Comp 3350: Computer Organization & Assembly Language
HW # 8: Theme: Integer Arithmetic

All main questions carry equal weight.
Points 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. ***Credit will NOT be given for answers that do not show work.***

```
mov al, 32h
rol al, 3           ;      A: al = 91h
mov al, D2h
ror al, 1           ;      B: al = 69h
stc
mov al, 3Bh
rcl al, 2           ;      C: al = 0EEh
stc
mov al, 17h
rcr al, 1           ;      D: al = 8Bh
```

A).

00110010	
01100100	CF=0
11001000	CF=0
10010001	CF=1

B).

11010010	
01101001	CF=0

C).

00111011	CF=1
01110111	CF=0
11101110	CF=0

D).

00010111	CF=1
10001011	CF=1

2. Write instructions that calculate EAX*23 using binary multiplication.

```
mov eax, 1h
push eax
mov ebx, eax
shl eax, 4
shl ebx, 2
add eax, ebx
pop ebx
shl ebx, 1
add eax, ebx
inc eax
```

3. The time stamp of a file uses bits 0-2 for hours, bits 3-7 for days, bits 8-11 for months, and bits 12-15 for years. Write instructions that extract the months and copy the value to a byte variable named "months".

```
TITLE Extract months
INCLUDE Irvine32.inc
.data
months BYTE ?
.code
main PROC
mov eax, 0110110001001011b
mov bh, ah
and bh, 00001111b
mov months, bh
exit
main ENDP
end main
```

COMP 3350 WALTER CONWAY

```
        3 4 5 6
      * 2 3 0 0
      ---------
        0 0 0 0
      1 0 0 0 0 0
    1 9 D 0 2 0 0
+   6 8 A C 0 0 0
    -------------
    7 2 7 C 2 0 0
```

4. What will be the contents of AX and DX after the following operation?

```
mov dx, 0
mov ax, 3456h
mov cx, 2300h
mul cx
```

5. Implement the following expression in assembly language, using 32-bit unsigned operands:

Alpha = (Alpha + Beta) * (Gamma - Delta) / Iota

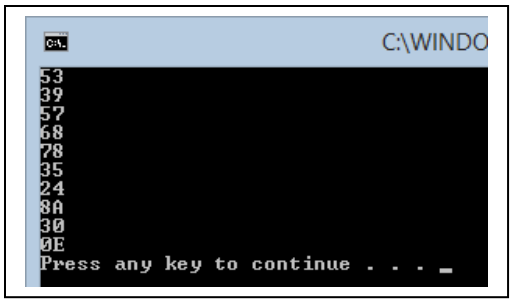
```
.data
alpha DWORD 2
beta DWORD 3
gamma DWORD 5
delta DWORD 1
iota DWORD 4
.code
main proc
mov eax, alpha
add eax, beta
push eax
mov ebx, gamma
sub ebx, delta
mov eax, ebx
mov edx,0
div iota
pop ebx
mul ebx
```

6. 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?

```
mov ax, 1h
mov dx, 0h
sub ax, 1h
sbb dx, 0
```

DX:AX = 0:0
What might be the use of such a sequence of instructions in a 16-bit computer?
Subtracting numbers that are larger than 16 bits.

7. Write a program that adds two operands using extended addition. See textbook pg 257.
Use the following:
Blue BYTE 12h, 14h, 34h, 23h, 65h. 78h, 56h, 99h, 20h, 05h
Yellow BYTE 41h, 25h, 23h, 45h, 12h, BCh, CDh, F1h, 10h, 09h
Green BYTE 10 dup(0)
Display the contents of the array Green.



```
TITLE Using extended addition to add blu and ye together in gr
INCLUDE Irvine32.inc
.data
Blu BYTE 12h, 14h, 34h, 23h, 65h, 78h, 56h, 99h, 20h, 05h
Ye BYTE 41h, 25h, 23h, 45h, 12h, 0BCh, 0CDh, 0F1h, 10h, 09h
gr BYTE 10 dup(0)
.code
main PROC
clc
mov ecx, 11
mov edx, 10
mov esi, OFFSET blu
mov edi, OFFSET ye
mov ebx, OFFSET gr

L1:
       mov al, [esi + edx]
       mov ah, [edi + edx]
       adc al, ah
       mov byte ptr [ebx + edx],al
       dec edx

loop l1

mov esi, offset gr
mov ecx, 10
mov ebx, type byte
l2:
       mov eax, [esi]
       call writehexb
       call crlf
       inc esi
loop l2

exit
main ENDP
END main
```

Blue
Yellow
Green
Are reserve words in irvine32.inc