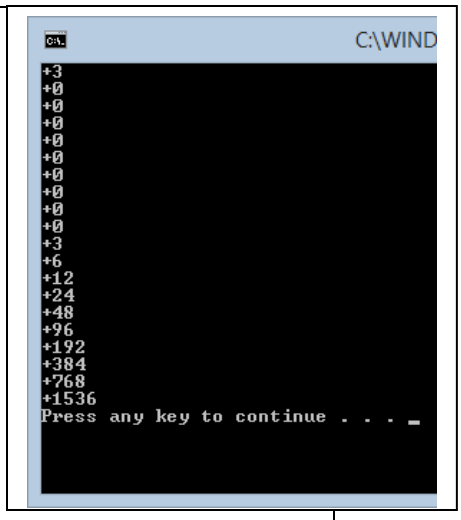
Comp 3350: Computer Organization & Assembly Language
HW # 9: Theme: Advanced Procedures, Stack Parameters, Locals and BCD

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

1. Write a procedure named *ArrayFill* that fills an array of ten (10) numbers with the geometric progression 3, 6, 12, 24, … etc. You must only use the first two integers in the series to generate the series. The procedure receives two arguments: the first is the offset of an array, and the second is an integer that specifies the array length. The first argument is passed by value and the second is passed by reference. In the main program, you should set the parameters of the array and print the array values before and after call to the procedure.

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



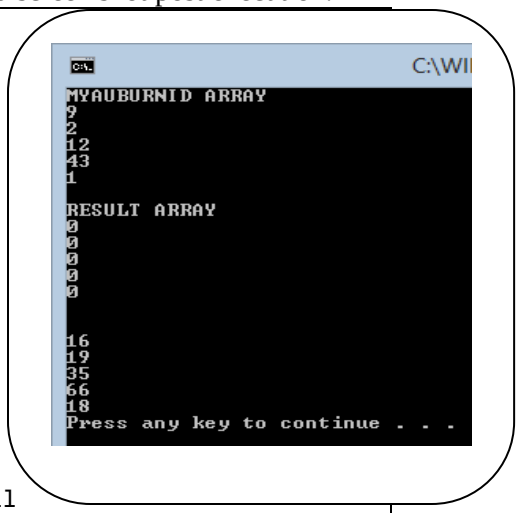

```
TITLE ARRAYFILL
INCLUDE Irvine32.inc

.data
count = 10
;A0=3
;A1=6
;geometricArray WORD 3,6,12,24,48,96,192,384,768,1536
geometricArray WORD 3 , 9 DUP(0)
.code

main PROC
push OFFSET geometricArray ; Argument 2
push count ; Argument 1
CALL DisplayArray ; Displaying the array before ArrayFill

push OFFSET geometricArray ; Argument 2
push count ; Argument 1
CALL ArrayFill ; generate geometric progression and filling that progression in
the array

push OFFSET geometricArray ; Argument 2
push count ; Argument 1
CALL DisplayArray ; Displaying the array after ArrayFill
exit
main ENDP

;============================
; Needs two argument one Array offset and two the Array Count
; First is a call by reference, the second is call by value
;============================
ArrayFill PROC
push ebp
mov ebp, esp
mov esi, [ebp+12] ; address offset
mov ecx, [ebp+8] ; loop counter
mov ebx, 0 ; index
L2:
mov eax, [esi + ebx]
shl eax,1
add ebx, TYPE geometricArray
```

```
mov [esi + ebx], eax
loop L2
pop ebp
ret
ArrayFill ENDP

;============================
; Needs two argument one Array offset and two the Array Count
; First is a call by reference, the second is call by value
;============================
DisplayArray PROC
push ebp
mov ebp, esp
mov esi, [ebp+12] ; address offset
mov ecx, [ebp+8] ; loop counter
mov ebx, 0 ; index
L1:
movsx eax, WORD PTR [esi + ebx]
add ebx, TYPE geometricArray
CALL WriteInt
CALL crlf
Loop L1
pop ebp
ret 8
DisplayArray ENDP

END main
```

2. Draft a program that adds two BCD numbers (9-digits each). The first BCD number is stored in an array named *myAuburnID*, and the second in an array named *All1s*. The first number is your actual Auburn ID number; the second is 111111111h. Your program should do the following:
   1) Display contents of the memory before program execution,
   2) Add *myAuburnID and All1s.*
   3) Store the sum in a variable named *Result*, and;
   4) Display contents of memory post execution.

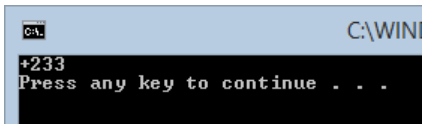
   Please embed your code into your homework solution along with a screen shot post execution.

```
TITLE BCD
INCLUDE Irvine32.inc
.data
myAuburnID BYTE 09,02,12,43,01
ALL1s BYTE 01h,11h,11h,11h,11h
Result BYTE 5 DUP(0)
MSG1 BYTE "RESULT ARRAY",0
MSG2 BYTE "MYAUBURNID ARRAY",0
.code
main PROC
mov edx, OFFSET MSG2
CALL WriteString
mov edx,0
CALL CRLF
push OFFSET myAuburnID ; Argument 2
push 5 ; Argument 1
CALL DisplayArray ; Displaying the array before ArrayFill
CALL crlf

mov edx, OFFSET MSG1
CALL WriteString
mov edx,0
CALL CRLF

push OFFSET Result ; Argument 2
push 5 ; Argument 1
CALL DisplayArray ; Displaying the array before ArrayFill

CALL crlf
mov ecx, 6
mov edx, 5
mov esi, OFFSET myAuburnID
mov edi, OFFSET ALL1s
mov ebx, OFFSET Result

L1:
       mov al, [esi + edx]
       mov ah, [edi + edx]
       add al, ah
       daa
       mov byte ptr [ebx + edx],al
       dec edx
loop l1

CALL CRLF
push OFFSET Result ; Argument 2
push 5 ; Argument 1
```



```
CALL DisplayArray ; Displaying the array before ArrayFill
exit
main ENDP

;============================
; Needs two argument one Array offset and two the Array Count
; First is a call by reference, the second is call by value
;============================
DisplayArray PROC
push ebp
mov ebp, esp
mov esi, [ebp+12] ; address offset
mov ecx, [ebp+8] ; loop counter
mov ebx, 0 ; index
L1:
movzx eax, BYTE PTR [esi + ebx]
add ebx, TYPE BYTE
CALL WriteDec
CALL Crlf
Loop L1
pop ebp
ret 8
DisplayArray ENDP

END main
```

3. Draft a procedure called *AddThree* that is similar to the *AddTwo* program within the slides (at or about #30). *AddThree* receives three parameters *x*, *y* and *z* from the stack and outputs a value equal to x + 2*y + z. In order to receive credit, your submission must do the following:
   1) Pass parameters *x*, *y*, and *z* by value on the stack using the push instruction. Use *x= 102*, *y = 54* and *z = 23*,
   2) *AddThree* must get its inputs from the stack,
   3) Call *AddThree* and print the result to the screen.




```
TITLE ADD THREE
INCLUDE Irvine32.inc
.data
xval DWORD 102
yval DWORD 54
zval DWORD 23
.code
main PROC
push zval
push yval
push xval
CALL addThree
exit
main ENDP
;=====================================
;Add Three
;Takes three parameters x,y,z
;=====================================
addThree PROC
push ebp
mov ebp, esp
mov eax, [ebp + 12]
shl eax, 1
add eax, [ebp + 8 ]
add eax, [ebp + 16]
pop ebp
CALL WriteInt
CALL crlf
ret 12
addThree ENDP
END main
```