**1.Identification and Explain various blocks in 8086 microprocessor architecture**

**ANS:=1. Bus Interface Unit (BIU)**

The **BIU** handles all data and address transfers between the microprocessor and memory or I/O devices. It is responsible for fetching instructions, reading/writing data, and managing the address bus.

**🔹 Components of BIU:**

* **Instruction Queue (6 bytes):**
  + Prefetches instructions from memory to speed up execution.
  + Works like a pipeline to improve efficiency (fetch while execute).
* **Segment Registers (CS, DS, SS, ES):**
  + Used to access different segments of memory.
  + Each is 16-bit, and when combined with an offset, gives a 20-bit physical address.
    - **CS**: Code Segment – holds the base address of the code.
    - **DS**: Data Segment – holds data variables.
    - **SS**: Stack Segment – used during stack operations.
    - **ES**: Extra Segment – used for additional data storage.
* **Instruction Pointer (IP):**
  + Holds the offset of the next instruction within the code segment.

**🔷 2. Execution Unit (EU)**

The **EU** is responsible for decoding and executing instructions. It takes instructions from the BIU’s instruction queue and processes them.

**🔹 Components of EU:**

* **ALU (Arithmetic Logic Unit):**
  + Performs arithmetic (add, subtract, etc.) and logic (AND, OR, NOT, etc.) operations.
* **General Purpose Registers:**
  + **AX** (Accumulator), **BX**, **CX**, **DX** – each 16-bit, can be used as 8-bit registers too (AH, AL).
* **Pointer and Index Registers:**
  + **SP (Stack Pointer)**, **BP (Base Pointer)**, **SI (Source Index)**, **DI (Destination Index)** – used in addressing and stack operations.
* **Flag Register:**
  + Contains status flags (Zero, Carry, Sign, Overflow, etc.) that reflect results of operations and control execution.
* **Instruction Decoder:**
  + Decodes the instruction fetched into signals the control unit and ALU can act upon.
* **Control Unit:**
  + Sends control signals to various parts of the processor to carry out instructions

.

**2.Use assembly language programming (ALP) tools and directives.**

**ANS:=** Tools Used in ALP (for 8086):

1. Assembler (e.g., MASM, TASM):
   * Converts assembly code into machine code (object file).
2. Linker:
   * Combines object files into a single executable.
3. Debugger (e.g., DEBUG in DOS):
   * Used to test and debug assembly programs.
4. Editor:
   * Used to write the assembly code (Notepad, DOSBox editor, etc.).
5. Emulator/Simulator:
   * Simulates 8086 processor environment for testing code.

Common Directives in 8086 Assembly:

| Directive |  |
| --- | --- |
| .MODEL |  |
| .DATA |  |
| .CODE |  |
| .STACK |  |
| DB |  |
| DW |  |
| END |  |
| PROC/ENDP |  |
| ASSUME |  |

**3. ALP to perform addition and subtraction of two given numbers.**

**ANS:=** .MODEL SMALL

.STACK 100H

.DATA

NUM1 DB 0AH

NUM2 DB 05H

SUM DB ?

DIFF DB ?

.CODE

MAIN PROC

MOV AX, @DATA ; Initialize data segment

MOV DS, AX

; ---- Addition ----

MOV AL, NUM1

ADD AL, NUM2

MOV SUM, AL

; ---- Subtraction ----

MOV AL, NUM1

SUB AL, NUM2

MOV DIFF, AL

MOV AH, 4CH

INT 21H

MAIN ENDP

END MAIN

**4. ALP for multiplication of two signed numbers**

**ANS:=** DATA SEGMENT

NUM1 DB -05H

NUM2 DB -04D RESULT DW ?

DATA ENDS

CODE SEGMENT

ASSUME DS: DATA, CS: CODE

START: MOV AX,DATA

MOV DS, AX

MOV AH, 00H

MOV AL, NUM1

MOV BL, NUM2

IMUL BL

MOV RESULT, AX

MOV AH, 09H

INT 21H CODE

ENDS END

START

**5:** **ALP for multiplication of two unsigned numbers.**

**ANS:=** DATA SEGMENT

NUM1 DB 05H

NUM2 DB 04H RESULT DW ?

DATA ENDS

CODE SEGMENT

ASSUME DS: DATA, CS: CODE

START: MOV AX, DATA MOV DS,AX

MOV AL,NUM1

MOV BL,NUM2

MUL BL

MOV RESULT,AX

MOV AH, 09H

INT 21H CODE

ENDS END

START

**6: ALP to perform division of two signed numbers.**

**ANS:-** DATA SEGMENT

NUM DW -0123H

NUM1 DB -12H

Q\_RES DB ? R\_RES DB ?

DATA ENDS

CODE SEGMENT

ASSUME DS: DATA,CS: CODE

START:MOV AX,DATA MOV DS,AX

MOV AX,NUM

MOV BL,NUM1

IDIV BL

MOV Q\_RES, AL MOV R\_RES,AH

MOV AH, 09H

INT 21H

CODE ENDS

END START

**7: ALP to perform division of two signed numbers.**

**ANS:**- DATA SEGMENT

NUM DW 0000H

NUM1 DW 0123H

NUM2 DW 0012H RES DW ?

RES1 DW ? DATA ENDS

CODE SEGMENT

ASSUME DS: DATA,CS: CODE

STAR:

MOV AX,DATA MOV DS, AX

MOV DX, NUM

MOV AX, NUM1

MOV BX, NUM2

DIV BX

MOV RES, AX ;quetient

MOV RES1, DX ;remainder

MOV AH, 09H

INT 21H

CODE ENDS END START

**8:** **ALP to perform addition and subtraction of two BCD numbers.**

**ANS:-** data segment

num1 db 25h

num2 db 17h

sum db ?

diff db ?

data ends

code segment

Assume cs: Code, ds: data

mov ax, data

mov ds, ax

mov al, num1

add al, num2

daa mov sum, al

mov al, num1

sub al, num2

daa mov diff, al

mov ah, 4ch

int 21h

Code ends

end start

**9:** **ALP to find sum of series**

**ANS:** .model small

.stack 100h

.data

N dw 10

SUM dw 0

.code

main:

mov ax, @data

mov ds, ax

mov cx, N

mov ax, 0

mov bx, 1

sum\_loop:

add ax, bx

inc bx

loop sum\_loop

mov SUM, ax

; Exit program

mov ah, 4ch

int 21h

end main

**10:**

**ALP to find smallest number from array of numbers**

**ANS:-**

DATA SEGMENT

ARRAY DW 12H,31H,02H,45H,65H

SMALL DW 0 DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:MOV AX,DATA MOV DS,AX

MOV CX,05H

MOV SI,OFFSET ARRAY

MOV AX,[SI]

DEC CX

UP: INC SI

INC SI

CMP AX,[SI]

JC NEXT

MOV AX,[SI]

NEXT:LOOP UP

MOV SMALL,AX

MOV AH, 09H

INT 21H

CODE ENDS

END START

**11: ALP to find largest number from array of numbers**

**ANS:-** DATA SEGMENT

ARRAY DB 12H,31H,02H,45H,65H

LARGE DB 0 DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:MOV AX,DATA

MOV DS,AX

MOV CX,05H

MOV SI,OFFSET ARRAY

MOV AL,[SI]

DEC CX

UP: INC SI

CMP AL,[SI]

JNC NEXT

MOV AL,[SI]

NEXT:LOOP UP

MOV LARGE,AL

MOV AH, 09H

INT 21H

CODE ENDS

END START

**12: ALP to arrange numbers in an array in ascending order.**

**ANS:-** DATA SEGMENT

ARRAY DB 12H,11H,21H,9H,19H

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:MOV AX,DATA MOV DS,AX

MOV BX,05H

UP1:MOV SI,OFFSET ARRAY

MOV CX,04H

UP:MOV AL,[SI] CMP AL,[SI+1]

JC DN

XCHG

AL,[SI+1]

XCHG AL,[SI]

DN:INC SI

LOOP UP

DEC BX

JNZ UP1

MOV AH, 09H

INT 21H

CODE ENDS

END START

**13: ALP to find the length of string.**

**ANS:-** .model small

.stack 100h

.data

str db 'Hello, World!$'

len db 0

.code

main:

mov ax, @data

mov ds, ax

lea si, str

mov cx, 0

find\_len:

mov al, [si]

cmp al, '$'

je done\_len

inc si

inc cx

jmp find\_len

done\_len:

mov len, cl

; Exit program

mov ah, 4ch

int 21h

end main

**14: ALP to arrange numbers in an array in descending order.**

**ANS:** DATA SEGMENT

ARRAY DB 12H,11H,21H,9H,19H

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:MOV AX,DATA

MOV DS,AX

MOV BX,05H

UP1:MOV SI,OFFSET ARRAY MOV CX,04H

UP:MOV AL,[SI] CMP AL,[SI+1]

JNC DN

XCHG

AL,[SI+1]

XCHG AL,[SI]

DN:INC SI

LOOP UP

DEC BX

JNZ UP1

MOV AH, 09H

INT 21H

CODE ENDS

END START

**15: ALP to perform concatenation of two strings.**

**ANS:-** DATA SEGMENT

STR1 DB 'ABCD$'

STR2 DB 'PQRS$'

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START: MOV DX,DATA

MOV DS, DX LEA SI,STR1

LEA

DI,STR2

MOV AL,'$'

UP: CMP AL,[SI]

JZ NEXT

INC SI

JMP UP

NEXT: CMP AL,[DI] JZ EXIT

MOV BL,[DI]

MOV [SI], BL

INC SI

INC DI

JMP

NEXT

EXIT: MOV [DI],AL

MOV AH,4CH INT 21H

CODE ENDS

END START

**16: ALP for string operations such as string reverse**

**ANS:-** DATA SEGMENT

STR1 DB 'ABCDEF$'

STR2 DB 10 DUP('$')

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START: MOV DX,DATA MOV DS, DX

LEA SI,STR1

MOV CL,06H LEA DI,STR2

ADD DI,05H MOV AL,'$'

UP: MOV AL,[SI]

MOV [DI],AL

DEC DI

INC SI

LOOP UP

MOV AH,4CH INT 21H

CODE ENDS

END START

**17:** **ALP for string operations such as string copy**

**ANS**: .model small

.stack 100h

.data

src db 'Hello, 8086!$',

dest db 20 dup('$')

.code

main:

mov ax, @data

mov ds, ax

mov es, ax

lea si, src

lea di, dest

copy\_loop:

lodsb

stosb

cmp al, '$'

**18: ALP to check a given number is odd or even.**

**ANS:** DATA SEGMENT

NO1 DB 13H

MS1 DB 10,13, "NUMBER IS ODD$"

MS2 DB 10,13, "NUMBER IS EVEN$" DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START: MOV DX,DATA

MOV DS, DX

MOV

AL,NO1

MOV AH,00H MOV BH,01H MOV BL,02H

DIV BL

CMP AH,BH

JE NEXT

LEA DX,MS2

MOV AH,09H

INT 21H

NEXT:LEA DX,MS1

MOV AH,09H

INT 21H

MOV AH,4CH

INT 21H

CODE ENDS

END START

**19: ALP to count number of' ‘0' and '1's in a given number.**

**Ans:** DATA SEGMENT NO1 DB 13H ZERO DB ? ONE DB ?

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START: MOV DX,DATA

MOV DS, DX

MOV AL,NO1

MOV BH,00H

MOV BL,00H

MOV CL,08H

UP:RCL AL,01H

JC NEXT

INC BL

LOOP UP

NEXT: INC BH

LOOP UP

EXIT:MOV ZERO, BL

MOV ONE, BH

MOV AH,4CH

INT 21H

CODE ENDS

END START

**20:** **ALP to perform arithmetic operations on given numbers using procedure.**

**Ans:**