Nikhil Chakravarthy

10/17/2019

EECS 349

HW 2

Q2)

```
DATA   SEGMENT
X  DB  25
Y  DB  32
Z  DW  ?
DATA  ENDS

CODE  SEGMENT
        ASSUME  CS:COSE,DS:DATA
START:
        MOV AX,DATA                     ; data to ax
        MOV DS,AX               ; ax (which is data) to ds
        MOV AL,X                ; x to al
        MUL AL                          ; ax = al*al = x*x
        MOV BX,0                ; bx=0
        MOV BL,Y                ; bl=y
        ADD BL,BL               ; bl = bl+bl = 2y
        ADC BH,0                ; bh=bh+0+(CF value)
        ADD BL,Y                ; bl = bl+y=3y
        ADC BH,0                ; bh = bh+0=0
        SUB AX,BX               ; ax = ax-bx = x*x-bx
        SHR AX,1                ; shift ax right 1 bit, (ax div by 2)
        MOV Z,AX            ; z = ax, finished, z = ax = (x*x - bx)/2, bx = ?
        MOV AH,4CH
        INT 21H                        ; sys exit, result is z=(x*x - bx)/2
CODE  ENDS
        END  START
```

Q3) Bubble Sort in Assembly

```
.data
array dd 34, 12, 3, 18
szMsg db "%d",0ah,0

.code
start:
   mov ecx,3   ; set ecx to 3
L2:
   push ecx    ; push ecx to stack (stack has (,3) )
   xor esi, esi    ; zero esi
   mov ecx,3   ; set ecx to 3

L0:
   mov ebx,array[esi]  ; ebx is array at esi
   cmp ebx,array[esi 4]    ;compare ebx (array at esi), and array at esi 4 shifted (next value)
   jb L1   ; go to L1 if ebx (array at esi) < array at esi 4 shifted
```

```
    xchg ebx,array[esi 4]   ; exchange values of ebx and array at esi 4 shifted
    mov array[esi],ebx  ; moves the current value of ebx (array esi 4) to array esi, making this plus the line above a swap
    add esi,4   ; shift esi value 4 (next value in array)

L1:
    loop L0     ; jump to L0
    pop ecx     ; pop stack into ecx
    loop L2     ; jump to L2
    xor esi,esi ; reset esi
    mov ecx,4   ; set ecx to 4

L3:
    push ecx    ; push ecx to stack
    invoke printf,offset szMsg,array[esi]   ; print at array esi
    add esi,4   ; shift esi
    pop ecx     ; pop off stack to ecx
    loop L3     ; recurse on L3
    ret         ; return when array fully printed
end start
```