# Microprocessor and Assembly Language Programming

*Pokhara University*

## Section 1: 8085 Microprocessor ALP

Q.No.1: Write an ALP for 8085 to count the integers available in an array starting from memory location 2500H to 2510H for exactly divisible by two and save result in register B. [2013 Fall]

```
LXI H, 2500H        ; Load HL with the starting address of the array (2500H)
MVI C, 11H          ; Load the count of numbers (11 elements, from 2500H to 2510H)
into C
MVI B, 00H          ; Initialize counter (B) to 0

NEXT_ELEMENT:
MOV A, M            ; Load the current number from memory into the accumulator
ANI 01H             ; Check if the number is divisible by 2 (AND with 01H)
JNZ SKIP_INCREMENT  ; If the result is not 0 (odd number), skip incrementing the
counter

INR B               ; Increment the counter for even numbers

SKIP_INCREMENT:
INX H               ; Increment HL to point to the next memory location
DCR C               ; Decrement the counter
JNZ NEXT_ELEMENT    ; If more numbers remain, repeat the loop

HLT                 ; Halt the program
```

Q.No.2: Write an 8085 program to find odd and even number among six bytes of data: 23H, 41H, 56H, AFH, C5H and 47H stored in memory location starting from 2500H and place the odd and even numbers un the memory location starting from 2600H and 2400H respectively.

[2013 Spring]

```
LXI H, 2500H        ; Load HL with the starting address of the data (2500H)
LXI D, 2400H        ; Load DE with the starting address for even numbers (2400H)
LXI B, 2600H        ; Load BC with the starting address for odd numbers (2600H)
MVI C, 06H          ; Load the counter (C) with the number of bytes (6)

NEXT_NUMBER: MOV A, M          ; Load the current number from memory into accumulator
ANI 01H             ; Check if the number is even (AND with 01H)
JZ STORE_EVEN       ; If zero (even), jump to STORE_EVEN

STORE_ODD:
MOV A, M            ; Load the number again into the accumulator
STAX B              ; Store the odd number at the address pointed by BC
INX B               ; Increment BC to point to the next odd memory location
JMP NEXT_STEP       ; Jump to NEXT_STEP

STORE_EVEN:
MOV A, M            ; Load the number again into the accumulator
STAX D              ; Store the even number at the address pointed by DE
INX D               ; Increment DE to point to the next even memory location
```

```
NEXT_STEP:
INX H                ; Increment HL to point to the next number in the source array
DCR C                ; Decrement the counter
JNZ NEXT_NUMBER      ; If more numbers remain, repeat the loop
HLT                  ; Halt the program
```

Q.No.3: Write an assembly language program for intel 8085 to compute

$$\sum_{i=1}^{n} \frac{x_i}{2}$$

Where $X_i$ are three numbers stored at memory locations A001H,A002H and A003H. store the result at memory location starting at D000H. [2014 Fall]

```
LDA A001H         ; Load the first number (X1) into accumulator
RRC               ; Divide X1 by 2 (right shift)
MOV B, A          ; Store the result in register B

LDA A002H         ; Load the second number (X2) into accumulator
RRC               ; Divide X2 by 2
ADD B             ; Add the halved X2 to the result in register B
MOV B, A          ; Store the sum back in register B

LDA A003H         ; Load the third number (X3) into accumulator
RRC               ; Divide X3 by 2
ADD B             ; Add the halved X3 to the result in register B

STA D000H         ; Store the final result in memory location D000H
HLT               ; Halt the program
```

Q.No.4: Write a 8085 program to sort given 10 numbers from memory location 2200H in the ascending order.

[2014 Spring]

```
MVI B, 09       ; Initialize counter
START           ; LXI H, 2200H: Initialize memory pointer
MVI C, 09H      ; Initialize counter 2
BACK: MOV A, M  ; Get the number
INX H           ; Increment memory pointer
CMP M           ;Compare number with next number
JC SKIP         ;If less, don’t interchange
JZ SKIP         ;If equal, don’t interchange
MOV D, M
MOV M, A
DCX H
MOV M, D
INX H           ;Interchange two numbers
SKIP;DCR C      ;Decrement counter 2
JNZ BACK        ;If not zero, repeat
DCR B           ;Decrement counter 1
```

```
JNZ START

HLT             ;Terminate program execution
```

Q.No.5: Write a 8085 ALP to find out the largest number in an array available memory location starting from 2500H to 2510H and store the result in D000H. [2015 Fall]

```
LXI H, 2500H        ; Load HL with the starting address of the array
MOV A, M            ; Load the first number into the accumulator
INX H               ; Increment HL to point to the next number

MVI C, 10H          ; Load the counter with the number of elements (16 in decimal)

NEXT_NUMBER:
CMP M               ; Compare the accumulator with the current number
JC SKIP_UPDATE      ; If the current number is smaller, skip update
MOV A, M            ; Otherwise, update the accumulator with the current number

SKIP_UPDATE:
INX H               ; Increment HL to point to the next number
DCR C               ; Decrement the counter
JNZ NEXT_NUMBER     ; If counter is not zero, repeat

STA D000H           ; Store the largest number in memory location D000H
HLT                 ; Halt the program
```

Q.No.6: Write an 8085 program for the following type of addition $1^2 + 2^2 + 3^2 + \cdots + 9^2$ .

[2015 Spring , 2019 Fall]

```
MVI D, 00H            ; Initialize sum to 0 (D register holds the sum)
MVI E, 01H            ; Initialize counter to 1 (E register holds the counter)

LOOP:
MOV A, E              ; Move the counter value to the accumulator (A)
CALL SQUARE           ; Call subroutine to calculate A^2 (square of E)

ADD D                 ; Add the square to the sum (D holds the current sum)
MOV D, A              ; Store the updated sum back in the D register

INX E                 ; Increment counter E (to move to next number)
MVI A, 0x0A           ; Compare if counter reaches 10 (End of loop)
CMP A
JZ DONE               ; If counter reaches 10, stop the loop

JMP LOOP              ; Repeat loop if counter is less than 10

DONE:
MVI H, 0D000H         ; Load HL with memory location D000H to store the result
MOV M, D              ; Store the final sum at D000H
```

```
HLT                     ; Halt the program

SQUARE:
MOV B, A               ; Copy the value of A (current number) into B
CALL MULTIPLY          ; Call subroutine to calculate A^2

RET

MULTIPLY:
MVI C, 00H             ; Clear register C (used for storing result)
MOV D, A               ; Copy value of A into D (first operand)
MOV B, A               ; Copy value of A into B (second operand)

MOV A, D               ; Load first operand (A) into A
CALL ADD               ; Add these two numbers to get the square

MOV A, C               ; Retrieve result into accumulator
RET
```

Q.No.7: Write a 8085 program to find smallest of three numbers. [2016 Fall]

```
LXI H, 2500H           ; Load HL with address of the first byte (2500H)
MOV A, M               ; Load the low byte of the number into A (A = low byte)
MOV B, A               ; Copy the low byte into B for comparison

INX H                  ; Increment HL to point to the second number (2501H)
MOV A, M               ; Load the second number into the accumulator (A = second
byte)
CALL COMPARE           ; Compare A with the number in B (current smallest)

INX H                  ; Increment HL to point to the third number (2502H)
MOV A, M               ; Load the third number into the accumulator (A = third byte)
CALL COMPARE           ; Compare A with the current smallest (in B)

LXI H, 3000H           ; Load HL with the address to store the smallest number
(3000H)
MOV M, B               ; Store the smallest number in memory location 3000H

HLT                     ; Halt the program

COMPARE:
CMP B                  ; Compare A with B (current smallest)
JC SKIP                ; If A >= B, skip the update
MOV B, A               ; If A < B, update B with the new smallest value
SKIP:
RET                    ; Return from the subroutine
```

Q.No.8: Write an ALP in 8085 to find whether the given number us palindrome or not.

[2016 Fall]

```
LXI H, 2500H           ; Load HL with address of the first byte (2500H)
MOV A, M               ; Load the low byte of the number into A (A = low byte)
MOV B, A                ; Copy the low byte into B for comparison

INX H                   ; Increment HL to point to the second byte (2501H)
MOV A, M                ; Load the high byte of the number into A (A = high byte)
MOV C, A                ; Copy the high byte into C for comparison

CALL REVERSE            ; Call subroutine to reverse the number

; After reversing, compare the original and reversed number
MOV A, B                ; Load the original low byte into A
CMP D                    ; Compare it with the reversed low byte
JNZ NOT_PALINDROME       ; If not equal, it's not a palindrome

MOV A, C                ; Load the original high byte into A
CMP E                    ; Compare it with the reversed high byte
JNZ NOT_PALINDROME       ; If not equal, it's not a palindrome

MVI H, 3000H            ; Load HL with address 3000H to store the result
MVI M, 01H              ; Store 01 (Palindrome) in memory 3000H
JMP END_PROGRAM         ; Jump to end of program

NOT_PALINDROME:
MVI H, 3000H            ; Load HL with address 3000H to store the result
MVI M, 00H              ; Store 00 (Not Palindrome) in memory 3000H

END_PROGRAM:
HLT                      ; Halt the program
REVERSE:
MOV D, B                ; Copy the original low byte to D (reversed low byte)
MOV E, C                ; Copy the original high byte to E (reversed high byte)
RET                      ; Return from the reverse subroutine
```

Q.No.9: Write an ALP to multiply the number 37H and 42H and keep result in memory.

[2016 Spring]

```
MVI H, 2500H            ; Load HL with the address of the first byte (2500H)
MVI L, 2501H            ; Load L with the address of the second byte (2501H)
MOV A, M                ; Load the first number (37H) into A
MOV B, A                ; Copy the first number into register B
INX H                   ; Increment HL to point to the second number (2501H)
MOV A, M                ; Load the second number (42H) into A

; Multiply the numbers in A and B (A * B)
MOV C, A                ; Copy the second number into register C
MOV A, B                ; Load the first number into A
MUL                      ; Multiply A (37H) by C (42H), store the result in A
and B

; Store the result in memory at address 2600H
LXI H, 2600H            ; Load HL with address 2600H (where result will be
stored)
MOV M, A                ; Store the low byte of the result in memory 2600H
```

```
INX H                  ; Increment HL to point to the next memory location
MOV M, B               ; Store the high byte of the result in memory 2601H

HLT                     ; Halt the program
```

Q.No.10: Write an 8085 ALP that checks an array of 10 memory location starting from C000H for odd numbers including 0 and transfers the odd numbers in memory location from C00EH.

[2017 Fall]

```
LXI H, C000H         ; Load HL with address of the first byte in the array (C000H)
LXI D, C00EH         ; Load DE with address where odd numbers will be stored (C00EH)

MVI C, 0             ; Initialize counter C to 0 (count of odd numbers)

LOOP:
MOV A, M             ; Load the number from the current memory location into A
ANI 01H              ; AND the number with 01H to check if it is odd
JZ SKIP              ; If the result is 0, number is even, skip to next number

MOV B, A             ; If odd, move the number to register B
MOV M, B             ; Store the odd number in memory at the address in DE
INX D                  ; Increment DE to the next location for storing the next odd
number
INX H                 ; Increment HL to the next number in the array (C001H -> C002H
-> ...)

SKIP:
INX C                ; Increment the counter to check next number in the array
MOV A, C             ; Load the counter into A
CPI 0AH              ; Compare if the counter is 10 (all numbers processed)
JZ DONE              ; If counter reaches 10, stop the loop

JMP LOOP             ; Repeat loop for next number

DONE:
HLT                   ; Halt the program
```

Q.No.11: Write 8085 assembly programming language to find smallest number among ten 8-bit data's stored in memory location 5000H to 5009H. also store that value in 9100H and display output device 51H. [2018 Spring]

```
LXI H, 5000H          ; Load HL with address of the first byte (5000H)
MOV A, M              ; Load the first number into A (accumulator)
MOV B, A              ; Copy the first number to B (smallest number so far)
LXI D, 9100H          ; Load DE with the address where the smallest number will be
stored (9100H)
MVI C, 09H            ; Set loop counter to 9 (for 10 numbers)

LOOP:
INX H                 ; Move to the next memory location
MOV A, M              ; Load the next number into A
```

```
CMP B                  ; Compare the number with the current smallest number (B)
JC SKIP                ; If A is greater than or equal to B, skip the update
MOV B, A               ; Update B with the new smallest number

SKIP:
DCR C                  ; Decrement loop counter
MOV A, C               ; Load loop counter into A
CPI 00H                ; Check if all 10 numbers are processed
JNZ LOOP               ; If not, repeat the loop

MOV M, B               ; Store the smallest number in memory location 9100H
MVI A, B               ; Load the smallest number into A
MVI 51H, A             ; Output the smallest number to the output device at memory
location 51H

HLT                     ; Halt the program
```

Q.No.12: Write an assembly language program in 8085 to check whether the number stored in memory location D001H is positive or negative. If positive store it in C001H else store in C002H. [2019 Spring]

```
LDA D001H              ; Load the number from memory location D001H into
accumulator A
RLC                    ; Rotate left through the carry (This shifts bit 7 to
carry and bit 0 to bit 1)
JNC POSITIVE           ; If carry is not set, the number is positive, jump to
POSITIVE

; If the number is negative
LDA D001H              ; Reload the number from memory location D001H into
accumulator A
STA C002H              ; Store the negative number in memory location C002H
JMP DONE               ; Jump to DONE

POSITIVE:
LDA D001H              ; Reload the number from memory location D001H into
accumulator A
STA C001H              ; Store the positive number in memory location C001H

DONE:
HLT                    ; Halt the program
```

Q.No.13: Write an ALP for 8085 to count the integers available in an array starting from 7 memory location 3500H to 3510H for exactly divisible by two and save result in the register B.

[2020 Fall]

```
LXI H, 3500H          ; Load HL with the address of the first element (3500H)
MVI B, 00H            ; Initialize register B (counter) to 0
```

```
LOOP:
MOV A, M              ; Load the number at the current memory location into A
ANI 02H               ; AND the number with 02H (checking the least significant bit)
JZ EVEN               ; If result is 0, the number is divisible by 2 (even number)

INX H                 ; Increment HL to the next memory location
JMP SKIP              ; Skip counting the odd number

EVEN:
INX H                 ; Increment HL to the next memory location
INR B                 ; Increment the counter in B for the even number

SKIP:
MOV A, H              ; Load the higher byte of HL into A to check if we have
reached the end
CPI 35H               ; Compare the higher byte with 35H (for the address 3510H)
JZ DONE               ; If reached the end, stop the loop

JMP LOOP              ; Repeat the loop for the next memory location

DONE:
HLT                   ; Halt the program
```

Q.No.14: Write an ALP in 8085 to count the number of 1 in the given string '101101010' and display the result in C0C0H. [2020 Spring]

```
LXI H, 5000H           ; Load HL register with the starting address of the binary
string (can be 5000H for example)
MVI B, 00H             ; Initialize register B (counter) to 0
MVI C, 00H             ; Initialize register C for storing the result

; Assume the string is stored at memory locations 5000H to 5008H
LOOP:
MOV A, M               ; Load the current byte (8 bits) from memory into the
accumulator
MVI D, 08H             ; Set up the bit counter to 8 (since we have 8 bits in a
byte)
COUNT_BITS:
RLC                   ; Rotate left through carry, this shifts bits into carry and
MSB into A
JC INCREMENT_COUNT    ; If carry is set (bit was 1), increment the counter
JMP NEXT_BIT           ; Otherwise, move to the next bit

INCREMENT_COUNT:
INR B                  ; Increment the counter in B (each time a '1' bit is found)

NEXT_BIT:
DCR D                  ; Decrement bit counter
MOV A, D               ; Check if we have processed all 8 bits in the byte
CPI 00H                ; If D is 0, we have processed 8 bits, so move to the next
byte
JNZ COUNT_BITS         ; Repeat the loop for the remaining bits if D is not zero
```

```
INX H                 ; Move to the next byte in memory
MOV A, H              ; Load the high byte of HL into A (to check if we have
reached the end of the string)
CPI 09H               ; Compare it with the last address (5008H for 9 bytes)
JZ DONE               ; If we have processed all bytes, jump to DONE

JMP LOOP              ; Repeat the loop for the next byte

DONE:
LXI H, C0C0H          ; Load HL with the address where the count result should be
stored (C0C0H)
MOV M, B              ; Store the count of '1's in memory location C0C0H
HLT                   ; Halt the program
```

Q.No.15: Write an ALP in 8085 to check whether the number stored in memory location 2060H is prime of not. If the number is prime, store FFH in memory location C00FH else store 00H.

[2021 Fall]

```
LDA 2060H            ; Load the number from memory location 2060H into accumulator
MOV B, A             ; Move the number to register B (for comparison)
MVI C, 02H           ; Initialize register C to 2 (start of divisor check)

; Check if the number is less than 2 (not prime)
CMP B                ; Compare the number with the content of register B
JC NOT_PRIME         ; If number < 2, it's not prime, jump to NOT_PRIME

; Loop to check divisibility from 2 to B-1
DIV_LOOP:
MOV A, B             ; Move the number from B to accumulator A
CALL DIVIDE          ; Call the divide subroutine to check divisibility
JZ NOT_PRIME         ; If remainder is 0, the number is not prime, jump to NOT_PRIME

INX C                ; Increment divisor
MOV A, C             ; Compare if divisor reached the number
CMP B
JZ PRIME             ; If divisor is equal to number, it's prime, jump to PRIME
JMP DIV_LOOP         ; Repeat division loop

NOT_PRIME:
MVI A, 00H           ; If not prime, move 00H to accumulator
STA C00FH            ; Store 00H at memory location C00FH
JMP DONE              ; Jump to DONE

PRIME:
MVI A, FFH           ; If prime, move FFH to accumulator
STA C00FH            ; Store FFH at memory location C00FH

DONE:
HLT                   ; Halt the program

; Divide subroutine: Divides A by C, stores quotient in A, remainder in B
DIVIDE:
```

```
MOV D, C              ; Move divisor C to D
MOV E, A              ; Move numerator A to E
MOV A, D              ; Load divisor into A
DIV D                 ; Perform the division operation
MOV B, A              ; Store remainder in B
MOV A, E              ; Restore numerator A
RET
```

Q.No.16: Write an 8085 program to find smallest number among array of numbers.

[2022 Fall]

```
LXI H, 2500H         ; Load HL register pair with the starting address of the array
(2500H)
MOV A, M             ; Load the first number from memory (2500H) into the
accumulator
MOV B, A             ; Copy the first number into register B (to keep track of the
smallest number)

MOV C, 00H           ; Initialize register C to 00 (it will be used as a counter for
the array length)
MVI D, 09H           ; Load D register with the number of elements in the array
(assuming 10 elements)

NEXT_ELEMENT:
INX H                ; Move HL to the next memory location (next element in the
array)
MOV A, M             ; Load the next element from memory into the accumulator
CMP B                ; Compare the next element with the current smallest (in
register B)
JC NEXT_COMPARE      ; If the next element is greater than or equal to the smallest,
continue

MOV B, A             ; If the next element is smaller, update register B with the
new smallest

NEXT_COMPARE:
DCR D                ; Decrement the counter (D) for the array length
MOV A, D             ; Move the counter to accumulator
CPI 00H              ; If counter is 0, all elements have been processed
JNZ NEXT_ELEMENT     ; If counter is not 0, continue with the next element

LXI H, 3000H         ; Load HL with the memory location where the smallest number
will be stored (3000H)
MOV M, B             ; Store the smallest number (in register B) to memory location
3000H

HLT                   ; Halt the program
```

Q.No.17: Write an ALP to find out greatest number among ten 8-bit data stored in memory location C000H to C009H. also store that value in D000H. [2023 Spring]

```
LXI H, C000H        ; Load HL register pair with the starting address of the array
(C000H)
MOV A, M            ; Load the first number from memory into the accumulator
MOV B, A            ; Copy the first number into register B (to keep track of the
greatest number)

MVI C, 09H          ; Load C register with the count of remaining elements (9
elements left)

NEXT_ELEMENT:
INX H               ; Increment HL to point to the next memory location
MOV A, M            ; Load the next number from memory into the accumulator
CMP B               ; Compare the next number with the current greatest (in
register B)
JNC SKIP_UPDATE     ; If the current number is smaller or equal, skip the update
MOV B, A            ; Update B with the new greatest number if the current number
is larger

SKIP_UPDATE:
DCR C               ; Decrement the counter for remaining elements
JNZ NEXT_ELEMENT    ; If more elements are left, continue to the next

LXI H, D000H        ; Load HL with the address to store the greatest number (D000H)
MOV M, B            ; Store the greatest number (in register B) at memory location
D000H

HLT                 ; Halt the program
```

Q.No.18: Write an ALP in 8085 to transfer 10 bytes of data of memory address starting from 5000H to 6000H. [2023 Fall (new)]

```
LXI H, 5000H        ; Load HL with the starting address of the source (5000H)
LXI D, 6000H        ; Load DE with the starting address of the destination (6000H)
MVI C, 0AH          ; Load counter (C register) with the number of bytes to
transfer (10 bytes)

TRANSFER_LOOP:
MOV A, M            ; Load the current byte from the source (HL) into the
accumulator
MOV E, A            ; Move the byte from accumulator to register E (to store in
memory)
MOV M, E            ; Store the byte at the destination (pointed by DE)
INX H               ; Increment HL to point to the next source byte
INX D               ; Increment DE to point to the next destination address
DCR C               ; Decrement the counter
JNZ TRANSFER_LOOP   ; If counter is not zero, repeat the loop

HLT                 ; Halt the program
```

Q.No.19: Write ALP program in 8085 to find the sum of even numbers located from address D050H-D059H and store the result in E055H memory address. [2024 Spring (new)]

```
LXI H, D050H        ; Load HL register pair with the starting address (D050H)
MVI B, 00H          ; Initialize the sum (B register) to 0
MVI C, 0AH          ; Load C register with the count of numbers (10 numbers in
total)

NEXT_ELEMENT:
MOV A, M            ; Load the current number into the accumulator
ANI 01H             ; Perform AND operation with 01H to check if the number is even
JNZ SKIP_ADD        ; If the result is not 0 (odd number), skip addition

MOV D, A            ; Move the even number to register D
MOV A, B            ; Move the current sum to accumulator
ADD D               ; Add the even number to the current sum
MOV B, A            ; Store the updated sum back in register B

SKIP_ADD:
INX H               ; Increment HL to point to the next memory location
DCR C               ; Decrement the counter
JNZ NEXT_ELEMENT    ; If more numbers are left, repeat the loop
LXI H, E055H        ; Load HL with the address to store the sum (E055H)
MOV M, B            ; Store the final sum from register B into memory
HLT                 ; Halt the program
```

Q.No.20: Write ALP program in 8085 to positive and negative numbers from given 10 numbers stored in memory location starting at 5000H.

```
    LXI H, 5000H        ; Load starting address of input numbers into HL
    LXI D, 6000H        ; Load starting address for positive numbers into DE
    LXI B, 7000H        ; Load starting address for negative numbers into BC
    MVI C, 0AH          ; Load count of numbers (10) into C
START:MOV A, M            ; Load current number into accumulator
    ANI 80H             ; Mask the MSB to check the sign
    JZ POSITIVE         ; If MSB is 0, it's a positive number

NEGATIVE:
    MOV A, M            ; Load current number into accumulator
    STAX B              ; Store the number at BC (negative numbers location)
    INX B               ; Increment BC to point to the next location
    JMP NEXT            ; Jump to process the next number
POSITIVE:MOV A, M           ; Reload current number into accumulator
    STAX D              ; Store the number at DE (positive numbers location)
    INX D               ; Increment DE to point to the next location
NEXT:INX H               ; Increment HL to point to the next number
    DCR C               ; Decrement count
    JNZ START           ; Repeat until all numbers are processed
    HLT                 ; Halt the program
```

Q.No.1: Write ALP program in 8085 to find the sum of even numbers located from address D050H-D059H and store the result in E055H memory address.

```
; Program to calculate the sum of even numbers between D050H-D059H
; and store the result at E055H

LXI H, D050H     ; Load the starting address of the array into HL pair
MVI C, 0AH       ; Counter for 10 bytes (10 = D059H - D050H + 1)
MVI D, 00H       ; Initialize D register to store the sum (D = 0)

START: MOV A, M  ; Load the current memory value into A
ANI 01H          ; Check if the number is even (A & 01H)
JNZ NEXT         ; If result is not zero, skip adding to sum

ADD D            ; Add the value in A to D
MOV D, A         ; Store the result back into D

NEXT: INX H      ; Increment HL to point to the next memory location
DCR C            ; Decrement counter
JNZ START        ; If counter is not zero, repeat the loop

LXI H, E055H     ; Load the destination address (E055H) into HL
MOV M, D         ; Store the sum at E055H

HLT              ; Halt the program
```

## Section 2: 8086 Microprocessor ALP

Q.No.2: The six data bytes are stored from memory location 3000H to 3005H. Write an 8086 ALP to transfer block of data to new location 5000H to 5005H. [**2013 Fall]**

```
.MODEL SMALL
.STACK 100H

.CODE
MAIN PROC
    ; Initialize data segment (DS) to 3000H
    MOV AX, 3000H            ; Load segment address 3000H into AX
    MOV DS, AX               ; Set DS to 3000H (source segment)

    ; Initialize extra segment (ES) to 5000H
    MOV AX, 5000H            ; Load segment address 5000H into AX
    MOV ES, AX               ; Set ES to 5000H (destination segment)

    ; Set up pointers and counter
    MOV SI, 0000H            ; Set SI (source offset) to 0000H
    MOV DI, 0000H            ; Set DI (destination offset) to 0000H
    MOV CX, 06H              ; Set CX (counter) to 6 (number of bytes to transfer)

    ; Transfer data
TRANSFER_LOOP:
    MOV AL, DS:[SI]          ; Load byte from DS:SI into AL
    MOV ES:[DI], AL          ; Store byte from AL into ES:DI
    INC SI                   ; Increment SI (source offset)
    INC DI                   ; Increment DI (destination offset)
    LOOP TRANSFER_LOOP       ; Repeat until CX = 0

    ; End of program
    MOV AH, 4CH              ; Terminate program
    INT 21H
MAIN ENDP
END MAIN
```

Q.No.3: Write an assembly language programming for 8086 to find the square root of a given number. Assume that a number is of two digits and is perfect square**.  [2013 Spring]**

```
.MODEL SMALL
.STACK 100H

.DATA
    NUMBER DB 25             ; Two-digit perfect square number (change as needed)
    RESULT DB ?              ; Variable to store the square root

.CODE
MAIN PROC
    MOV AX, @DATA            ; Initialize data segment
    MOV DS, AX

    MOV AL, NUMBER           ; Load the number into AL
    MOV BL, 4                ; Start checking from 4 (since 4^2 = 16, the smallest two-digit
perfect square)
```

```
FIND_SQUARE_ROOT:
    MOV AL, NUMBER            ; Reload the number into AL
    MOV AH, 0                 ; Clear AH for division
    DIV BL                    ; Divide AX by BL (AL = AX / BL, AH = AX % BL)
    CMP AL, BL                ; Compare AL (quotient) with BL
    JE FOUND_SQUARE_ROOT      ; If AL == BL, then BL is the square root
    INC BL                    ; Increment BL to check the next number
    JMP FIND_SQUARE_ROOT      ; Repeat the process

FOUND_SQUARE_ROOT:
    MOV RESULT, BL            ; Store the square root in RESULT

    ; End of program
    MOV AH, 4CH               ; Terminate program
    INT 21H
MAIN ENDP
END MAIN
```

Q.No.4: Write an 8086 ALP for MASM to display the string "POKHARA UNIVERSITY" without using 09h on screen. Explain all steps and assume necessary data. [**2014 Fall]**

```
.MODEL SMALL
.STACK 100H

.DATA
    MESSAGE DB 'POKHARA UNIVERSITY$'  ; Define the string to display
    LEN EQU $-MESSAGE                 ; Calculate the length of the string

.CODE
MAIN PROC
    MOV AX, @DATA             ; Initialize data segment
    MOV DS, AX

    MOV SI, OFFSET MESSAGE    ; Load the offset address of the string into SI
    MOV CX, LEN               ; Load the length of the string into CX (counter)

DISPLAY_LOOP:
    MOV DL, [SI]              ; Load the current character into DL
    CMP DL, '$'               ; Check if the character is the end-of-string marker
    JE END_PROGRAM            ; If yes, end the program
    MOV AH, 02H               ; Set AH to 02H (display character function)
    INT 21H                   ; Call interrupt to display the character
    INC SI                    ; Move to the next character in the string
    LOOP DISPLAY_LOOP         ; Repeat until all characters are displayed

END_PROGRAM:
    MOV AH, 4CH               ; Terminate program
    INT 21H
MAIN ENDP
END MAIN
```

Q.No.5: Write an 8086 ALP for MASM to find a square of a given number. [**2014 Spring]**

```
.MODEL SMALL    ; Define the memory model as SMALL (code and data in separate segments)
.STACK 100H     ; Define a stack of size 256 bytes
```

```
.DATA
    NUM DB 5      ; Input number (change this value as needed)
    SQUARE DW ?   ; Variable to store the square of the number

.CODE
MAIN PROC
    MOV AX, @DATA ; Load the data segment address into AX
    MOV DS, AX    ; Set DS (data segment) to point to the .DATA section

    ; Load the number into AL (since MUL operates on AL/AX)
    MOV AL, NUM   ; Move the value of NUM into AL

    ; Compute the square: AL * AL = AX
    MUL AL        ; Multiply AL by AL, result stored in AX

    ; Store the result (AX) into SQUARE
    MOV SQUARE, AX

    ; Terminate the program
    MOV AH, 4CH   ; DOS interrupt function to exit the program
    INT 21H       ; Call DOS interrupt

MAIN ENDP
END MAIN
```

Q.No.6: Write a program to reverse the given string of 8086. [**2014 Spring** ]

```
.model small
.stack 100h

.data
    str db "HELLO, 8086!", "$"  ; String to reverse
    msg db "Reversed: $"

.code
start:
    mov ax, @data
    mov ds, ax

    ; Print "Reversed: "
    mov dx, offset msg
    mov ah, 09h
    int 21h

    ; Push characters onto stack (excluding '$')
    lea si, str
push_loop:
    mov al, [si]
    cmp al, '$'
    je pop_chars
    push ax
    inc si
    jmp push_loop

pop_chars:
    ; Pop characters and print them
    mov ah, 02h  ; DOS function to print character
pop_loop:
    pop dx
    int 21h
    cmp dx, str  ; Stop if we reach the start of the string
    jne pop_loop

    ; Exit
    mov ah, 4Ch
    int 21h

end start
```

Q.No.7: Write an 8086 ALP to find the factorial of 08H using DOS BIOS Interrupt. [**2015 fall]**

```
.model small
.stack 100h

.data
    num       db 08h              ; Number to calculate factorial (8)
    fact      dw 1                ; Store factorial (16-bit, safe for 8!)
    msg       db "Factorial: $"
    result    db 6 dup(0)         ; Buffer for ASCII conversion

.code
start:
    mov ax, @data
    mov ds, ax

    ; Print "Factorial: "
    mov dx, offset msg
    mov ah, 09h
    int 21h

    ; Compute factorial of num
    mov cx, num     ; CX = 8
    mov ax, 1       ; AX = 1 (initialize factorial)

fact_loop:
    mul cx          ; DX:AX = AX * CX (handle overflow)
    loop fact_loop  ; Decrease CX until 1

    ; Convert AX to ASCII (since result fits in 16-bit)
    mov bx, ax      ; Copy factorial result
    lea di, result+4
    mov byte ptr [di+1], '$' ; Set '$' at the end

convert:
    mov dx, 0
    mov ax, bx
    mov cx, 10
    div cx          ; AX = quotient, DX = remainder
    add dl, '0'     ; Convert remainder to ASCII
    mov [di], dl    ; Store digit
    dec di
    mov bx, ax
    cmp bx, 0
    jne convert

    ; Print the result
    lea dx, [di+1]
    mov ah, 09h
    int 21h

    ; Exit program
    mov ah, 4Ch
    int 21h
end start
```

Q.No.8: Write an 8086 ALP in masm to display the string "POKHARA UNIVERSITY" in reverse order. [**2015 fall ]**

```
.model small

.stack 100h


.data

    str db "POKHARA UNIVERSITY ", "$"  ; String to reverse

    msg db "Reversed: $"


.code

start:

    mov ax, @data

    mov ds, ax


    ; Print "Reversed: "

    mov dx, offset msg

    mov ah, 09h

    int 21h


    ; Push characters onto stack (excluding '$')

    lea si, str
```

```
push_loop:

    mov al, [si]

    cmp al, '$'

    je pop_chars

    push ax

    inc si

    jmp push_loop


pop_chars:

    ; Pop characters and print them

    mov ah, 02h  ; DOS function to print character

pop_loop:

    pop dx

    int 21h

    cmp dx, str  ; Stop if we reach the start of the string

    jne pop_loop


    ; Exit

    mov ah, 4Ch

    int 21h


end start
```

Q.No.9 : Write an 8086 ALP for MASM in DOS-BIOS mode to display the string "Pokhara University" on screen without using 09H. [**2016 fall]**

```
.MODEL SMALL    ; Define the memory model as SMALL (code and data in separate segments)
.STACK 100H      ; Define a stack of size 256 bytes

.DATA
    MESSAGE DB 'Pokhara University$'  ; Define the string to display
    LEN EQU $-MESSAGE                 ; Calculate the length of the string

.CODE
MAIN PROC
    MOV AX, @DATA             ; Initialize data segment
    MOV DS, AX

    MOV SI, OFFSET MESSAGE   ; Load the offset address of the string into SI
    MOV CX, LEN              ; Load the length of the string into CX (counter)
```

```
DISPLAY_LOOP:
    MOV DL, [SI]             ; Load the current character into DL
    CMP DL, '$'              ; Check if the character is the end-of-string marker
    JE END_PROGRAM           ; If yes, end the program
    MOV AH, 02H              ; Set AH to 02H (display character function)
    INT 21H                  ; Call interrupt to display the character
    INC SI                   ; Move to the next character in the string
    LOOP DISPLAY_LOOP        ; Repeat until all characters are displayed

END_PROGRAM:
    MOV AH, 4CH              ; Terminate program
    INT 21H
MAIN ENDP
END MAIN

-----------------------------------------------OR--------------------------
.model small
.stack 100h

.data
    str db "Pokhara University", 0  ; String with null termination

.code
start:
    mov ax, @data
    mov ds, ax

    ; Load SI with string address
    lea si, str

print_loop:
    mov al, [si]      ; Load character
    cmp al, 0         ; Check if end of string (null terminator)
    je exit           ; If yes, exit program

    mov ah, 0Eh       ; BIOS teletype output function
    int 10h           ; Call BIOS to print character

    inc si            ; Move to next character
    jmp print_loop    ; Repeat until null terminator

exit:
    mov ah, 4Ch       ; DOS terminate program function
    int 21h

end start
```

Q.No.10: Write an assembly language program for 8086 to find the sum of two numbers input by the user through the keyboard and display the sum on the screen.

Q.No.11: Write an 8086 ALP for MASM to display the "POKHARAUNIVERSITY" in reverse order. [**2016 spring]**

```
.model small
.stack 100h

.data
    str db "POKHARAUNIVERSITY$"

.code
start:
    mov ax, @data
    mov ds, ax

    mov si, 17
    lea di, str
    add di, si
    dec di

print_loop:
    mov dl, [di]
    mov ah, 02h
    int 21h
    dec di
    dec si
    jnz print_loop

    mov ah, 4Ch
    int 21h
end start
```

Q.No.12: Write an assembly language program in 8086 which reads a string from a keyboard and then displays the string in the reverse order. [**2017 Fall]**

```
.model small
.stack 100h

.data
    str db 30 dup('$')   ; Buffer for string
    msg1 db "Enter string: $"
    msg2 db 0Dh,0Ah, "Reversed: $"

.code
start:
    mov ax, @data
    mov ds, ax

    mov dx, offset msg1
    mov ah, 09h
    int 21h

    mov dx, offset str
    mov ah, 0Ah
    int 21h

    mov dx, offset msg2
    mov ah, 09h
    int 21h

    lea si, str
    add si, 29

reverse_loop:
    cmp byte ptr [si], '$'
    je done
    mov dl, [si]
    mov ah, 02h
    int 21h
    dec si
    jmp reverse_loop

done:
    mov ah, 4Ch
    int 21h
end start
```

Q.No.13: Write a program in 8086 to find whether a number is positive or negative. [**2017 Fall]**

```
.model small
.stack 100h

.data
    msg_pos db "Positive$", 0
    msg_neg db "Negative$", 0
    msg_in  db "Enter a number: $"

.code
start:
    mov ax, @data
    mov ds, ax

    mov dx, offset msg_in
    mov ah, 09h
    int 21h

    mov ah, 01h
    int 21h

    cmp al, '0'
    jge positive

negative:
    mov dx, offset msg_neg
    jmp display

positive:
    mov dx, offset msg_pos

display:
    mov ah, 09h
    int 21h

    mov ah, 4Ch
    int 21h
end start
```

Q.No.14: Write an assembly language program in 8086 which reads a string from the keyboard and then displays the string in the reverse order. For example if the input string read from the keyboard is "POKHARA UNIVERSITY", the output should be "YTISREVINU ARAHKOP". [**2017 spring]**

```
.model small
.stack 100h

.data
    str db 30 dup('$')   ; Buffer for string
    msg1 db "Enter string: $"
    msg2 db 0Dh,0Ah, "Reversed: $"

.code
start:
    mov ax, @data
    mov ds, ax

    mov dx, offset msg1
    mov ah, 09h
    int 21h

    mov dx, offset str
    mov ah, 0Ah
    int 21h

    mov dx, offset msg2
    mov ah, 09h
    int 21h

    lea si, str
    add si, 29
```

```
reverse_loop:
    cmp byte ptr [si], '$'
    je done
    mov dl, [si]
    mov ah, 02h
    int 21h
    dec si
    jmp reverse_loop

done:
    mov ah, 4Ch
    int 21h
end start
```

Q.No.15: Write a program in 8086. to display "MICROPROCESSOR IS EASY TO LEARN" in lower case. **[2018 Fall]**

```
.model small
.stack 100h

.data
    str db "MICROPROCESSOR IS EASY TO LEARN$"

.code
start:
    mov ax, @data
    mov ds, ax

    lea si, str

lowercase_loop:
    mov al, [si]
    cmp al, '$'
    je done
    add al, 20h
    mov dl, al
    mov ah, 02h
    int 21h
    inc si
    jmp lowercase_loop

done:
    mov ah, 4Ch
    int 21h
end start
```

Q.No.16: Write an assembly language program in 8086 to display the string "Computer Engineering" at console. [**2018 Spring]**

```
.model small
```

```
.stack 100h

.data
    msg db "Computer Engineering$"

.code
start:
    mov ax, @data
    mov ds, ax

    mov dx, offset msg
    mov ah, 09h
    int 21h

    mov ah, 4Ch
    int 21h
end start
```

Q.No.17: Write an 8086 ALP for MASM in DOS mode to print each word of a string in different lines**. [2019 Fall]**

```
.model small
.stack 100h

.data
    str db "POKHARA UNIVERSITY$"

.code
start:
    mov ax, @data
    mov ds, ax

    lea si, str
print_loop:
    mov al, [si]
    cmp al, ' '
    je new_line
    mov dl, al
    mov ah, 02h
    int 21h
    inc si
    jmp print_loop

new_line:
    mov dl, 0Dh
    mov ah, 02h
    int 21h
    mov dl, 0Ah
    mov ah, 02h
    int 21h
    inc si
    jmp print_loop
end start
```

Q.No.18: Write an assembly language program in 8086 to find the largest number among 10 blocks of data and store the largest value in location "largest" . [**2020 fall]**

```
 .model small
.stack 100h

.data
    data db 10, 5, 8, 20, 12, 9, 30, 15, 19, 18
    largest db 0

.code
start:
    mov ax, @data
    mov ds, ax

    lea si, data

    mov al, [si]   ; First element is the initial largest
    inc si
    mov cx, 9      ; 9 more values to compare

find_largest:
    mov bl, [si]
    cmp al, bl
    jge next
    mov al, bl

next:
    inc si
    loop find_largest

    mov dx, offset largest
    mov [dx], al

    mov ah, 4Ch
    int 21h
end start
```

Q.No.19: Write an 8086 program to enter a string from the keyboard. Count the number of repetitions of letter 'a' or 'A'. If the count is even, display "POKHARA" else display "UNIVERSITY". [**2021 Fall]**

```
.model small
.stack 100h

.data
    prompt db "Enter a string: $"
    msg1 db "POKHARA$"
    msg2 db "UNIVERSITY$"
    count db 0           ; To count occurrences of 'a' or 'A'

.code
start:
    mov ax, @data
    mov ds, ax

    ; Display the prompt
    mov dx, offset prompt
    mov ah, 09h
    int 21h

    ; Read the string
    lea dx, str
    mov ah, 0Ah
    int 21h

    ; Initialize count to 0
    mov al, 0
    mov [count], al

    ; Count occurrences of 'a' or 'A'
    lea si, str + 2     ; Start of input string
    mov cx, 30          ; Maximum length of input string

count_loop:
    mov al, [si]
    cmp al, '$'
    je check_count      ; End of string reached
    cmp al, 'a'
    je increment_count
    cmp al, 'A'
    je increment_count
    jmp next_char

increment_count:
    inc byte ptr [count]

next_char:
    inc si
    loop count_loop

check_count:
    ; Check if count is even or odd
    mov al, [count]
    and al, 1
    jz display_pokhara
    jmp display_university

display_pokhara:
    mov dx, offset msg1
    mov ah, 09h
    int 21h
    jmp end_program

display_university:
    mov dx, offset msg2
    mov ah, 09h
    int 21h

end_program:
    mov ah, 4Ch
    int 21h
end start
```

Q.No.20: Write an 8086 ALP to check whether a given string is palindrome or not. [**2021 spring]**

```
.model small
.stack 100h

.data
    prompt db "Enter a string: $"
    msg_palindrome db "Palindrome$"
    msg_not_palindrome db "Not Palindrome$"
    count db 0           ; To store the string length

.code
start:
    mov ax, @data
    mov ds, ax

    ; Display the prompt
    mov dx, offset prompt
    mov ah, 09h
    int 21h

    ; Read the string
    lea dx, str
    mov ah, 0Ah
    int 21h

    ; Set up pointers for comparison
    lea si, str + 2  ; Start of the string
    lea di, str + 30 ; End of the string (max length)
    dec di           ; Move back one byte from the max length

    ; Compare characters from both ends
    mov al, 1         ; Assume it's a palindrome

compare_loop:
    cmp si, di
    jge palindrome_check
    mov al, [si]
    cmp al, [di]
    je continue_check
    mov al, 0         ; Not a palindrome
    jmp display_result

continue_check:
    inc si
    dec di
    jmp compare_loop

palindrome_check:
    cmp al, 1
    je display_palindrome

display_result:
    cmp al, 1
    je display_palindrome
    mov dx, offset msg_not_palindrome
    mov ah, 09h
    int 21h
    jmp end_program

display_palindrome:
    mov dx, offset msg_palindrome
    mov ah, 09h
    int 21h

end_program:
    mov ah, 4Ch
    int 21h
end start
```

Q.No.21: Write an ALP in 8086 to display POKHARA UNIVERSITY without using 09h function. [**2022 fall]**

```
.model small
.stack 100h

.data
    msg db "POKHARA UNIVERSITY$"

.code
start:
    mov ax, @data
    mov ds, ax

    lea si, msg
print_loop:
    mov al, [si]
    cmp al, '$'
    je end_program
    mov dl, al
    mov ah, 02h
    int 21h
    inc si
    jmp print_loop

end_program:
    mov ah, 4Ch
    int 21h
end start
```

Q.No.22: Write an ALP for 8086 to user input a string from the keyboard and display its reverse form in the screen. [**2023 Spring]**

```
.model small
.stack 100h

.data
    prompt db "Enter a string: $"
    str db 30 dup('$')   ; Buffer for the string

.code
start:
    mov ax, @data
    mov ds, ax

    ; Display the prompt
    mov dx, offset prompt
    mov ah, 09h
    int 21h

    ; Read the string
    lea dx, str
    mov ah, 0Ah
    int 21h

    ; Find the end of the string (ignoring the first 2 bytes of the input buffer)
    lea si, str + 2     ; Skip the first 2 bytes (length and unused byte)
    mov di, si

find_end:
    cmp byte ptr [di], '$'
    je reverse_string
    inc di
    jmp find_end

reverse_string:
    dec di

reverse_loop:
    cmp di, si
    jl end_program
    mov dl, [di]
    mov ah, 02h
    int 21h
    dec di
    jmp reverse_loop

end_program:
    mov ah, 4Ch
    int 21h
end start
```

*The End*