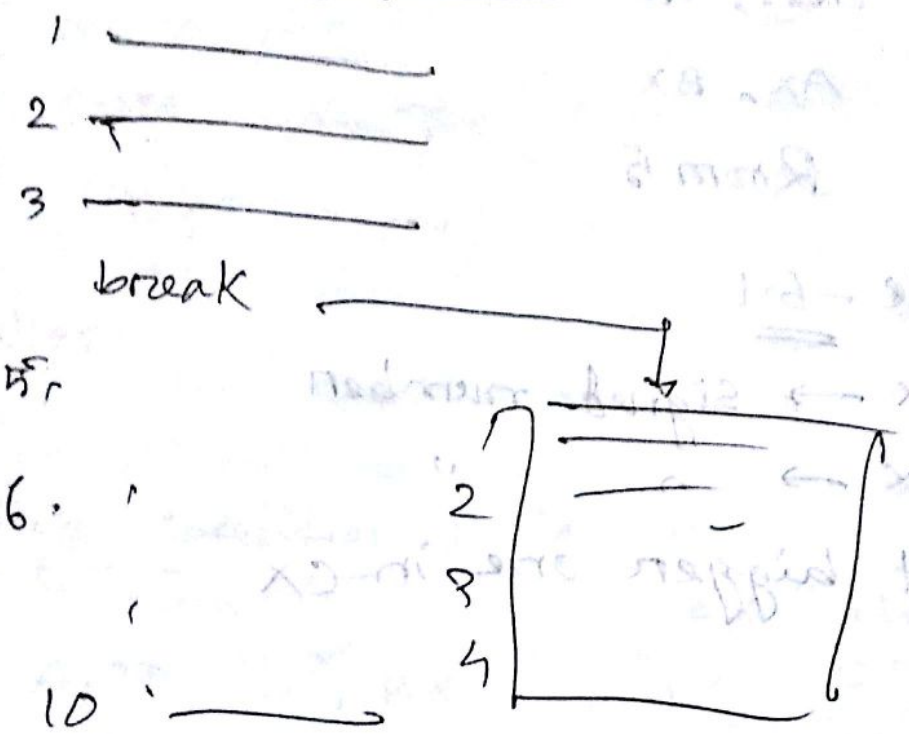
Flags Register — 16 bit



control flag — 3 (operation control)
status " — 6

4(B)-Day Date: 03/7/2017

Chapter-6



Control Instructions:

1. Conditional control
   - → Signed control
   - → Unsigned "
   - → Flag " [for individual bit]
2. Unconditional control

117

compare

CMP dest, src

CMP AX, BX

JLE Room 5

Example - 6.1

AX → signed number

BX → " "

Put bigger one in CX

JA

Data

5

10

2

4

20

21

(20)

Jump if above (20)

JZ → Jump if zero

if zero flag = 0

Example 6.3

6.4

Unconditional

```
MOV  AX, BX
SUB  AX, BX
JMP  Next
Next:
```

Plug Conditional Control

```
AND  AX, BX
OR   AX, BX
```

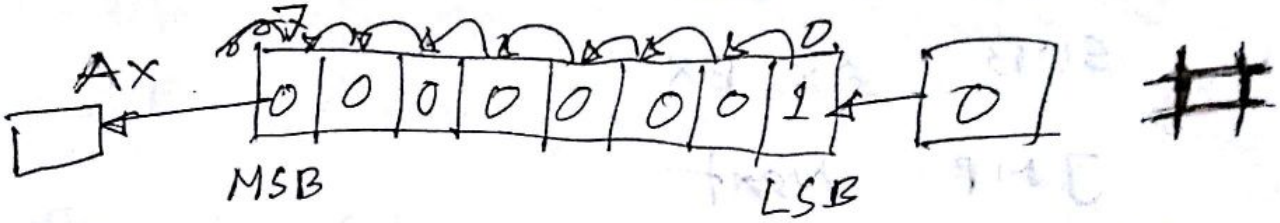
AX 1111
BX 0001
AND → 0001

```
XOR  AX, BX
```

↓

0 বসানোর সহজ উপায় (XOR AX, AX)

Chapter-7



00000010
00000100
00001000
⋮

} Left shift

SHL ↓

ভাবের value এক double হয়ে যাবে

SHR ⟶ ভাবের value এক অর্ধেক হবে

NOT → Negative করে (5 to -5)

Example 7.8

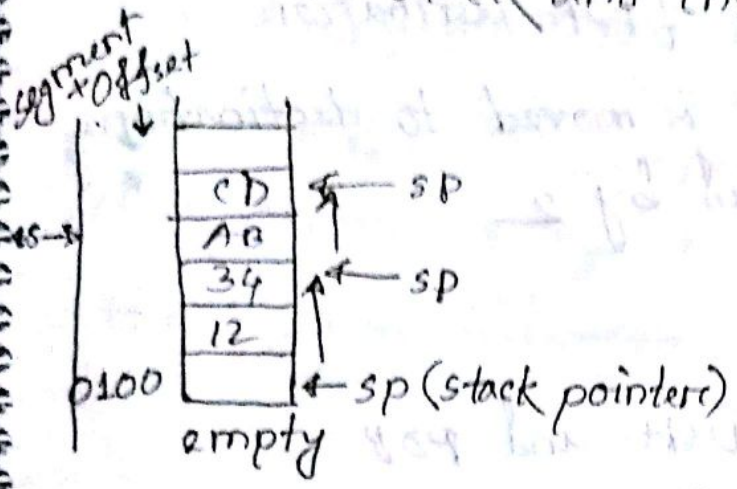
```
MOV  CL,3
SAL  AX,CL
```

4(E)-day

Date: 8/7/2017

Chapter-8

The stack and introduction to procedure



SS + SP
32 bit

- STACK 100H দ্বারা stack কে empty করা top address 0100.

PUSH source

source → Register, Memory

- PUSH AX; AX = 1234
- PUSH BX; BX = ABCD

1. sp is decreased by 2
2. A copy of source content is moved
3. to the address specified by ss:sp source unchanged

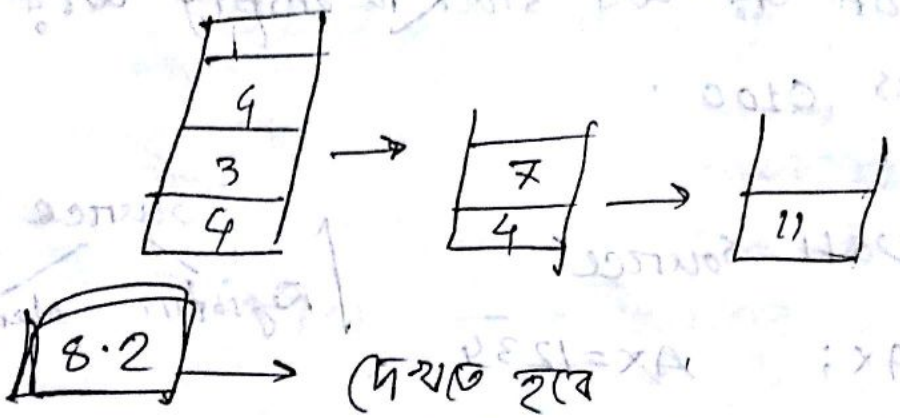
• PUSHF / F → Flag

• POP AX / POP destination

1. SS:SP ⇒ content is moved to destination
2. SP is decreased by 2

POP F

Addition using PUSH and POP

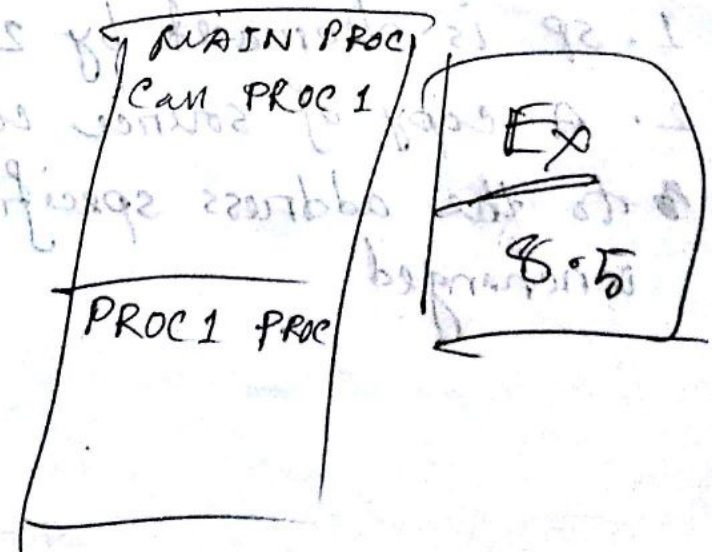


দেখতে হবে

Procedure (Function)

8.3

MAIN PROC
Call PROC 1
next inst
PROC1 PROC
1st inst
RET



Ex
8.5

Chapter-9

Multiplication and Division Instructions

MUL Source → Unsigned
IMUL Source → Signed

Ex. MUL BX
IMUL BX

for byte multiplication:

1. One number is contained in source
2. Other number is assumed to be in AL
3. Product in AX

For word multiplication:

1. One number is contained in source
2. Other is assumed to be in AX
3. Product in DX:AX

n byte x n byte = max 2n byte

" + " = n+1 byte

DIV Source
IDIV Source

CT-2

Chapter-4

Introduction to IBM PC Assembly Language

statement

- Instruction → which the assembler translates into machine code
- Assembler directive → which instructs the assembler to perform some specific tasks.

Both instructions and directives have up to four fields:

name	operation	operand(s)	comment
START:	MOV	CX, 5	; initialize counter

Operation

for an instruction →

Symbolic operation code (opcode)

→ The assembler translates a symbolic opcode to a machine language.

Example: MOV, ADD, SUB

For an assembler directive →

→ Pseudo-operation code (pseudo-op)
→ are not translated into machine code.

Example: PROC

Operand

For instruction

```
NOP          ; No operand, does nothing
INC AX       ; one operand
ADD WORD1,2  ; two operands
```

destination operand

source operand

For Assembler directive

The operand field usually contains more information about the directive.

Comment

The comment field of a statement is used by the programmer to say something

Binary → 1001B (or B)

Decimal → 123d (or D)

Hexadecimal → 0ABh (or H)

↳ must be decimal digit. Otherwise assembler would be unable to differentiate between variable and hexadecimal number.

Characters

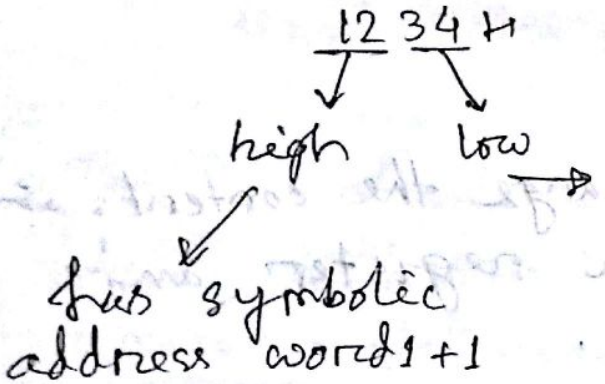
'Helo' / "A"

Pseudo-op		stands for
DB	→	define byte
DW	→	" word
DD	→	" doubleword
DQ	→	" quad word
DT	→	" tenbytes

* In assembly language, an array is just a sequence of memory bytes or words.

```
B_array  DB  10h, 20h, 30h
```

* High and low bytes of a word



* EQU (Equates)

```
name  EQU  constant
LF    EQU  0AH
```

[0AH → line feed character]

The name LF may now be used in place of 0AH anywhere in the program.

→ assigns the name LF to 0AH.

* MOV

→ MOV instruction is used to transfer data between registers.

MOV dest, source

* XCHG

→ is used to exchange the contents of two registers, or a register and a memory location.

XCHG AH, BL

* MOV and XCHG between memory location is not allowed.

MOV word1, word2 ✗

* ADD and SUB

ADD dest, source

SUB dest, source

* INC and DEC

INC destination
DEC destination

* NEG

→ is used to negate the contents of the destination.

NEG dest.

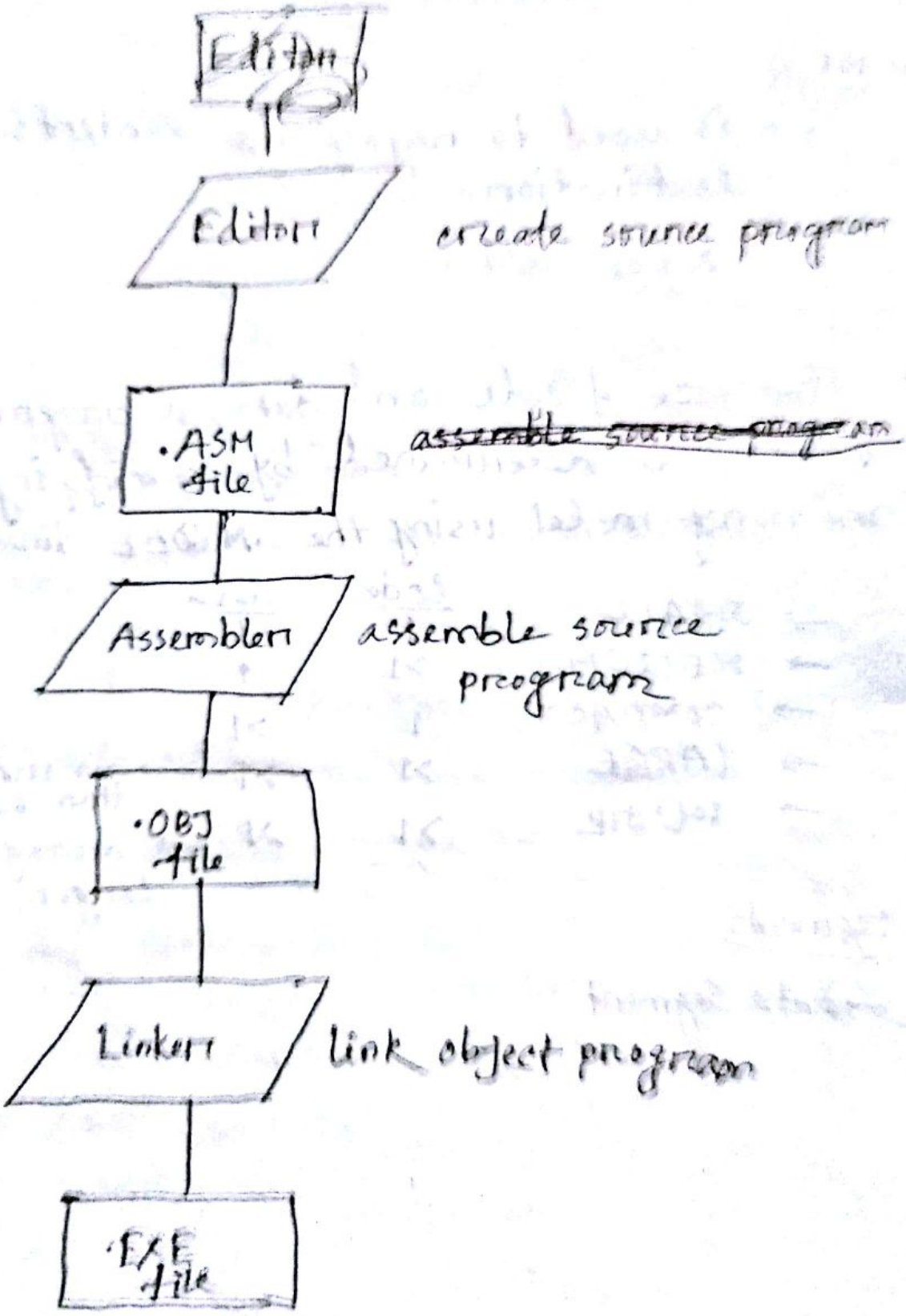
* The size of code and data a program can have is determined by specifying a memory model using the .MODEL directive.

	Code	data	
→ SMALL	1	1	
→ MEDIUM	>1	1	
→ COMPACT	1	>1	
→ LARGE	>1	>1	→ no array larger than 64K bytes
→ HUGE	>1	>1	→ arrays may be larger than 64K bytes.

Segments

→ Data Segment

Programming steps



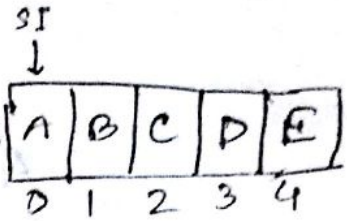
9(A) - Day

Date: 13/8/2017

The String Instruction

STR1 DB 'ABCDE'

1 byte



DB →

Seg:	Off	
	0200	A
	0201	B
	0202	C
	0203	D
	0204	E

DW →

0200 A

0202 B

↓

DF (Direction Flag) → string operation

DF = 0, Left to right

DF = 1, right to left

CLD → clear Direction Flag

STD → Set " "

SI → Source

DI → Destination

DS: SI → Source

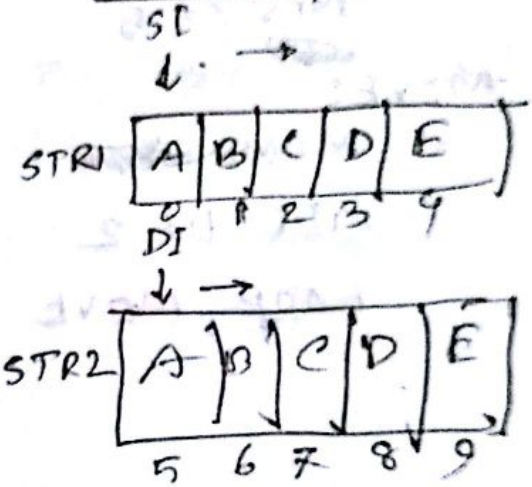
ES: DI → Destination

~~MOD~~ ~~MOV SB~~ MOVSB

```
MOV AX, @DATA
MOV DS, AX
MOV ES, AX

LEA SI STR1
LEA DI STR2

CLD

MOVSB
MOVSB
MOVSB
MOVSB
MOVSB
```

```
CLD
LEA SI STR1
LEA DI STR2

MOV CX, 5
REP MOVSB
```

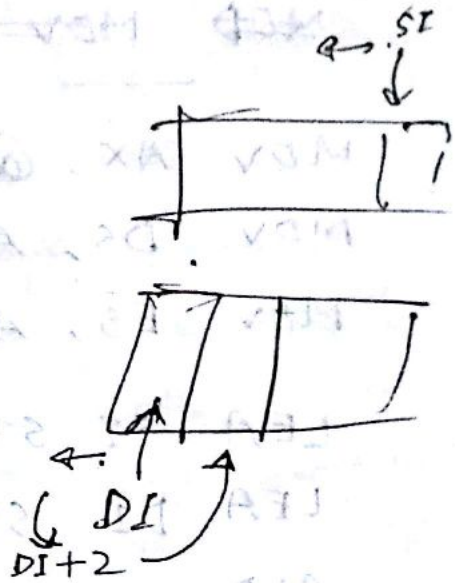
After MOVSB



Offset add. handle করতে হবে SI এবং DI তখন করতে হবে।

Reverse

```
LEA SI STR1+4
LEA DI STR2
STD
MOV CX, 5
CLD
MOVE:
MOVSB
ADD DI, 2
LOOP MOVE
```



Store String

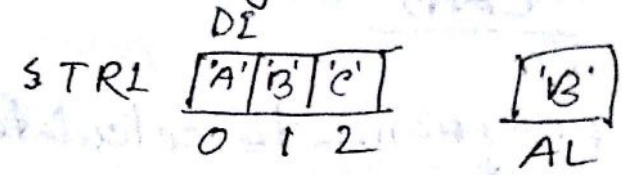
STOSB

Move the content of AL to the address ES:DI

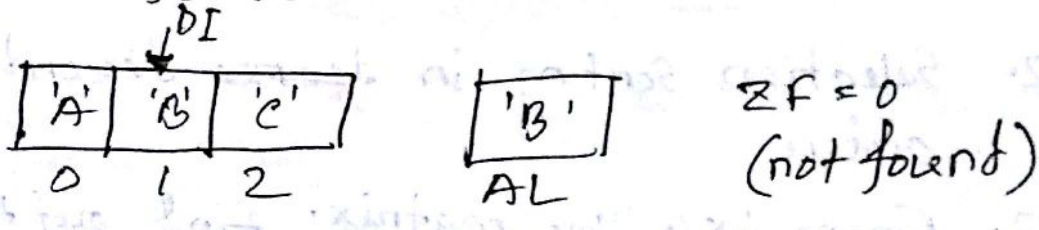
LODSB (Mem to reg.) [load string]

DS:SI → [AX]

SCASB (Scan string)



After SCASB



CMPSB (Compare string)

2(b)-Day Date: 20/8/2017

LAB

1. Write a program to calculate factorial N where N<10
2. Selection Sorting in ~~decrea~~ descending order
3. From 4x4 by matrix. find out the maximum and minimum element.

~~BX~~ , (N)

N(N-1)(N-2)

~~MUL CX, CX-1~~

N×N

```
.MODEL SMALL
.DATA
   x  dw 5
..
.code
      mov  ax, @data
      mov  ds, ax        → counter
      mov  cx, x  }
           =5     }  5
      mov  ax, x  }      → result
HERE:
      DEC x          ; x = 9
      MUL x          ; ax = (x * ax)
      DEC cx                ax = 5 x 4 = 20
      JNZ HERE              [cx = ?]
      mov  ax, 4CH
      int  21H
      end
```

POP DX

DIV DL

Scanned by CamScanner

Prob (3)

(i) input: WE ARE CSE14

Output: 41ESC ERA EW

(ii) search for CSE14

11(A) - Day Date : 29/8/2017

Direct Memory Access (DMA)

Addressing mode → chapter 10

Interrupt structure } → After Eid

RISC CISC

Memory → RAM

Process → Computing (ALU)

DMA থাকে I/O কার্য CPU এর প্রয়োজন

Acknowledgement — Data ঠিকভাবে সম্পন্ন হয়েছে

Control Instruction

- Opcode
- Operand

Data bus

Address bus

DMA → improvement performance in terms of time.

11(D)-Day Date: 11/9/2017

Use of RISC Today

Cache memory → RAM ও Processor
এর মাঝখানে

RISC PowerPC CISC

RISC → Reduced Instruction Set Computers
RISC architecture → no cache needed.

PowerPC → Improved version of RISC
→ integer এর জন্য আলাদা ALU
→ float " " " ALU

CISC → Complex Instruction Set Computers
Like →
ADD AX. BX. CX. DX. EX (যদিও)