Assignment

COEN 311

Computer Organization and Software

Assignment #5

Sunil Kublalsingh

ID: 40212432

Professor: Anjali Agarwal

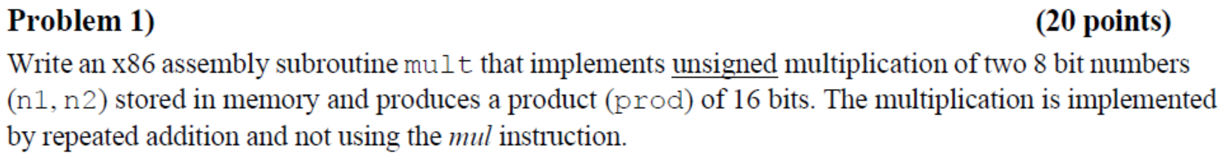
“I certify that this submission is my original work and meets the Faculty’s Expectations of Originality”

Sunil Kublalsingh
Student ID: 40212432

April 13, 2022

Sunil Kublalsingh
Student ID: 40212432
Course: COEN 311 W – Computer Organization and Software
Professor: Anjali Agarwal

Question 1



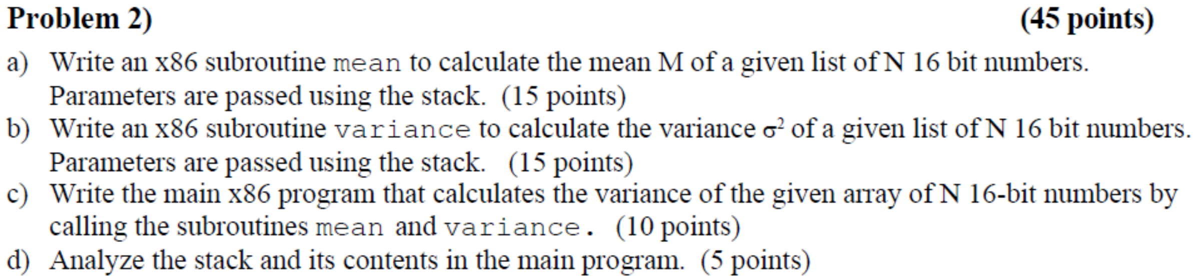
Problem 1) **(20 points)**

Write an x86 assembly subroutine `mult` that implements unsigned multiplication of two 8 bit numbers (`n1, n2`) stored in memory and produces a product (`prod`) of 16 bits. The multiplication is implemented by repeated addition and not using the *mul* instruction.

```
 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
```

Sunil Kublalsingh
Student ID: 40212432
Course: COEN 311 W – Computer Organization and Software
Professor: Anjali Agarwal

Question 2



Problem 2) **(45 points)**

a) Write an x86 subroutine `mean` to calculate the mean M of a given list of N 16 bit numbers. Parameters are passed using the stack. (15 points)

b) Write an x86 subroutine `variance` to calculate the variance σ^2 of a given list of N 16 bit numbers. Parameters are passed using the stack. (15 points)

c) Write the main x86 program that calculates the variance of the given array of N 16-bit numbers by calling the subroutines `mean` and `variance.` (10 points)

d) Analyze the stack and its contents in the main program. (5 points)

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)

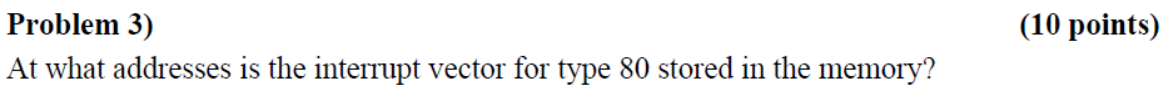
```
variance:
   mov bx, [esp+4] ; the mean
   mov ebp, [esp+6] ; base address of array
   mov cx, [esp+10] ; array size
   mov esi, 0
   mov dx, 0 ; use dx as the sum
   push dx ; but place it on the stack instead (16 bits)

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



Problem 3) **(10 points)**

At what addresses is the interrupt vector for type 80 stored in the memory?

Multiply by 4 because each location is 4 bytes.

80 * 4 = 320

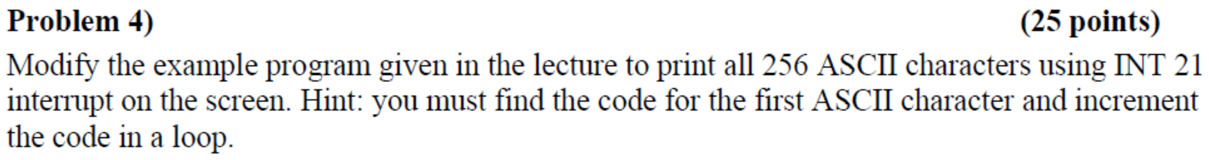
Convert to hex

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

IP=$0140

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

Question 4



Problem 4) **(25 points)**

Modify the example program given in the lecture to print all 256 ASCII characters using INT 21 interrupt on the screen. Hint: you must find the code for the first ASCII character and increment the code in a loop.

```
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
```