# Name: Muhammad Maiz Nadeem

Reg. ID: SP21-BCS-052

# Question 1:

# **Write a program that calculates the first 12 terms of fibonacci sequence and stores them in an array.**

# Answer:

|  |  |
| --- | --- |
| 1 | ORG 100h |
| 2 |  |
| 3 | .DATA |
| 4 |  |
| 5 | ARRAY *DW* 12 DUP(?) |
| 6 |  |
| 7 | .CODE |
| 8 |  |
| 9 | MOV CX, 12 |
| 10 | MOV SI, OFFSET ARRAY |
| 11 |  |
| 12 | XOR AX, AX |
| 13 | MOV BX, 0 |
| 14 | MOV DX, 1 |
| 15 | MOV [SI], BX |
| 16 | ADD SI, 2 |
| 17 | MOV [SI], DX |
| 18 | ADD SI, 2 |
| 19 |  |
| 20 | FIBO:   XOR AX, AX |
| 21 | ADD AX, BX |
| 22 | ADD AX, DX |
| 23 | MOV BX, DX |
| 24 | MOV DX, AX |
| 25 | MOV [SI], AX |
| 26 | ADD SI, 2 |
| 27 | LOOP FIBO |
| 28 |  |
| 29 | RET |

# Question 2:

# **(Modified version of exercise from chapter 10)**

# **To sort an array A of N elements by the bubblesort method, we proceed as follows:**

# **Pass 1: For j= 2 ... N, If A(j) <A(j - 1) then swap A(j) and A(j-1).**

# **This will place the largest element In position N.**

# **Pass 2: For j = 2 ... N -1, if A(j)< A(j-1) then swap A(j) and A(j -1).**

# **This will place the second largest element in position N -1.**

# **Pass N -1: If A(2) < A(1), then swap A[2) and A(1).**

# **At this point the array is sorted.**

# **Demonstration :**

# **Initial Data: 7 5 3 9 1**

# **Pass 1: 5 3 7 1 9**

# **Pass 2: 3 5 1 7 9**

# **Pass 3: 3 1 5 7 9**

# **Pass 4: 1 3 5 7 9**

# **Write a program segment to sort a byte array (DATA DB 7 5 3 9 1)by the bubblesort algorithm. The program should put the offset address of the array in Sl and the number of elements in variable N.**

# Answer:

|  |  |
| --- | --- |
| 1 | ORG 100h |
| 2 |  |
| 3 | .DATA |
| 4 |  |
| 5 | SWAP *DB* ? |
| 6 | N *DW* 6 |
| 7 | ARRAY *DB* 2, 6, 7, 1, 4, 7 |
| 8 |  |
| 9 | .CODE |
| 10 |  |
| 11 | START:      MOV SWAP, 0 |
| 12 | MOV SI, OFFSET ARRAY    ; storing array offset in SI |
| 13 | XOR BX, BX |
| 14 |  |
| 15 |  |
| 16 | COMPARISON: MOV AL, [SI+BX] |
| 17 | CMP AL, [SI+BX+1] |
| 18 | JBE NOSWAP |
| 19 |  |
| 20 | ;SWAP: |
| 21 | MOV SWAP, 1             ; indicates swapping has occurred |
| 22 | XCHG AL, [SI+BX+1] |
| 23 | MOV [SI+BX], AL |
| 24 |  |
| 25 |  |
| 26 | NOSWAP:     INC BX |
| 27 | CMP BX, N               ; using N as number of elements to loop through array |
| 28 | JNZ COMPARISON |
| 29 |  |
| 30 | CMP SWAP, 0             ; checks if swapping has occurred |
| 31 | JNZ START               ; if yes, loops again. if not, program ends |
| 32 |  |
| 33 | RET |

# Question 3:

# **Implement the following sorting algorithm for a byte array:**

# **i = N**

# **FOR i=N-1 times DO**

# **Find the position k of the largest element among A[1] ….A[i]**

# **Swap A[k] and A[i]**

# **i=i-1**

# **END FOR**

# **Usually algorithms are evaluated on speed by observing how many steps they took to sort a certain data set.**

# Answer:

|  |  |
| --- | --- |
| 1 | ORG 100h |
| 2 |  |
| 3 | .DATA |
| 4 | SWAP *DB* ? |
| 5 | N *DW* 6 |
| 6 | ARRAY *DB* 2, 6, 7, 1, 4, 8 |
| 7 |  |
| 8 | .CODE |
| 9 |  |
| 10 | MOV SI, OFFSET ARRAY            ; storing array offset in SI |
| 11 | SUB SI, 1 |
| 12 | INC N                           ; retaining value of N for first cycle |
| 13 |  |
| 14 | START: |
| 15 |  |
| 16 | DEC N                       ; decrementing N whenever the least number is caught |
| 17 | MOV SWAP, 0                 ; indicates least value is caught |
| 18 | INC SI                      ; incrementing SI whenever the least value is caught and stored in it |
| 19 | XOR BX, BX |
| 20 |  |
| 21 |  |
| 22 | COMPARISON: |
| 23 | MOV AL, [SI] |
| 24 | MOV DL, [SI+BX+1] |
| 25 | CMP AL, [SI+BX+1] |
| 26 | JBE NEXT |
| 27 |  |
| 28 | ;SWAP: |
| 29 | MOV SWAP, 1 |
| 30 | XCHG AL, [SI+BX+1] |
| 31 | MOV [SI], AL |
| 32 |  |
| 33 |  |
| 34 | NEXT: |
| 35 | INC BX |
| 36 | CMP BX, N                   ; using N as number of elements to loop through array |
| 37 | JNZ COMPARISON |
| 38 |  |
| 39 | CMP SWAP, 0                     ; checks if least value is caught |
| 40 | JNZ START                       ; if yes, loops again. if not, program ends |
| 41 |  |
| 42 | RET |

For the data set given in question 1 , which algorithm you feel is faster, Bubble sort or select sort? (Hint : See which algorithm does less swapping)

Selection sort requires less number of variables and has less cycles as it selects the indexes instead of exchanging them again and again.