Comp 3350: Computer Organization & Assembly Language

HW # 8. Theme: Integer Arithmetic

All main questions carry equal weight.

Points will be awarded to only those answers which have work clearly shown

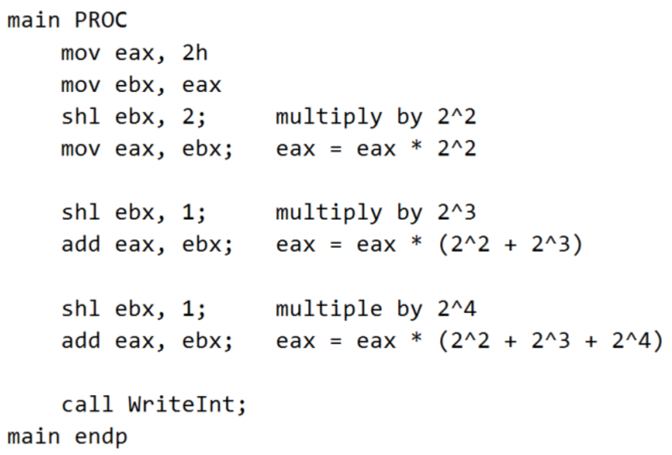
1. In the following code sequence, show the value of AL after each shift or rotate instruction has executed. This question is to be done by hand, not by running a program. `5 points * 4`

```
mov cl, 1;
mov al, 12h;                        al = 12h = 0001 0010
rol al, cl;                         al = 0010 0100 = 24h

mov al, 34h;                        al = 34h = 0011 0100
mov cl, 2
ror al, Cl;                         al = 0000 1101 = 0dh
stc
mov al, 56h;                        al = 56h = 0101 0110, CF = 1
mov cl, 3
rcl al, cl;                         al = 1011 0101 = b5h
stc
mov al, 78h;                        al = 0111 1000 = 78h, CF = 1
mov cl, 1
rcr al, cl;                         al = 1011 1100 = bch
```

2. (a) Write a program which calculates EAX**28_{10}* using binary multiplication. (Only typewritten code required) 10 points

EAX × 28_{10} = EAX × (2_2 + 2_3+2_4)



```
main PROC
    mov eax, 2h
    mov ebx, eax
    shl ebx, 2;     multiply by 2^2
    mov eax, ebx;   eax = eax * 2^2

    shl ebx, 1;     multiply by 2^3
    add eax, ebx;   eax = eax * (2^2 + 2^3)

    shl ebx, 1;     multiple by 2^4
    add eax, ebx;   eax = eax * (2^2 + 2^3 + 2^4)

    call WriteInt;
main endp
```

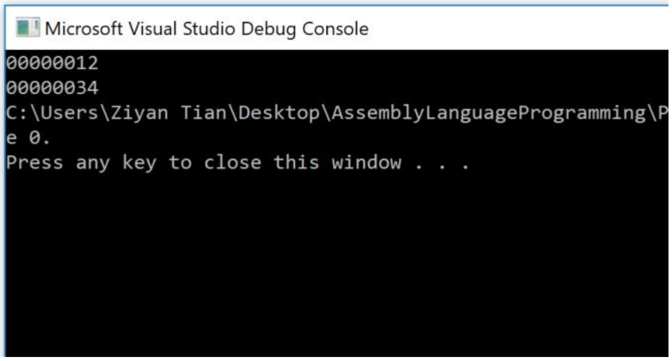
(b) Consider the following unsigned value: 1234ABCD. Let this value be stored in register EAX. Write a program that will extract the decimal digits from this value using shifts and logical instructions. Place the first two **decimal numeric** digits in DH and the other two into DL. Submit a screenshot of **the console output** of the program and the **asm/lst file**. 10 points (files: 5 points, screenshot 5 points)

```
.code
main PROC
        mov eax, 1234ABCDh
        shr eax, 16

        mov edx, 0
        mov edx, eax            ; DH = 12h, DL = 34h

        mov eax, edx            ; DH
        and eax, 0000FF00h
        shr eax, 8
        call WriteHex
        call crlf
        mov eax, edx            ; DL
        and eax, 000000FFh
        call WriteHex

invoke ExitProcess, 0
main endp
end main
```



```
00000012
00000034
C:\Users\Ziyan Tian\Desktop\AssemblyLanguageProgramming\P
e 0.
Press any key to close this window . . .
```

3. (a) What will be the contents of AX and DX after the following operation? What may happen if you do not set dx to 0 in the beginning? You must work this problem by hand, not by a program run.

```
mov dx, 0
mov ax, 1111h
mov cx, 2222h
mul cx

DX: AX = ax × cx = 1111h × 2222h = 246:8642 h
DX = 246h;   AX = 8642h                               1.5 points * 2
```

(b) When does an IDIV instruction cause an overflow? Provide an example.

2 points * 2

An IDIV instruction causes an overflow when the quotient is too big.

Example:

```
mov  eax,  ffffh
mov  bx,  1
idiv   bx
```

(c) What will be the values of DX:AX after the following instructions execute? What might be the use of such a sequence of instructions in a 16-bit computer? 1.5 points * 2

```
mov ax, 0h
mov dx, 0h
sub ax, 2h
sbb dx, 0

AX = 0h – 2h = FFFEh, CF = 1
DX = 0h – 0h – 1 = FFFFh
```

4. Write a program which will have the register ‘AX’ as its input. It would display the contents of AX in binary on the screen. You should use shifts to achieve the function. Demonstrate by using several values of AX.

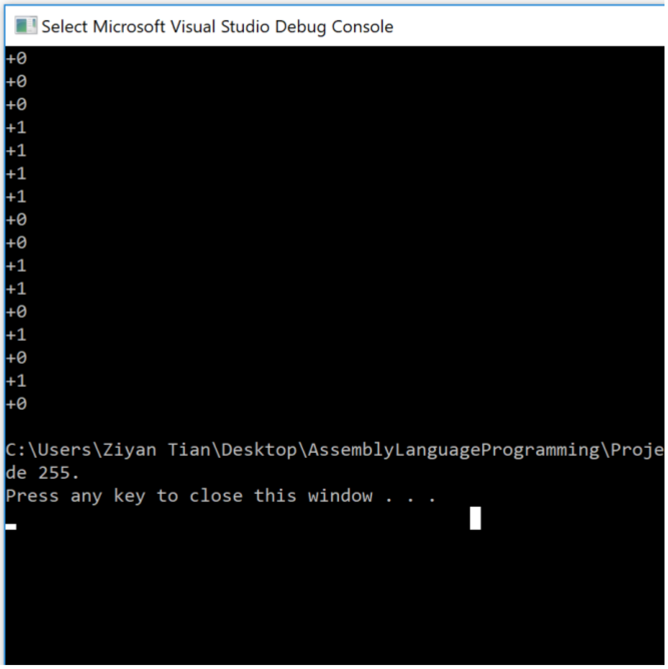
code:15 points screen shot: 5 points

```
.code
main PROC
    mov eax, 0
    mov ebx, 0
    mov ecx, 0
    mov edx, 0
    mov eax, 5678h

    call displayAX
invoke ExitProcess, 0
main endp

displayAX PROC
    mov ebx, eax
    mov ecx, 16
    mov edx, ebx
L1: mov edx, ebx
    and edx, 1h
    mov eax, edx
    call WriteInt
    call crlf
    shr ebx, 1
    loop L1
displayAX endp

end main
```



```
+0
+0
+0
+1
+1
+1
+1
+0
+0
+1
+1
+0
+1
+0
+1
+0

C:\Users\Ziyan Tian\Desktop\AssemblyLanguageProgramming\Proje
de 255.
Press any key to close this window . . .
```

5. Write a program that performs C = A + B using extended addition. See textbook pg. 270-271.
Use the following:

```
Apple         QWORD  1111222233334444h
Berry         QWORD  13572468ABCD0000h
Cherry QWORD  ?
```

You may only use16-bit registers to perform the addition, e.g. AX, BX etc.

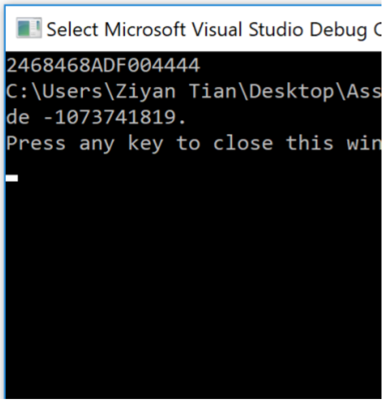
Submit the asm/list file and a screenshot of your code printing the contents of all the arrays after the run.

code:10 points screen shot: 5 points files: 5 points

```
.code
main PROC
    mov esi,OFFSET Apple
    mov edi,OFFSET Berry
    mov ebx,OFFSET Cherry
    mov ecx,TYPE Apple      ; LENGTHOF -> TYPE
    call Extended_Add

    mov esi,OFFSET Cherry
    mov ecx,TYPE Cherry     ; LENGTHOF -> TYPE
    call Display_Sum
    call crlf

invoke ExitProcess, 0
main ENDP
```



```
2468468ADF004444
C:\Users\Ziyan Tian\Desktop\Ass
de -1073741819.
Press any key to close this win
```