**Assignment**

**COEN 311**

**Computer Organization and Software**

**Assignment #5**

Sunil Kublalsingh

ID: 40212432

Professor: Anjali Agarwal

“I certify that this submission is my original work and meets the

Faculty’s Expectations of Originality”

Sunil Kublalsingh

Student ID: 40212432



April 13, 2022

# Question 1

Text

Description automatically generated with low confidence

mult:

; mult performs repeated addition so we get unsigned multiplication of n1 and n2

mov cl, [n2] ; place n2 as the loop counter

mov bl, [n1] ; place n1 as the loop addition portion

loop:

add al, bl ; al=al+bl

dec cl ; cl=cl-1

jnz loop ; we continue the loop only if the zero flag wasn't set yet

mov word [prod], ax ; if we didn't jump, then our final answer is in ax so we place it in prod

ret ; exit the subroutine and pop ip off the stack

section .data

n1 db 6

n2 db 3

section .bss

prod resw 1

section .text

global \_start

\_start:

call mult

\_exit:

mov eax,1

mov ebx,0

int 80h

# Question 2

Text

Description automatically generated

**a)**

mean:

mov bx, [esp+8] ; get the number of items in the array (16 bits of data)

mov ecx, ebx ; cx will be our counter so we know when to stop the loop

mov ebp, [esp+4] ; get the address of the array (32 bits/4bytes of data)

mov eax, 0 ; use ax to store the sum

mov si, 0 ; the starting index

loop:

add ax, [ebp+esi] ; get add the element in the array in ax

add esi, 2 ; increment si twice cuz its a word so we get our next element

dec ecx ; decrement ecx

jnz loop ; if ecx is not zero yet, continue the loop

idiv bx ; get the average

push ax ; place the average on the stack

add esp, 2 ; add 2 to the stack pointer so we can return properly

ret ; exit the subroutine and pop ip off the stack

**b)**

variance:

mov bx, [esp+4] ; the mean

mov ebp, [esp+6] ; base address of array

mov cx, [esp+10] ; array size

mov esi, 0

mov dx, 0 ; use dx as the sum

push dx ; but place it on the stack instead (16 bits)

loop2:

mov ax, [ebp+esi] ; place the first element in ax

sub ax, bx ; subtract mean from element

imul ax ; square the result

add [esp], ax ; add to sum (in stack)

add esi, 2 ; increment the index by 2 since its a word sized array

dec cx ; decrease the counter

jnz loop2 ; loop again if the counter is not 0 yet

pop ax ; place the result in ax

idiv bx ; divide result by array size to get variance

push ax ; place our result back on the stack

add sp, 2 ; add 2 to our stack pointer so we can return properly

ret

**c)**

section .data

list dw 2,3,5,1,6 ; the list

array\_size dw 5 ; we need to pass the array size since the array is size n

section .bss

prod resw 1

section .text

global \_start

\_start:

push word [array\_size] ; place the array size in the stack

push list ; place the list on the stack

call mean ; call the subroutine (also pushes the IP onto the stack)

push word [esp-6] ; push the mean back to the top of the stack

call variance ; call the variance subroutine

mov ax, [esp-6] ; get our variance from the stack

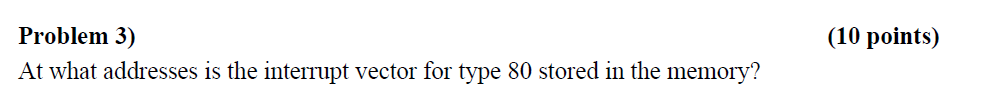
\_exit:

mov eax,1

mov ebx,0

int 80h

# Question 3



Multiply by 4 because each location is 4 bytes.

80 \* 4 = 320

Convert to hex

320 = 2^8 + 2^6 = 0000 0001 0100 0000

**IP=$0140**

**CS**=0140+2 = **$0142**

# Question 4

Text

Description automatically generated

section .text

global \_start

\_start:

mov ah, 0x02 ;print character to standard output

mov cl, FF ; the endpoint is when its FF (256)

mov dl, '0x00' ; place first character in dl

loop:

int 0x21 ; print the character

inc dl

cmp dl, cl

jne loop

\_exit:

mov eax, 1

mov ebx, 0

int 80h