# MICROPROCESSORS AND ITS PERIPHRALS

# ASSIGNMENT

*SUBMITTED BY*

PRITEESH GOYAL

15103005

CSE 2nd YEAR

*SUBMITTED TO*

POONAM SAINI

# INDEX

**QUESTION 1**: WAP to add a datatype located at segment = 2000H and offset = 500H to another datatype available at 600H in the same segment and store the result at 700H in the same segment.

**ASSEMBLY CODE:**

```
MOV AX,2000H;

MOV DS,AX;

MOV [500H],05H;

MOV AX,[500H];

MOV [600H],10H;

ADD AX,[600H];

MOV [600H],AX;

HLT;
```

**QUESTION 2:** WAP to move the content of memory location 500H to register BX and CX. Add immediate byte 5H to data residing in memory location whose address is computed using DS=2000H and offset=600H. Store the result of addition in 700H.

**ASSEMBLY CODE:**

```
MOV AX,2000H;

MOV DS,AX;

MOV BX,[500H];

MOV CX,BX;

ADD [600H],05H;

MOV DX,[600H];

MOV [700H],DX;

HLT;
```

**QUESTION 3:** Add contents of memory segment 2000h offset 500h and content of memory location 3000h offset 600h and store result in 5000h offset 700h.

**ASSEMBLY CODE:**

```
MOV AX,2000H;

MOV DS,AX;

MOV [500H],5H;

MOV BX,[500H];
MOV AX,3000H;
MOV DS,AX;
MOV [600H],10H;
ADD BX,[600H];
MOV AX,5000H;
MOV DS,AX;
MOV [700H],BX;
HLT;
```

**QUESTION 4:** Move a 16 byte string from offset 200H to 300H in the segment 7000H, with and without string manipulation operations.

**ASSEMBLY CODE:**

WITHOUT STRING MANIPULATION

```
MOV AX,7000H;
MOV DS,AX;
MOV SI,0200H;
MOV DI,0300H;
MOV CX,0010H;
BACK: MOV AL,[SI];
          MOV [DI],AL;
    INC SI;
    INC DI;
    LOOP BACK;
    HLT;
```

WITH STRING MANIPULATION

```
MOV AX,7000H
MOV DS, AX
MOV ES, AX
MOV CX,0010H
MOV SI,0200H
MOV DI,0300H
```

```
CLD
REP MOVSB
```

**QUESTION 5:** WAP a program to check whether a given string is palindrome or not .

**ASSEMBLY CODE:**

```
org 100h
INCLUDE 'emu8086.INC'
MOV AX,5000H
MOV DS, AX
MOV ES, AX
MOV DI, 100H
MOV SI, DI
MOV CX, 3H
MOV [100H], 'A'
MOV [101H], 'B'
MOV [102H], 'A'

ADD DI, 2H
SHR CX, 1

BACK:
MOV AL, [SI]
CMP AL, [DI]
JNZ NEXT
INC SI
DEC DI
LOOP BACK

PRINT "Palindrome"
ret
```

```
NEXT: PRINT "Not Palindrome"
ret
```

**QUESTION 6:** WAP to reverse a given string

**ASSEMBLY CODE:**

```
org 100h
INCLUDE 'emu8086.INC'
MOV AX,5000H
MOV DS, AX
MOV ES, AX
MOV SI, 103H
MOV [100H], 'A'
MOV [101H], 'K'
MOV [102H], 'Y'
MOV [103H], 'O'
MOV DI, 104H
BACK:
CMP DI,0FFH
JZ NEXT
INC SI
DEC DI
MOV BX,[DI]
MOV [SI],BX
LOOP BACK

NEXT: RET
```

**QUESTION 7:**Search a key element of ‘n’ 16  bit numbers using the Binary Search Algorithm

**ASSEMBLY CODE:**

```
      ASSUME DS: DATA, CS: CODE
DATA SEGMENT
NUM DW 1234H,5678H,6256H,7321H,8454H
            COUNT DW 05H
            KEY DW 5678H
MSG1 DB 10, 13, 'KEY FOUND', '$'
MSG2 DB 10, 13, 'KEY NOT FOUND', '$'
DATA ENDS
CODE SEGMENT
START: MOV AX, DATA
                        MOV DS, AX
                        XOR AX, AX
                        MOV BX, 1
                        MOV DX, COUNT
                        MOV CX, KEY
AGAIN: CMP BX, DX
                        JA NOTFOUND
                           MOV AX, BX
     ADD AX, DX
                       SHR AX, 1
                   MOV SI, AX
                       DEC SI
                       ADD SI, SI
                        CMP CX, NUM[SI]
                        JE FOUND
JA ABOVE
                        DEC AX
```

```
                 MOV DX, AX
               JMP AGAIN
                 ABOVE: INC AX
             MOV BX, AX
                 JMP AGAIN
FOUND: MOV DX, OFFSET MSG1
      MOV AH, 09H
      INT 21H
      JMP STOP
NOTFOUND: MOV DX, OFFSET MSG2
      MOV AH, 09H
      INT 21H
 STOP: MOV AH, 4CH
CODE  ENDS
END START
```

**QUESTION 8.** WAP to implement bubble sort to sort an array of numbers in ascending and descending order.

**i. Ascending order**

```
DATA SEGMENT
   ARR DB 12,6,34,23,98,13,65,22
    LEN DW $-ARR
    I DB 0
ENDS

CODE SEGMENT
  ASSUME DS:DATA CS:CODE
START:
   MOV AX,DATA
   MOV DS,AX
```

```
      MOV CX,LEN-1
OUTER:
      LEA SI,ARR
      MOV BX,0
INNER:
      INC BX
      MOV AL,ARR[SI]
      INC SI
      CMP ARR[SI],AL
      JB SKIP
      XCHG AL,ARR[SI]
      MOV ARR[SI-1],AL
SKIP:
      CMP BX,CX
      JL  INNER
      LOOP OUTER
      MOV AH,4CH
ENDS
END START
```

**ii. <u>Ascending Order</u>**

```
DATA SEGMENT
     ARR DB 12,6,34,23,98,13,65,22
       LEN DW $-ARR
       I DB 0
ENDS
CODE SEGMENT
```

```
    ASSUME DS:DATA CS:CODE
START:
      MOV AX,DATA
      MOV DS,AX
      MOV CX,LEN-1
OUTER:
      LEA SI,ARR
      MOV BX,0
INNER:
      INC BX
      MOV AL,ARR[SI]
      INC SI
      CMP AL,ARR[SI]
      JL SKIP
      XCHG AL,ARR[SI]
      MOV ARR[SI-1],AL
SKIP:
      CMP BX,CX
      JB  INNER
      LOOP OUTER
      MOV AH,4CH
ENDS
END START
```