Part 1 - Short Answers [3 points]

1. What will be the value of EAX after each of the following instructions execute?

```
mov eax, TYPE myString
                                      ;a. 00000001h
mov eax, SIZEOF myBytes
                                      :b. 00000005h
mov eax, LENGTHOF myBytes
                                      :c. 00000005h
mov eax, LENGTHOF myWords
                                      ;d. 00000007h
mov eax, SIZEOF myWords
                                      ;e. 0000000Eh
mov eax, SIZEOF myString
                                      ;f. 0000000Dh
mov eax, SIZEOF myDWords
                                      ;g. 0000000Ch
mov eax, LENGTHOF myDWords
                                      ;h. 00000003h
```

2. Write a single instruction that moves the last two bytes in myBytes to the CX register. What will be the resulting value of CX?

```
mov cx, WORD PTR [myBytes+3]
```

3. Write an instruction that moves the fourth byte in myWords to AL register.

```
mov al, BYTE PTR [myWords+3]
```

Part2 – Programming Tasks [10 points]

4. Fibonacci Numbers – With the use of a loop, write an assembly program that calculates the first nine values of the Fibonacci number sequence and put the result in eax register. The Fibonacci number sequence is described by the formula:

```
Fib(1) = 1, Fib(2)=1, Fib(n) = Fib(n-1) + Fib(n-2)
```

You will get this sequence for the first nine values: 1, 1, 2, 3, 5, 8, 13, 21, 34. The result in eax needs to be 34.

```
.data
       current DWORD 1h
       previous DWORD 0h
.code
       main PROC
       mov eax, Oh
       mov ebx, 0h
       mov ecx, 8h
L1:
       mov eax, current
       mov ebx, previous
       mov previous, eax
       add eax, ebx
       mov current, eax
       loop L1
      invoke ExitProcess,0
main ENDP
end main
5. Copy a String in Reverse Order – With the use of a loop and indexed addressing, write an
assembly program that copies a string from source to target in a reverse order. Use the
following variables:
source BYTE "CS397 Assembly Language Programming", 0
target BYTE SIZEOF source DUP('?')
.data
  source BYTE "CS397 Assembly Language Programming",0
 target BYTE SIZEOF source DUP('?')
.code
main PROC
       mov esi,0
       mov edi, LENGTHOF source - 1
       mov ecx, SIZEOF source
```

```
mov eax, 0
mov al,source[esi]
mov target[edi],al
inc esi
dec edi
loop L1
```

INVOKE Exit Process, 0

Part 3 - Runtime Stack [7 points]

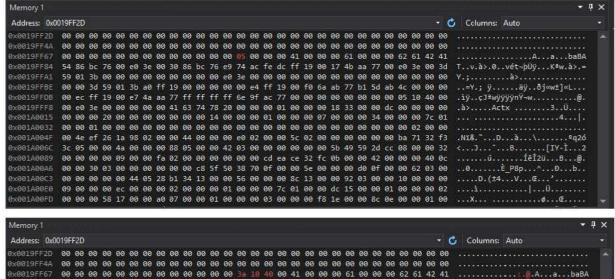
```
Using Microsoft Visual Studio, execute the codes below and answer questions 6-8.
.code
SumOfFirstNTerms PROC
; Sum of the first N terms of the arithmetic series,
; for example, if N = 5, then the result is 5+4+3+2+1=15
; Receives: EAX as N
; Returns: EAX = sum of the first N terms
; Requires: N needs to be a positive integer
       mov ecx, eax
       mov eax, 0
L1:
      add eax, ecx
      loop L1
       ret; line7
SumOfFirstNTerms ENDP
       main PROC
       mov eax, 'ABab'
       push eax; line1
       push 00000061h; line2
       push 00000041h; line3
       push 5; line4
       pop eax; line5
       call SumOfFirstNTerms; line6
       pop eax
main ENDP
end main
```

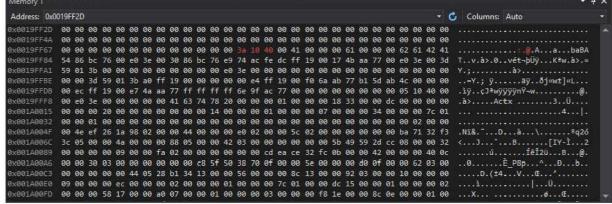
6. Show the screenshot of the stack in the memory (like the ones shown below) after the

codes are executed.









7. Explain how ESP and the stack are changed after each line of code (as specified by line1 – line7) is executed.

For example,

Line1: ESP is changed to 0x0019FF7C and XXXX is pushed onto the stack Line2: (so on)

Line 1: ESP is changed to 0x0019FF80 and ABab is pushed onto the stack.

Line 2: Esp is changed to 0x0019FF7C and a is pushed onto the stack.

Line 3: Esp is changed to 0x0019FF78 and A is pushed onto the stack.

Line 4: ESP is changed to 0x0019FF74 and 5 was pushed onto the stack

Line 5: ESP is changed to 0x0019FF78, the stack didn't change, and 5 was put into EAX.

Line 6: ESP didn't change and 40 10 3a was put onto the stack.

Line 7: ESP is changed to 0019FF7C and the stack did not change.

8. What is the final value of EAX once all the codes are executed? 00000041h