Name - Sadab Hussain
Enrollment No- 9220170310 7

Sub-Fundamental of Microprocessor

8086 Program Assignment

1.) Write an Assembly language program to add two 32 bit number where 1st number is stored at 1000H to 1003H and second number is stored at 1010H to 1013H memory location result should be store at 1020H location onward (LSB will at lower address location).

soln. PA1.asm

```
.MODEL SMALL
.STACK 100H
.CODE
START:
        MOV SI, 1000H     ; SI points to first Number
        MOV DI, 1010H     ; DI points to second Number
        MOV BX, 1020H     ; BX points to result location
        CLC               ; clear carry flag
        MOV CX, 4         ; set loop counter for 4 bytes (32-bits)
ADD_LOOP:
        MOV AL, [SI]      ; Load byte from First Number
        ADD AL, [DI]      ; Add byte from second Number
        MOV [BX], AL      ; store result byte
        INC SI            ; Increment to next byte
        INC DI
        INC BX
        ADC AL, 0         ; Handle carry
        LOOP ADD_LOOP     ; Repeat for next byte.
        MOV AH, 4CH
        INT 21H
END START
```

2.) Write Assembly language program to compute multiplication using repetitive addition.

Soln. PAC2.asm

```
.MODEL SMALL
.STACK 100H
.DATA
    NUM1 DB  5    ; Multiplicand
    NUM2 DB  4    ; Multiplier
    RESULT DW 0   ; Result
.CODE
START :
    MOV AX,@DATA
    MOV DS, AX
    MOV AL, NUM1  ; Load Multiplicand
    MOV BL, NUM2  ; Load Multiplier
    XOR BX, BX    ; Clear BX (result)
MULTIPLY:
    ADD BX, AX    ; Add multiplicand to BX
    DEC BL        ; Decrement multiplier
    JNZ MULTIPLY  ; Repeat until BL = 0.
    MOV RESULT, BX ; Store Result
; exit program
    MOV AH, 4CH
    INT 21H
END START
```

3.) Write Assembly language program to count number of 1's in the given 16-bit word.

Soln:) PAC3.asm

```
.MODEL SMALL
.STACK 100H
.DATA
    NUM DW 0F5A3H
    COUNT DB 0
.CODE
START:
```

```
MOV AX, NUM      ; Load 16-bit word
MOV CX, 16       ; set loop for 16-bits
XOR BL,BL        ; clear count (BL)
COUNT_ONES:
SHR AX, 1        ; shift right, LSB into carry
JNC SKIP         ; IF no carry (0), skip
INC BL           ; increment count for 1
SKIP:
LOOP COUNT_ONES  ; Reapeat 16 times
MOV COUNT, BL    ; store count
MOV AH, 4CH      ; exit
INT 21H
ENDSTART
```

4.) Write Assembly language program to seprate 2 nibble of the 8-bit words.

soln.) PAC4.asm

```
.MODEL SMALL
.STACK 100H
.DATA
    BYTE_VAL DB 5AH      ; 8-bit word
    UPPER_NIBBLE DB 0    ; To store upper nibble
    LOWER_NIBBLE DB 0    ; To store lower nibble
.CODE
START:
    MOV AL, BYTE_VAL     ; Load the 8-bit word into AL
    MOV AH, AL           ; copy AL to AH for upper nibble
    SHR AH, 4            ; shift right 4bits to get upper nibble
    MOV UPPER_NIBBLE, AH ; store upper nibble
    AND AL, 0FH          ; Mask upper nibble ; keep lower nibble
    MOV LOWER_NIBBLE, AL ; store lower nibble
    MOV AH, 4CH
    INT 21H
END START
```

5.) Write a Assembly language program to division by repetitive subtraction.

Soln. PACS.asm

```
.MODEL SMALL
.STACK 100H
.DATA
    DIVIDEND DW 20      ; Dividend (value to be divided)
    DIVISOR  DW 6       ; Divisor (value to divide by)
    QUOTIENT DB 0       ; To store quotient
    REMAINDER DW 0      ; To store remainder
.CODE
START:
    MOV AX, @DATA       ; Initialize data segment
    MOV DS, AX
    MOV AX, DIVIDEND    ; Load dividend into AX
    MOV BX, DIVISOR     ; Load divisor into BX
    XOR CX, CX          ; Clear CX (to count quotient)
DIV_LOOP:
    CMP AX, BX          ; Compare dividend with divisor
    JL DONE             ; If dividend < divisor, finish
    SUB AX, BX          ; Subtract divisor from dividend
    INC CX              ; Repeat until dividend < divisor
    JMP DIV_LOOP
DONE:
    MOV QUOTIENT, CL    ; Store quotient
    MOV REMAINDER, AX   ; Store remainder (left in AX)
    MOV AH, 4CH
    INT 21H
END START
```

6.) Write a program to Read an array of 10 elements and add 5 into each number and store them in the same location.

Soln.

```
.MODEL SMALL
.STACK 100H
.DATA
    ARRAY DB 10 DUP(1)      ; Array of 10 elements initialized with 1
.CODE
START:
    MOV AX,@DATA            ; Initilize data segment
    MOV DS, AX
    MOV SI, 0               ; Index for Array
    MOV CX, 10              ; Loop For 10 elements
AND_LOOP:
    ADD ARRAY[SI],5 ; Add 5 to each elements
    INC SI                  ; MOV to next element
    LOOP ADD_LOOP           ; Repeat 10 times
    MOV AH, 4CH             ; Exit.
    INT 21H
END START
```

7) Assembly language program to count number of positive and negative numbers from an array of 10 numbers starts form 100H location.

Soln.

```
.MODEL SMALL
.STACK 100H
.DATA
    POS_COUNT DB 0 ; positive count
    NEG_COUNT DB 0 ; Negative count
.CODE
START:
    MOV SI, 100H    ; start address of array
    MOV CX, 10      ; Loop for 10 elements
CHECK_NUM:
    MOV AL,[SI]     ; Load number into AL
```

```
    CMP AL, 0
    JL NEGATIVE      ; IF AL<0, JUMP to Negative
    INC POS_COUNT    ; Increment positive count
    JMP NEXT
NEGATIVE:
    INC NEG_COUNT    ; Increment negative count
NEXT:
    INC SI           ; Move to next element
    LOOP CHECK_NUM   ; Repeat for all 10 elements
    MOV AH, 4CH      ; exit
    INT 21H
END START
```

8) Assembly language program to count number of odd and even numbers.

Soln.

```
.MODEL SMALL
.STACK 100H
.DATA
    ODD_COUNT DB 0       ; odd count
    EVEN_COUNT DB 0      ; Even count
    ARRAY DB 10 DUP(1, 2, 3, 4, 5, 6, 7, 8, 9)
.CODE
START:
    MOV SI, OFFSET ARRAY ; Array start
    MOV CX, 10           ; Loop for 10 elements
CHECK_NUM:
    MOV AL, [SI]         ; Load number
    AND AL, 1            ; Check LSB
    JZ EVEN              ; IF 0, its even
    INC ODD_COUNT        ; Increment odd count
    JMP NEXT
EVEN:
```

vision

```
    INC EVEN_COUNT      ; Increment even count
NEXT:
    INC SI              ; Next element
    LOOP CHECK_NUM      ; Repeat for 10 elements
    MOV AH, 4CH         ; exit
    INT 21H
END START
```

9) Assembly language program to compare two strings.

Soln

```
PRG9.ASM
.MODEL SMALL
.STACK 100H
.DATA
    STR1 DB 'HELLO', 0      ; First string
    STR2 DB 'HELLO', 0      ; Second string
    RESULT DB 0             ; 0 if equal, 1 if not
.CODE
START:
    MOV SI, OFFSET STR1     ; Load STR1
    MOV DI, OFFSET STR2     ; Load STR2
CMP_LOOP:
    MOV AL, [SI]            ; Load char from STR1
    MOV BL, [DI]            ; Load char from STR2
    CMP AL, BL              ; Compare chars
    JNE NOT_EQUAL           ; If not equal, jump
    CMP AL, 0               ; Check for end (null terminator)
    JE EQUAL                ; If end, strings are equal
    INC SI                  ; Move to next char
    INC DI
    JMP CMP_LOOP            ; Repeat
NOT_EQUAL:
    MOV RESULT, 1           ; String not equal
```

vision

```
EQUAL:
    MOV AH, 4CH     ;exit
    INT 21H
END START
```

10) Write a program to check number if prime or not.

soln. PAC10.asm

```
.MODEL SMALL
.STACK 100H
.DATA
    NUMBER DW 11.   ;Number to check
    RESULT DB 0     ; 0=Prime, 1=Not prime
.CODE
START:
    MOV AX, NUMBER  ; load number
    MOV CX, AX      ; copy number
    MOV BX, 2       ; Divisor starts from 2
CHECK_LOOP:
    CMP BX, CX      ; If divisor >= Number, it's prime
    JAE IS_PRIME    ; If divisor >= number ; it's prime
    MOV DX, 0       ; Clear DX for division
    DIV BX          ; Divide Ax by Bx
    CMP DX, 0       ; check remainder
    JE NOT_PRIME    ; If remainder is 0, it's not prime
    INC BX          ; Increment divisor
    MOV AX, CX      ; Restore number
    JMP CHECK_LOOP
NOT_PRIME:
    MOV RESULT 1    ; Not prime
IS_PRIME:
    MOV AH, 4CH     ; exit program
    INT 21H
END START
```

11.) Assembly language program to find the sum of 1 to 15 number.

Soln PAC11.asm

```
.MODEL SMALL
.STACK 100H
.DATA
    SUM DB 0          ; variable for the sum
.CODE
START:
    MOV CX, 15        ; set counter to 15
    XOR AL, AL        ; clear AL for SUM
SUM_LOOP:
    ADD AL, CL        ; Add current number (CL) to AL
    DEC CX            ; Decrement counter
    JNZ SUM_LOOP      ; Repeat until CX ≠ 0
    MOV SUM, AL       ; store result in SUM
    MOV AH, 4CH       ; exit program
    INT 21H           ; Terminate program
END START
```

12.) Assembly Language program to compute sum of five 8 bit number stored in memory.

Soln. PAC12.asm

```
.MODEL SMALL
.STACK 100H
.DATA
    NUMBERS DB 10, 20, 30, 40, 50 ; Array of 5 numbers
    SUM DB 0                      ; variable for the sum
.CODE
.START:
    MOV AX, @DATA                 ; Initialize data segment
    MOV DS, AX
```

VISION

```
        MOV CX,5          ; Set counter for 5 numbers
        XOR AL,AL         ; Clear AL for sum
SUM_LOOP:
        ADD AL,[NUMBERS]  ; Add current number
        INC NUMBERS       ; Move to next number
        LOOP SUM_LOOP     ; Repeat
        MOV SUM,AL        ; Store result in SUM
        MOV AH,4CH        ; Exit program
        INT 21H           ; Return to DOS
END START
```

13) Assembly language program to find maximum from 10 numbers stored in memory.

Soln, MAX13.ASM

```
.MODEL SMALL
.STACK 100H
.DATA
    NUMBERS DB 10,20,35,40,15,30,25,60,35,50 ; Array of 10 numbers
    MAX DB 0                                 ; Variable for the maximum
.CODE
START:
    MOV AX, @DATA          ; Initialize data segment
    MOV DS, AX             ; Set counter for 10 numbers
    MOV CX, 10             ; Set counter for the numbers.
    MOV AL,[NUMBERS]       ; Assume first number is max
    MOV SI, OFFSET NUMBERS ; Load address of Numbers
MAX_LOOP:
    INC SI                 ; Move to the next number
    DEC CX                 ; Decrement counter
    CMP CX, 0              ; Check if the counter is zero
    JZ STORE_MAX           ; If CX is zero, go to store the max
```

vision

```
    CMP AL, [SI]        ; compare current max (AL) with the next number
    JLE CONTINUE        ; IF AL >= NUMBERS, continue
    MOV AL, [SI]        ; update max. if current > max
CONTINUE:
    JMP MAX_LOOP        ; Repeat for all numbers
STORE_MAX:
    MOV MAX, AL         ; store max in MAX
    MOV AH, 4CH         ; Terminate program.
    INT 21H
END START
```

14) Write a program that compute Factorial of n (n<=7).

Soln: PAC-14.asm

```
.MODEL SMALL
.STACK 100H
.DATA
    n DB 5              ; Input number (0 <= n <= 7)
    Factorial DB 0      ; Variable for the factorial result
.CODE
START:
    MOV AX, @DATA       ; Initialize data segment
    MOV DS, AX          ;
    MOV AL, n           ; Load n into AL
    MOV BL, AL          ; Copy n to BL (counter)
    MOV AL, 1           ; Initialize AL to 1 (Factorial starts from 1)
    CMP BL, 0           ; check if n is 0
    JE FACTORIAL_ZERO   ; If n is 0, jump to FACTORIAL_ZERO
FACTORIAL_LOOP:
    MUL BL              ; AL = AL * BL (update factorial)
    DEC BL              ; decrement BL
    JNZ FACTORIAL_LOOP  ; Repeat until BL is 0
    JMP END             ; Jump to end
```

vision

```
FACTORIAL_ZERO:
    MOV AL, 1          ; 0! = 1
END:
    MOV Factorial, AL  ; store the result in Factorial
    MOV AX, 4C00H      ; terminate program.
    INT 21H
ENDSTART
```

15) Write a program to compute length of the string.

Soln) PAC_15.asm

```
.MODEL SMALL
.STACK 100H
.DATA
    String DB 'Hello, world!', 0   ; Null-terminated string
    length DB 0                    ; variable to store the length
.CODE
START:
    MOV AX, @DATA          ; Initialize data segment
    MOV DS, AX
    MOV SI, OFFSET string  ; Load address of the string
    MOV CX, 0              ; Initialize counter
COUNT_LOOP:
    MOV AL, [SI]           ; Load current character
    CMP AL, 0              ; Check for null terminator
    JE STORE_LENGTH        ; Jump if null terminator found
    INC CX                 ; Increment counter
    INC SI                 ; Move to next character
    JMP COUNT_LOOP         ; Repeat loop
STORE_LENGTH:
    MOV length, CL         ; Store the length
    MOV AX, 4C00H          ; Terminate program
    INT 21H
END START
```

16) Check two number are "equal or not" using logical operation.

Soln) PAC_16.asm

```
.MODEL SMALL
.STACK 100H
.DATA
    num1 DB 5          ; First number
    num2 DB 5          ; second number
    result DB 0        ; variable to store the result (1 for equal, 0 for not equal)
.CODE
START:
    MOV AX, @DATA      ; Initialize data segment
    MOV DS, AX
    MOV AL, num1       ; Load first number into AL
    CMP AL, num2       ; Compare AL with the second number
    JE NUM_EQUAL       ; Jump if equal
    ; If not equal
    MOV result, 0      ; Set result to 0 (not equal)
    JMP END            ; Jump to end
NUM_EQUAL:
    MOV result, 1      ; Set result to 1 (equal)
END:
    MOV AX, 4C00H      ; Terminate program.
    INT 21H
END START
```

17) Write a assembly language program using subroutine to sort 10 number using Selection sort.

Soln) PAC17.asm

```
.MODEL SMALL
.STACK 100H
.DATA
```

```
numbers DB 64,34,25,12,22,90,45,78,56 ; Array of 10 numbers
length DB 10
.CODE
START:
    MOV AX, @DATA
    MOV DS, AX
    CALL SELECTION_SORT
    MOV AX, 4C00H
    INT 21H
SELECTION_SORT:
    MOV CX, length          ; Outer loop counter
    DEC CX                  ; Start from last index
OUTER_LOOP:
    MOV SI, 0               ; Inner loop index
    MOV DI, CX              ; Assume the first unsorted element is the main
INNER_LOOP:
    INC SI                  ; Check next element
    CMP SI, length          ; End of array?
    JAE SWAP                ; If Yes, swap the found main
    MOV AL, [numbers+SI]    ; Load current element
    CMP AL, [numbers+DI]    ; Compare with current min
    JAE INNER_LOOP          ; Continue if not smaller
    MOV DI, SI              ; Update min index
    JMP INNER_LOOP
SWAP:
    MOV AL, [numbers+CX]    ; Swap elements
    MOV BL, [numbers+DI]
    MOV [numbers+CX], BL
    MOV [numbers+DI], BL
    LOOP OUTER_LOOP         ; Repeat Outer loop
    RET
END START
```