Comp 3350: Computer Organization & Assembly Language
HW # 9: Theme: Advanced Procedures, Stack Parameters, Locals and BCD

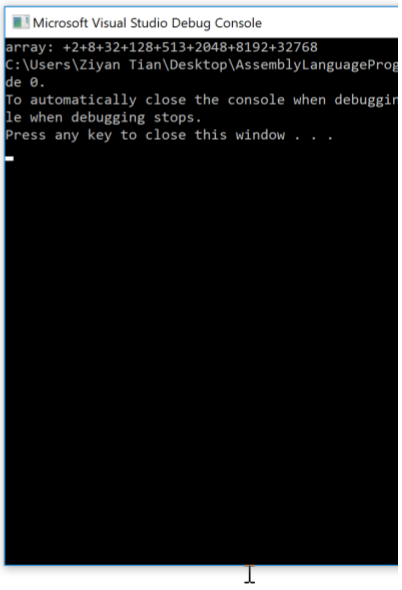
(All main questions carry equal weight. Credit awarded to only those answers for which work has been shown.)

---

1. Write a procedure named *Geometric Progression* that fills an array of eight (8) numbers with the Geometric series. The procedure receives three arguments: the first is the offset of an array, the second is the first term and the third is the ratio. The first argument is passed by value and the others by reference. In the main program, you should set the parameters and print the series. Please run your program with several different first term and ratios.

   Please embed your code into your homework solution along with a screen shot of the run of the program. code 25 points screenshot 8 points

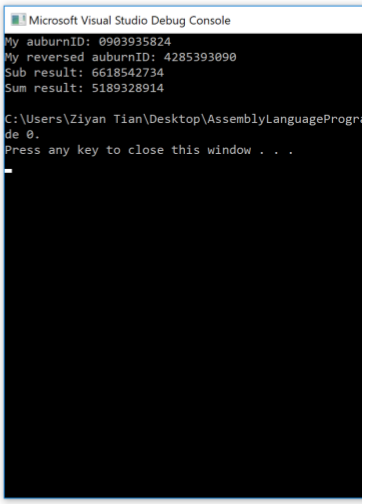
```
.data
str1        BYTE "array: ", 0
firstTerm   DWORD   4
ratio       DWORD   2
arr         DWORD ?

.code
main PROC
    push OFFSET arr
    push OFFSET firstTerm
    push OFFSET ratio
    mov edx, OFFSET str1
    call WriteString
    call GP

    mov ecx, 8
    mov esi, OFFSET arr
L1: mov eax, [esi]
    call WriteInt
    add esi, 4
    loop L1

invoke ExitProcess, 0
main endp

GP PROC
    push ebp
    mov ebp, esp
    mov esi, [ebp + 8]      ; esi = address of ratio
    mov eax, [esi]          ; eax = ratio
    mov esi, [ebp + 12]     ; esi = address of firstTerm
    mov ebx, [esi]          ; ebx = firstTerm
    mov esi, [ebp + 16]         ; esi -> arr[0]
    ;pop ebx        ; ratio
    ;pop eax        ; firstTerm
    ;pop esi        ; offset of the array

    mov ecx, 8

    mov [esi], eax

L2: add esi, 4
    mul ebx
    mov [esi], eax
    loop L2

    pop ebp
    ret
GP endp

end main
```



2. Draft a program that subtracts one BCD number from another (10-digits each). The first BCD number is stored in an array named *myAuburnID*, and the second in an array named *myAurbunIdRev*. The first number is your actual Auburn ID (with a prefix single zero digit and the remaining digits as the 9-digits of your *Auburn ID*); the second is the value of *MyAuburnId* written backwards. Your program should do the following: code 25 points, screenshot 8 points
   1) Use shifts/rotates using *myAuburnID* to fill the array *myAuburnIdRev* 5 points
   2) Display contents of the memory locations in question 5 points
   3) Subtract *myAuburnIDRev from myAurbunId* using BCD arithmetic 5 points
   4) Store the sum in a variable named *Result*, and 5 points
   5) Display contents of memory post execution. 5 points

   Please embed your code into your homework solution along with a screen shot post execution.

```
.data
str1            BYTE "My auburnID: ",0
str2            BYTE "My reversed auburnID: ",0
str3            BYTE "Sub result: ", 0
str4            BYTE "Sum result: ", 0
myAuburnID      BYTE 09h, 03h, 93h, 58h, 24h
myAurbunIdRev   BYTE 10 DUP (?)
tmp             BYTE 10 DUP (?)
subResult       BYTE 10 DUP (?)
addResult       BYTE 10 DUP (?)

.code
main PROC

    ;1) Use shifts/rotates using myAuburnID to fill the array myAuburnIdRev
    mov ecx, 5
    mov eax, OFFSET myAuburnID
    mov ebx, OFFSET myAurbunIdRev

L1: mov dl, BYTE PTR [eax + ecx - 1]
    ror dl, 4
    mov [ebx], dl
    add ebx, 1
    loop L1

    ;2) Display contents of the memory locations in question
    mov edx, OFFSET str1
    call WriteString
    mov ecx, 5
    mov edx, OFFSET myAuburnID
    call printArray

    mov edx, OFFSET str2
    call WriteString
    mov ecx, 5
    mov edx, OFFSET myAurbunIdRev
    call printArray

    ;3) Subtract myAuburnIDRev from myAurbunId using BCD arithmetic
    call caltheTensComplement
    push OFFSET myAuburnId
    push OFFSET tmp
    push OFFSET subResult
    call calSumBCD

    mov edx, OFFSET str3
    call WriteString
    mov ecx, 5
    mov edx, OFFSET subResult
    call printArray

    ;4) Store the sum in a variable named Result, and
    push OFFSET myAuburnId
    push OFFSET myAurbunIdRev
    push OFFSET addResult
    call calSumBCD

    ;5) Display contents of memory post execution.
    mov edx, OFFSET str4
    call WriteString
    mov ecx, 5
    mov edx, OFFSET addResult
    call printArray

invoke ExitProcess, 0
main endp
```



```
caltheTensComplement PROC
    mov ecx, 5
    mov esi, OFFSET myAurbunIdRev
    mov edi, OFFSET tmp

L3: mov al, BYTE PTR [esi]
    mov bl, al
    and bl, 0Fh
    mov bh, 09h
    sub bh, bl
    mov dl, bh

    mov bl, al
    and bl, 0F0h
    mov bh, 090h
    sub bh, bl
    add dl, bh

    mov [edi], dl
    add esi, 1
    add edi, 1
    loop L3

    mov esi, OFFSET tmp
    add esi, 4
    mov ah, 1   ; carry

L4: cmp ah, 0
    jz  L9
    mov dx, 0h
    mov al, BYTE PTR [esi]
    mov bl, al
    and bl, 0Fh
    cmp ah, 01h
    jnz L6
    cmp bl, 09h
    jnz L5
    mov bl, 00h
    mov ah, 01h
    jmp L6
L5: add bl, 01h
    mov ah, 00h

L6: mov dl, bl
    mov bl, al
    and bl, 0F0h
    cmp ah, 01h
    jnz L8
    cmp bl, 090h
    jnz L7
    mov bl, 00h
    mov ah, 01h
    jmp L8
L7: add bl, 010h
    mov ah, 00h

L8: add dl, bl
    mov [esi], dl
    sub esi, 1
    loop L4

L9:     ret
caltheTensComplement ENDP
```

```
    calSumBCD PROC
        push ebp
        mov ebp, esp
        mov ecx, 5
        mov esi, [esp + 16]     ; BCD1
        mov edi, [esp + 12]     ; BCD2
        mov ebx, [esp + 8]      ; result = BCD1 + BCD2
        add esi, 4
        add edi, 4
        add ebx, 4
        push ebx
        mov ah, 0   ; carry
        mov bh, 0

    L10: mov ebx, 0
         mov al, BYTE PTR [esi]
         mov bl, BYTE PTR [edi]
         mov dl, al
         and dl, 0Fh
         mov dh, bl
         and dh, 0Fh
         mov bh, ah
         add bh, dl
         add bh, dh
         jc L15
         cmp bh, 0Ah
         jb L11
    L15: sub bh, 0Ah
         mov ah, 010h
         jmp L12
    L11: mov ah, 00h
    L12: mov al, BYTE PTR [esi]
         mov bl, BYTE PTR [edi]
         mov dl, al
         and dl, 0F0h
         mov dh, bl
         and dh, 0F0h
         add dh, ah
         add dh, dl
         jc L16
         cmp dh, 0A0h
         jb L13
    L16: sub dh, 0A0h
         mov ah, 01h
         jmp L14
    L13: mov ah, 00h
    L14: shr bx, 8
         add dh, bl
         pop ebx
         mov [ebx], dh
         sub ebx, 1
         push ebx
         sub esi, 1
         sub edi, 1
         loop L10

         pop ebx
         pop ebp
         ret
    calSumBCD ENDP
printArray PROC
L2: mov eax, [edx]
    mov ebx, 1
    add edx, 1
    call WriteHexB
    loop L2

    call crlf
    ret
printArray ENDP
end main
```

3. Consider an isosceles triangle A with base 8 and height 14. Consider another triangle B formed using vertices which are the center of the sides of triangle A. Consider another triangle C whose vertices are similarly formed from B. Repeat this process ad infinitum. Express the sum of the areas of all such triangles using a series and its closed form sum. Compute the areas *(a)* by using only the first two terms of the series and *(b)* by using the closed form of the series sum. Write a program to find the sums and use shifts to compute. What is the difference in the two computed sums? code 25 points, screenshot 8 points

   Please embed your code into your homework solution along with a screen shot post execution.

$$sum_{\infty} = \frac{8 \times 14}{2} \times \left(1 + \frac{1}{4} + \frac{1}{4^2} + \cdots + \frac{1}{4^n}\right)$$
$$= 56 \times \left(1 \times \frac{1 - \frac{1}{4^n}}{1 - \frac{1}{4}}\right), when\ n \to \infty, 1 - \frac{1}{4^n} \to 1$$
$$= 56 \times \frac{4}{3} = 74.666666667$$

   using the series, n = 2

$$sum = 56 \times \left(1 + \frac{1}{4}\right) = 70$$

   using its closed form sum

$$sum = 56 \times \tfrac{4}{3} = 74.666666667$$

```
.data
str1    BYTE    "using the series: sum = ", 0
str2    BYTE    "using the closed form: sum = ", 0
str3    BYTE    "the difference = ", 0
base    BYTE    8
height  BYTE    14
ans1    BYTE    ?
ans2    BYTE    ?

.code
main PROC
    mov ebx, 0
    mov bl, height
    shl bl, 3
    shr ebx, 1

; 1) using the series
    mov eax, ebx
    shr eax, 2
    add eax, ebx
    mov ans1, al

    mov edx, OFFSET str1
    call WriteString
    call WriteDec
    call crlf

; 2) using its closed form sum
    mov edx, 0
    mov eax, ebx
    shl eax, 2
    mov ebx, 3
    idiv ebx
    mov ans2, al

    mov edx, OFFSET str2
    call WriteString
    call WriteDec
    call crlf

;3) the difference
    mov bl, ans1
    sub al, bl

    mov edx, OFFSET str3
    call WriteString
    call WriteDec
    call crlf

invoke ExitProcess, 0
main endp
end main
```