|  |
| --- |
| **Q1: You have the two collections of number that are Multiplicand and Multiplier respectively. You need to multiply each element of Multiplicand to its corresponding Multiplier and store the result in Product array. You must use loop instruction to solve this problem [Marks: 10] Multiplicand BYTE 31h , 6Bh , 0Ch, 11h, 2Fh**  **Multiplier WORD 1Ch, 90h, 3Ah, 16h, 1Eh**  **Product DWORD 5 DUP(0)** |

**[Solution]**

INCLUDE Irvine32.inc

.data

Multiplicand BYTE 31h, 6Bh, 0Ch, 11h, 2Fh

Multiplier WORD 1Ch, 90h, 3Ah, 16h, 1Eh

Product DWORD 5 DUP(0)

.code

main proc

xor eax, eax ; Clear eax (eax = 0)

xor ebx, ebx ; Clear ebx (ebx = 0)

xor esi, esi ; Clear esi (ESI = 0, acting as the index)

move ecx, 5

multiply\_loop:

mov al, [Multiplicand + esi]

mov bx, [Multiplier + esi \* TYPE Multiplier]

mul bl ; AX = AL \* BL

movzx eax, ax ; Zero-extend AX to EAX

mov [Product + esi \* TYPE Product], eax

inc esi

Loop multiply\_loop

exit

main endp

end main

|  |
| --- |
| **Q2: Convert the following code into assembly language. Keep in mind that all the comparison instruction must be implemented through eflag register status flag (carry flag and zero flag). int array[] = {3,29,101,65,53}; int size, temp;**  **for (int step = 0; step < size - 1; step++)**  **{**  **int min\_idx = step;**  **for (int i = step + 1; i < size; i++)**  **{**  **if (array[i] < array[min\_idx])**  **min\_idx = i;**  **}**  **temp = array[min\_idx];**  **array[min\_idx] = array[step];**  **array[step] = temp;**  **}** |

**[Solution]**

.data

array DW 3,29,101,65,53

size DW ?

min\_indx DW ?

step DW ?

.code

Main Proc

mov size, lengthof array -1

mov ecx, size

mov step, 0

outerLoop:

mov min\_indx, step

push ecx

push step

mov ecx, size

innerLoop:

inc step

cmp step, ecx ;InnerLoop Comparision

jz: LexitInnerLoop

mov esi, step`

mov edi, min\_indx

mov edx, array[edi]

cmp array[esi], edx ; if(arr[i]<arr[min\_indx])

jz LNextIteration

jnc LNextIteration

mov min\_indx,esi ; min\_indx = i or step

LNextIteration:

jmp innerLoop

LexitInnerLoop:

mov eax, array[edi] ; temp = array[min\_indx]

mov ebx, array[esi] ; ebx = array[step or esi]

mov array[edi], ebx ; SWAPING

mov array[esi], eax ; SWAPING

pop step

pop ecx

inc step

Loop outerLoop

main endp

end main

|  |
| --- |
| **Q3: Q3: Solve the following question. [Marks: 5+5 = 10]**   1. **Write an assembly language program that converts the following 32-bit hexadecimal number**   **12438765h to 87654321h using shift and rotate instructions.** |

**[Solution]**

Include Irvine32.inc

.code

main PROC

mov eax, 12438765h

rol eax, 16

rol ax, 8

rol al, 4

call dumpregs

exit

main ENDP

End main

|  |
| --- |
| 1. **Given that EAX = 0Eh, ECX = 17h, EDX = 02h, and ESP = 0000 011Eh, draw out the run-time stack (diagrams), with addresses after each numbered (a, b and c) instruction. No points will be awarded if addresses are found missing/wrong.** |

**[Solution]**

**main PROC**

**SUB AL, 1** ;EAX = 0000 00**0D**h

**INC DH** ;EDX = 0000 **01**02h

**PUSH EAX ;a** ;[0000 011A] = 0000 000D

**SHL CL,2 ;**ECX = 0000 **005C**

**PUSH ECX ;b** ;[0000 0116] = 0000 005C

**ROR DL,1**

**PUSH EDX ;c** ;[0000 0112] = 0000 0001

**POP EDX**

**POP ECX**

**POP EAX**

**main ENDP**