**Assembly language Lab**

**Lab 01:**

Objective: Learn and apply the fundamentals of Assembly language

structure.

Tasks:

1.Create a C program with a basic structure.

2.Develop a basic structure for an assembly language program and compare it

with the structure of a basic C program.

Assessment: Write it 10 times and understand each line thoroughly.

**1) খুব সিম্পল C প্রোগ্রাম (DOS/Turbo C স্টাইল — একক কী পড়বে)**

#include <stdio.h>

int main(void) {

char a,b,c;

a = getch(); /\* একবার কী চাপলেই পড়বে \*/

putchar(a); /\*\_echo\*/

b = getch();

putchar(b);

c = getch();

putchar(c);

return 0;

}

**2) হুবহু আপনার স্টাইলে Assembly (MASM/TASM, register-only, কোনো msg/db নেই)**

.model small

.stack 100h

.data

.code

main proc

; --- first input ---

mov ah,1

int 21h

mov bl,al

mov ah,2

mov dl,0Ah

int 21h

mov ah,2

mov dl,0Dh

int 21h

; --- second input ---

mov ah,1

int 21h

mov bh,al

mov ah,2

mov dl,0Ah

int 21h

mov ah,2

mov dl,0Dh

int 21h

; --- third input ---

mov ah,1

int 21h

mov cl,al

mov ah,2

mov dl,0Ah

int 21h

mov ah,2

mov dl,0Dh

int 21h

; --- display inputs ---

mov dl,bl

mov ah,2

int 21h

mov dl,bh

mov ah,2

int 21h

mov dl,cl

mov ah,2

int 21h

; --- exit ---

mov ah,4Ch

int 21h

main endp

end main

**3) খুব সংক্ষিপ্ত মেপিং (C ⇄ ASM)**

* char a,b,c; ⇄ আমরা BL,BH,CL রেজিস্টার ব্যবহার করেছি।
* getch() ⇄ mov ah,1 / int 21h (AL-এ কী চলে আসে)
* putchar(x) ⇄ mov dl,x / mov ah,2 / int 21h
* return 0; ⇄ mov ah,4Ch / int 21h

**4) Assessment — 10 বার প্র‍্যাকটিস (সহজ পদক্ষেপ)**

১. দুইটি কপি প্রিন্ট করে — একবার C, একবার ASM।  
২. প্রতিবার লিখলে প্রতিটি লাইনের পাশে বাংলা হিসেবে কয়েক শব্দে কাজটি লিখুন।  
৩. ১ম বার: ঠিক যেমন আছে কপি করুন।  
4. ২য়-৫ম বার: শুধু রেজিস্টার পরিবর্তন করে লিখুন (e.g., BL→DL), দেখুন কোথায় ভুল হয়।  
5. ৬-৮ বার: ইনপুট এর সংখ্যা বদলান (2 টা বা 4 টা) — কিভাবে কোড বদলাতে হয় লক্ষ্য করুন।  
6. ৯-১০ বার: লাইনের কাছাকাছি দিয়ে ; দিয়ে বাংলা কমেন্ট লিখে বলুন “এই লাইনের কাজ কী” — মুখস্ত করলেই হয়ে যাবে।

Lab 02:

Objective: Learn and apply the fundamentals of Assembly language

structure.

Tasks:

1.Understand the basic structure and functionality of Data and Segment

Registers.

2.Learn and practice the following tasks: Entering a value and Displaying it,

using the Exit command, implementing the Interrupt command, and working

with Carriage Return and Line Feed.

Assessment: Take an input from user and display it.

## 🔹 Lab 02

**Objective:** Assembly-এর Data ও Segment Registers এর কাজ বোঝা, Interrupt ব্যবহার, Carriage Return (0Dh) আর Line Feed (0Ah) এর ব্যবহার প্র্যাকটিস করা।

**Task:** ইনপুট নেব, তারপর সেটাই আউটপুট করব।

### 👉 Assembly Code (হুবহু আগের স্টাইলে)

.model small

.stack 100h

.data

.code

main proc

; --- take input (user presses 1 key) ---

mov ah,1 ; DOS service: input char

int 21h ; AL এ ক্যারেক্টার আসবে

mov bl,al ; BL এ রেখে দিলাম

; --- newline (CR+LF) ---

mov ah,2

mov dl,0Ah

int 21h

mov ah,2

mov dl,0Dh

int 21h

; --- display input back ---

mov dl,bl ; BL থেকে DL এ

mov ah,2 ; DOS service: print char

int 21h

; --- exit program ---

mov ah,4Ch ; DOS terminate program

int 21h

main endp

end main

### 👉 ব্যাখ্যা (লাইন বাই লাইন)

* mov ah,1 / int 21h → কী-বোর্ড থেকে ইনপুট নিবে, AL রেজিস্টারে রাখবে।
* mov bl,al → ইনপুটটাকে BL-এ রাখলাম যেন পরে ব্যবহার করা যায়।
* mov ah,2 / mov dl,0Ah + mov dl,0Dh → Newline (Line Feed + Carriage Return)।
* mov dl,bl / mov ah,2 / int 21h → BL-এর ভ্যালু আউটপুট করবে।
* mov ah,4Ch / int 21h → প্রোগ্রাম বন্ধ হবে।

Lab 03:

Objective: Learn and apply the fundamentals of Assembly language

structure.

Tasks:

1.Declare a string in the data segment and display it from the code segment.

2.Take two inputs from the user and display their sum.

Assessment:Take two inputs from the user and display their subtraction.

## 🔹 Lab 03

**Objective:** Assembly language structure বুঝে data segment-এ string declare করা, দুটি ইনপুট নেওয়া ও গাণিতিক কাজ করা।

### 1) ****Task 1 → Declare a string & display it****

.model small

.stack 100h

.data

msg db 'Hello Assembly! $' ; string declare

.code

main proc

; initialize DS

mov ax,@data

mov ds,ax

; print string (AH=09)

mov dx,OFFSET msg

mov ah,09h

int 21h

; exit

mov ah,4Ch

int 21h

main endp

end main

### 2) ****Task 2 → Take two inputs & display their SUM****

.model small

.stack 100h

.data

.code

main proc

; first input

mov ah,1

int 21h

sub al,30h ; ASCII → number

mov bl,al

; second input

mov ah,1

int 21h

sub al,30h ; ASCII → number

mov bh,al

; sum = BL + BH

mov al,bl

add al,bh

add al,30h ; number → ASCII

; newline

mov ah,2

mov dl,0Ah

int 21h

mov dl,0Dh

int 21h

; display result

mov dl,al

mov ah,2

int 21h

; exit

mov ah,4Ch

int 21h

main endp

end main

### 3) ****Assessment → Two inputs & SUBTRACTION****

.model small

.stack 100h

.data

.code

main proc

; first input

mov ah,1

int 21h

sub al,30h ; ASCII → number

mov bl,al

; second input

mov ah,1

int 21h

sub al,30h

mov bh,al

; subtraction = BL - BH

mov al,bl

sub al,bh

add al,30h ; number → ASCII

; newline

mov ah,2

mov dl,0Ah

int 21h

mov ah,2

mov dl,0Dh

int 21h

; display result

mov dl,al

mov ah,2

int 21h

; exit

mov ah,4Ch

int 21h

main endp

end main

⚠️ মনে রাখবেন:

* এখানে **এক অঙ্কের সংখ্যা (0–9)** ইনপুট ধরেছি।
* তাই sub al,30h দিয়ে ASCII → number করেছি, আর আউটপুটের আগে add al,30h করেছি।
* যদি বড় সংখ্যা লাগে, তাহলে multi-digit handle করতে হবে (extra logic দরকার হবে)।

Lab 04:

Objective: Learn and apply the fundamentals of Assembly language

structure.(Conditional Operation)

Tasks:

1.Take a lowercase letter as input from the user and convert it to an uppercase

letter.

2.Practice and understand basic conditional operations.

Assessment: Take an uppercase lowercase letter as input from the user and

convert it to an lowercase letter.

## 🔹 Lab 04

**Objective:** Conditional operation বোঝা এবং ছোট্ট character manipulation করা।

### 1) ****Task 1 → Lowercase → Uppercase****

.model small

.stack 100h

.data

.code

main proc

; input (a–z)

mov ah,1

int 21h

mov bl,al ; save char in BL

; check if lowercase ( 'a' = 97, 'z' = 122 )

cmp bl,'a'

jl not\_lower

cmp bl,'z'

jg not\_lower

; convert lowercase to uppercase

sub bl,20h ; difference between 'a' and 'A'

not\_lower:

; newline

mov ah,2

mov dl,0Ah

int 21h

mov dl,0Dh

int 21h

; display result

mov dl,bl

mov ah,2

int 21h

; exit

mov ah,4Ch

int 21h

main endp

end main

### 2) ****Practice → Conditional Operations****

* cmp reg,value → তুলনা করে (subtraction করে কিন্তু result রাখে না)।
* jl → jump if less
* jg → jump if greater
* je → jump if equal
* এগুলো দিয়েই ছোট decision নেওয়া যায়।

### 3) ****Assessment → Uppercase → Lowercase****

.model small

.stack 100h

.data

.code

main proc

; input (A–Z)

mov ah,1

int 21h

mov bl,al

; check if uppercase ( 'A' = 65, 'Z' = 90 )

cmp bl,'A'

jl not\_upper

cmp bl,'Z'

jg not\_upper

; convert uppercase to lowercase

add bl,20h ; difference between 'A' and 'a'

not\_upper:

; newline

mov ah,2

mov dl,0Ah

int 21h

mov dl,0Dh

int 21h

; display result

mov dl,bl

mov ah,2

int 21h

; exit

mov ah,4Ch

int 21h

main endp

end main

⚠️ নোটঃ

* 'a'–'z' এর ASCII 97–122
* 'A'–'Z' এর ASCII 65–90
* Uppercase ↔ Lowercase রূপান্তর মানে শুধু **+20h বা -20h** করা।

Lab 05:

Objective: Learn and apply the fundamentals of Assembly language

structure.(Conditional Operation)

Tasks:

1.Take a numeric value in the AX register and perform the following:

● If AX is equal to 0, display "Zero"

● If AX is greater than 0, display "Positive"

● If AX is less than 0, display "Negative"

2.Take two inputs from the user and display the smaller value.

Lab Performance: Take three inputs from the user and display the largest

value.

## 🔹 Lab 05

**Objective:** Conditional operations (compare & jump) ব্যবহার করে প্রোগ্রাম বানানো।

### 1) ****Task 1 → AX চেক করা (Zero / Positive / Negative)****

.model small

.stack 100h

.data

msgZero db 'Zero$',0

msgPos db 'Positive$',0

msgNeg db 'Negative$',0

.code

main proc

mov ax,@data

mov ds,ax

; --- AX এ মান সেট করুন (উদাহরণ: 0005h) ---

mov ax,5

; --- Zero check ---

cmp ax,0

je isZero

; --- Positive or Negative ---

jg isPositive

jl isNegative

isZero:

mov dx,offset msgZero

mov ah,09h

int 21h

jmp exit

isPositive:

mov dx,offset msgPos

mov ah,09h

int 21h

jmp exit

isNegative:

mov dx,offset msgNeg

mov ah,09h

int 21h

exit:

mov ah,4Ch

int 21h

main endp

end main

### 2) ****Task 2 → দুই ইনপুট, ছোটটা দেখানো****

.model small

.stack 100h

.data

.code

main proc

; --- first input ---

mov ah,1

int 21h

sub al,30h

mov bl,al

; --- second input ---

mov ah,1

int 21h

sub al,30h

mov bh,al

; compare BL & BH

cmp bl,bh

jl first\_small ; jump if BL < BH

jge second\_small ; jump if BL >= BH

first\_small:

mov dl,bl

add dl,30h

mov ah,2

int 21h

jmp exit

second\_small:

mov dl,bh

add dl,30h

mov ah,2

int 21h

exit:

mov ah,4Ch

int 21h

main endp

end main

### 3) ****Lab Performance → তিন ইনপুট, সবচেয়ে বড়টা দেখানো****

.model small

.stack 100h

.data

.code

main proc

; --- first input ---

mov ah,1

int 21h

sub al,30h

mov bl,al

; --- second input ---

mov ah,1

int 21h

sub al,30h

mov bh,al

; --- third input ---

mov ah,1

int 21h

sub al,30h

mov cl,al

; assume BL is largest

mov al,bl

; compare with BH

cmp bh,al

jg bh\_is\_larger

jmp check\_third

bh\_is\_larger:

mov al,bh

check\_third:

cmp cl,al

jg cl\_is\_larger

jmp show\_result

cl\_is\_larger:

mov al,cl

show\_result:

add al,30h

mov dl,al

mov ah,2

int 21h

; exit

mov ah,4Ch

int 21h

main endp

end main

⚠️ মনে রাখবেন:

* এখানে শুধু **single-digit input (0–9)** ধরা হয়েছে।
* তাই sub al,30h দিয়ে ASCII → number করেছি, আর আউটপুটের আগে add al,30h করেছি।

Lab 06:

Objective: Learn and apply the fundamentals of Assembly language

structure.(Conditional Operation)

Tasks:

1.Take a value from the user and:

● If the value is 1 or 3, display "Odd."

● If the value is 2 or 4, display "Even."

2.Read a character and if it's an uppercase letter display it.

3.Read a character, if it's "y" or "Y", display it; otherwise, terminate the

program.

## 🔹 Lab 06

**Objective:** Conditional operations, multiple condition check, character validation।

### 1) ****Task 1 → Number input (1–4) → Odd / Even****

.model small

.stack 100h

.data

msgOdd db 'Odd$',0

msgEven db 'Even$',0

.code

main proc

mov ax,@data

mov ds,ax

; --- input number ---

mov ah,1

int 21h

sub al,30h ; ASCII → number

mov bl,al

; check 1 or 3 → Odd

cmp bl,1

je displayOdd

cmp bl,3

je displayOdd

; check 2 or 4 → Even

cmp bl,2

je displayEven

cmp bl,4

je displayEven

jmp exit

displayOdd:

mov dx,offset msgOdd

mov ah,09h

int 21h

jmp exit

displayEven:

mov dx,offset msgEven

mov ah,09h

int 21h

; --- exit ---

exit:

mov ah,4Ch

int 21h

main endp

end main

### 2) ****Task 2 → Read char, display if uppercase****

.model small

.stack 100h

.data

.code

main proc

; --- input char ---

mov ah,1

int 21h

mov bl,al

; check if uppercase (A–Z)

cmp bl,'A'

jl exit

cmp bl,'Z'

jg exit

; display

mov dl,bl

mov ah,2

int 21h

exit:

mov ah,4Ch

int 21h

main endp

end main

### 3) ****Task 3 → Read char, if 'y' or 'Y', display****

.model small

.stack 100h

.data

.code

main proc

; --- input char ---

mov ah,1

int 21h

mov bl,al

; check 'y' or 'Y'

cmp bl,'y'

je displayY

cmp bl,'Y'

je displayY

jmp exit ; otherwise terminate

displayY:

mov dl,bl

mov ah,2

int 21h

exit:

mov ah,4Ch

int 21h

main endp

end main

### 🔹 Notes:

* Task 1 → Multiple condition check using cmp + je.
* Task 2 → Uppercase check with ASCII range 'A'-'Z'.
* Task 3 → Single char validation with OR condition ('y' or 'Y').
* সব program-ই DOS interrupt 21h দিয়ে input/output এবং exit handle করছে।

Lab 07:

Objective: Learn and apply the fundamentals of Assembly language

structure.(Loop Operation)

Tasks:

1.Understand the basic loop structure and become familiar with the increment

operator.

2.Write a loop to display a sequence of digits from 0 to 9.

3.Write a loop to display the capital letters from A to Z.

Assessment: Create a loop from 0 to 9 and If the digit is positive, display

"Positive " If the digit is negative, display "Negative"

## 🔹 Lab 07

**Objective:** Loop, increment, conditional display।

### 1) ****Task 1 → Loop 0–9****

.model small

.stack 100h

.data

.code

main proc

mov cx,10 ; loop counter 10 বার

mov al,0 ; start from 0

loop\_digits:

add al,30h ; number → ASCII

mov dl,al

mov ah,2

int 21h

sub al,30h ; revert AL to number for next increment

inc al ; next number

loop loop\_digits

; exit

mov ah,4Ch

int 21h

main endp

end main

### 2) ****Task 2 → Loop A–Z****

.model small

.stack 100h

.data

.code

main proc

mov cx,26 ; letters A–Z

mov al,'A'

loop\_letters:

mov dl,al

mov ah,2

int 21h

inc al ; next letter

loop loop\_letters

; exit

mov ah,4Ch

int 21h

main endp

end main

### 3) ****Assessment → Loop 0–9 + Conditional display Positive / Negative****

.model small

.stack 100h

.data

msgPos db 'Positive$',0

msgNeg db 'Negative$',0

.code

main proc

mov ax,@data

mov ds,ax

mov cx,10 ; loop 10 times

mov al,0 ; start from 0

loop\_check:

cmp al,0

je showPos ; consider 0 as Positive

; If negative → actually none in 0-9, but structure

jl showNeg

showPos:

mov dx,offset msgPos

mov ah,09h

int 21h

jmp next

showNeg:

mov dx,offset msgNeg

mov ah,09h

int 21h

next:

inc al

loop loop\_check

; exit

mov ah,4Ch

int 21h

main endp

end main

### 🔹 Notes:

* Looping: mov cx, n + loop label → CX auto-decrement, zero → exit loop।
* Increment: inc al
* Conditional: cmp + je/jl/jg
* ASCII conversion: digit 0 → '0' = 30h

Lab 08:

Objective: Learn and apply the fundamentals of Assembly language

structure.(Logic, Shift and Rotate Operation)

Tasks:

1.Understand the basic Logic, Shift, and Rotate instructions.

2.Suppose DH contains 8Ah and CL contains 3. Determine the values of DH

and CF after executing the instruction SHR DH, CL.

3.Suppose DH contains 8Ah, CF = 1, and CL contains 3. Determine the

values of DH and CF after executing the instruction RCR DH, CL.

## 🔹 Lab 08

**Objective:** Logic, Shift, Rotate instructions বোঝা।

### 1) ****Task 1 → SHR (Shift Right)****

**Scenario:** DH = 8Ah, CL = 3 → SHR DH, CL

.model small

.stack 100h

.data

.code

main proc

mov dh,8Ah ; DH = 10001010b

mov cl,3 ; CL = 3

shr dh,cl ; Shift DH right by 3

; After shift:

; DH = 00010001b = 11h

; CF = last bit shifted out = bit 2 of original DH = 1

; exit

mov ah,4Ch

int 21h

main endp

end main

**Result:**

* **DH** = 11h
* **CF** = 1

### 2) ****Task 2 → RCR (Rotate through Carry Right)****

**Scenario:** DH = 8Ah, CF = 1, CL = 3 → RCR DH, CL

.model small

.stack 100h

.data

.code

main proc

mov dh,8Ah ; DH = 10001010b

stc ; Set CF = 1

mov cl,3 ; CL = 3

rcr dh,cl ; Rotate DH right through CF by 3

; After rotation:

; DH = 10110001b = B1h

; CF = last bit rotated out = bit 2 of original DH = 1

; exit

mov ah,4Ch

int 21h

main endp

end main

**Result:**

* **DH** = B1h
* **CF** = 1

### 🔹 Notes:

* **SHR** → logical shift right, MSB filled with 0, CF = last bit shifted out.
* **RCR** → rotate right through carry, CF participates in rotation.
* CL register = shift/rotate count.
* stc → set CF = 1

Lab 09:

Objective: Learn and apply the fundamentals of Assembly language

structure.(Stack Operation)

Tasks:

1.

MOV AX, 4561h

MOV BX, FF15h

MOV CX, 1231h

PUSH AX

PUSH CX

POP BX

Show the contents of registers (AX, BX, CX) and the stack:

● Before any PUSH operation

● After both PUSH operations

● After the POP BX operation

2.Reverse a string using the stack.

## 🔹 Lab 09

**Objective:** Stack ব্যবহার করে data save/restore ও string reverse করা।

### 1) ****Task 1 → PUSH/POP Example****

**Scenario:**

MOV AX,4561h

MOV BX,FF15h

MOV CX,1231h

PUSH AX

PUSH CX

POP BX

.model small

.stack 100h

.data

.code

main proc

; --- Initialize registers ---

mov ax,4561h

mov bx,0FF15h

mov cx,1231h

; --- Before PUSH ---

; AX=4561h, BX=FF15h, CX=1231h

; --- PUSH operations ---

push ax

push cx

; Stack now: Top -> CX, AX

; --- POP BX ---

pop bx

; BX = CX (1231h)

; Stack now: Top -> AX

; --- Registers now ---

; AX = 4561h

; BX = 1231h

; CX = 1231h (unchanged)

; exit

mov ah,4Ch

int 21h

main endp

end main

**Summary of stack & registers:**

| **Stage** | **AX** | **BX** | **CX** | **Stack Top → Bottom** |
| --- | --- | --- | --- | --- |
| Before PUSH | 4561h | FF15h | 1231h | empty |
| After 2 PUSH | 4561h | FF15h | 1231h | CX, AX |
| After POP BX | 4561h | 1231h | 1231h | AX |

### 2) ****Task 2 → Reverse a string using stack****

.model small

.stack 100h

.data

str db 'HELLO$',0

.code

main proc

mov ax,@data

mov ds,ax

lea si,str ; point to start of string

push\_loop:

mov al,[si]

cmp al,'$'

je reverse\_start

push ax ; push each char to stack

inc si

jmp push\_loop

reverse\_start:

lea di,str ; point to start again

pop\_loop:

cmp sp,0FFFEh ; stack empty check (stack grows down)

je exit\_stack

pop ax

mov [di],al

inc di

jmp pop\_loop

exit\_stack:

; exit

mov ah,4Ch

int 21h

main endp

end main

**Explanation:**

1. Push each character onto the stack (LIFO)।
2. Pop characters back to the string → reversed।

**Example:**

* Original: HELLO$
* Reversed: OLLEH$

Lab 10:

Objective: Learn and apply the fundamentals of Assembly language

structure.(Multiplication & Division Operation)

Tasks:

1.Perform a basic multiplication operation.

2.Perform a basic division operation.

Assessment: Perform specific multiplication and division operation where

AX contains 2 and BX contains 5.

## 🔹 Lab 10

**Objective:** Multiplication এবং Division operations বোঝা।

### 1) ****Task 1 → Basic Multiplication****

.model small

.stack 100h

.data

.code

main proc

; Initialize registers

mov ax,2 ; AX = 2

mov bx,5 ; BX = 5

; Multiply AX \* BX → result in AX

mul bx ; unsigned multiply: AX = AX \* BX

; After multiplication:

; AX = 0Ah (2 \* 5 = 10 decimal)

; exit

mov ah,4Ch

int 21h

main endp

end main

### 2) ****Task 2 → Basic Division****

.model small

.stack 100h

.data

.code

main proc

; Initialize registers

mov ax,10 ; AX = dividend

mov bx,5 ; BX = divisor

; Divide AX / BX → quotient in AL, remainder in AH (for 8-bit division)

; or AX / BX → quotient in AX, remainder in DX (for 16-bit division)

xor dx,dx ; clear DX before 16-bit division

div bx ; AX / BX → AX = quotient, DX = remainder

; After division:

; AX = 2 (quotient), DX = 0 (remainder)

; exit

mov ah,4Ch

int 21h

main endp

end main

### 🔹 Notes:

* mul reg → unsigned multiply, AX \* reg → AX (or DX:AX for 16-bit reg).
* div reg → unsigned divide, AX / reg → AX = quotient, DX = remainder.
* xor dx,dx → division এর আগে remainder clear করা প্রয়োজন।
* এই উদাহরণে:
  + Multiplication → 2 \* 5 = 10 → AX = 0Ah
  + Division → 10 / 5 → AX = 2 (quotient), DX = 0 (remainder)