Luke Demi

1) Describe, in your own words, what the sections of code labeled fun1, fun2, fun3, and fun4 do. Be as specific as you can. For example, if one function implements a search algorithm, identify the searching algorithm used.

> fun1: Fun1 calculates the average of the elements in the given array myList. It iterates through each element of the array, sums them up, and then divides the sum by the number of elements to compute the average.
>
> fun2: Fun2 implements a bubble sort algorithm to sort the array myList in ascending order. It iterates through the array multiple times, comparing adjacent elements and swapping them if they are in the wrong order, until the array is sorted.
>
> fun3: Fun3 calculates the midpoint value of the array myList. It iterates through the array until it finds the value -1, then calculates the midpoint index of the array and retrieves the value at that index.
>
> fun4: Fun4 prints out every element of the array myList. It iterates through the array, printing each element until it encounters the value -1, at which point it stops.

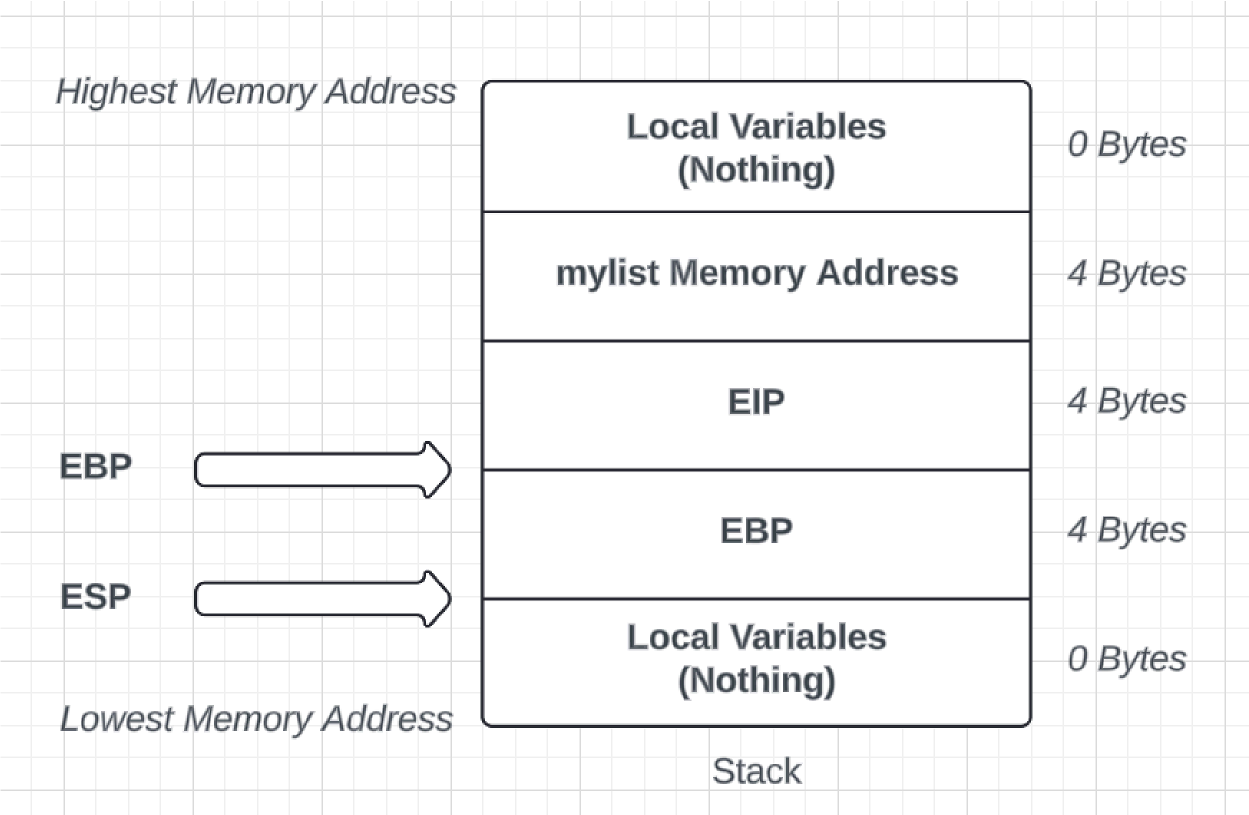
2) The first two lines of fun1, fun2, fun3, and fun4 are all identical. Why is this the case? What are these lines of code used for?

> These first two lines for all four of these functions are always push ebp and mov ebp, esp, which pushes the previous base pointer onto the stack and moves the current stack pointer into the base pointer. This is the basic preamble for establishing a stack frame. These lines of code are used for setting up a function's stack frame, providing a stable reference point and a new base pointer.

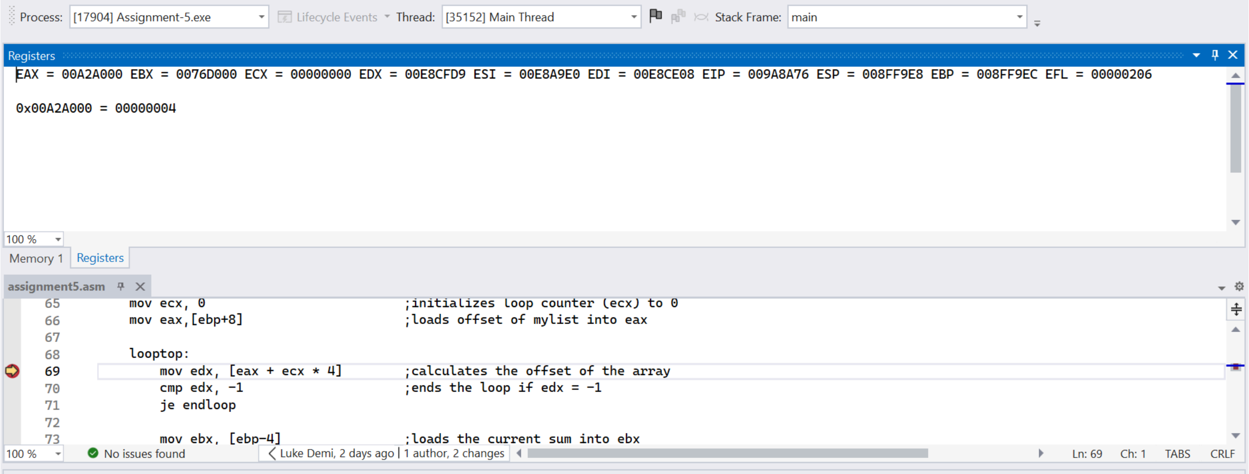
3) Demonstrate how a single line of C can map to multiple lines of assembly by writing a single line of C code that is equivalent to the collective behavior of lines 61, 62, and 63.

> int variable = 0;

4) Create a visual representation of the stack at line 68. This visual representation should describe what each one of those items is. This can be hand-drawn if you want. For reference, look at slides 11-19 of the Assembly Programming 3 slide deck.



5) Set a breakpoint at line 68 and execute the program. Take a screenshot that shows the register values at line #68.


Process: [17904] Assignment-5.exe Thread: [35152] Main Thread Stack Frame: main

```
EAX = 00A2A000 EBX = 0076D000 ECX = 00000000 EDX = 00E8CFD9 ESI = 00E8A9E0 EDI = 00E8CE08 EIP = 009A8A76 ESP = 008FF9E8 EBP = 008FF9EC EFL = 00000206

0x00A2A000 = 00000004
```

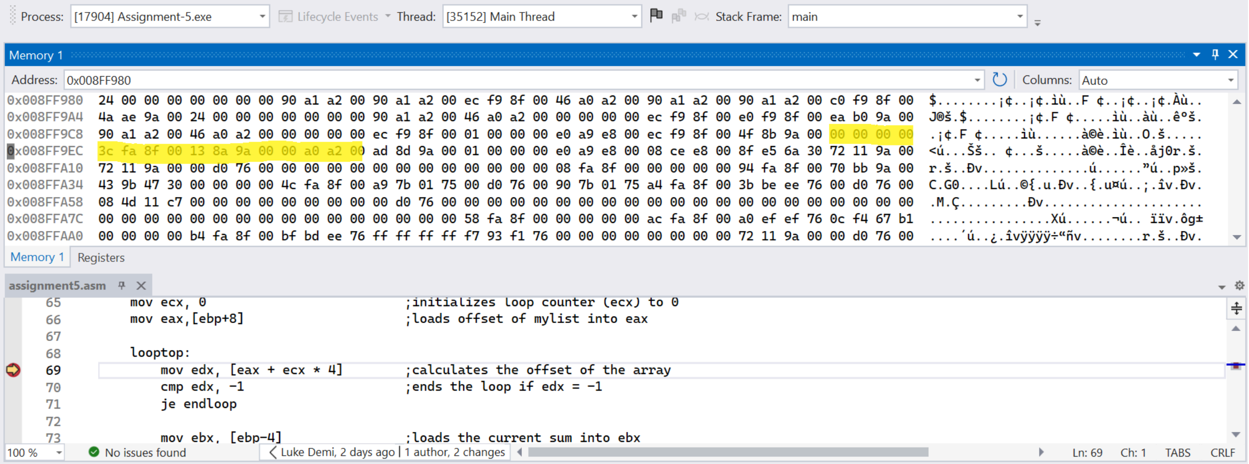
```
65       mov ecx, 0                          ;initializes loop counter (ecx) to 0
66       mov eax,[ebp+8]                     ;loads offset of mylist into eax
67
68       looptop:
69           mov edx, [eax + ecx * 4]        ;calculates the offset of the array
70           cmp edx, -1                     ;ends the loop if edx = -1
71           je endloop
72
73           mov ebx, [ebp-4]                ;loads the current sum into ebx
```

6) Analyze the register values. Use ESP and EBP to determine how much space is currently used by fun1's stack.

EBP = 008FF9EC; ESP = 008FF9E8

4 bytes is currently being used by fun1's stack because the difference between the base pointer and stack pointer is 4 bytes.

7) Use the value of the base pointer to locate the stack frame for fun1 in memory. Provide a screenshot that shows the stack frame of fun1 as well as any arguments that were passed into it.



Process: [17904] Assignment-5.exe Lifecycle Events Thread: [35152] Main Thread Stack Frame: main

Memory 1

Address: 0x008FF980 Columns: Auto

```
0x008FF980  24 00 00 00 00 00 00 00 90 a1 a2 00 90 a1 a2 00 ec f9 8f 00 46 a0 a2 00 90 a1 a2 00 90 a1 a2 00 c0 f9 8f 00  $........¡¢..¡¢.ìù..F ¢..¡¢..¡¢.Àù..
0x008FF9A4  4a ae 9a 00 24 00 00 00 00 00 00 00 90 a1 a2 00 46 a0 a2 00 00 00 00 00 ec f9 8f 00 e0 f9 8f 00 ea b0 9a 00  J®š.$........¡¢.F ¢.....ìù..àù..ê°š.
0x008FF9C8  90 a1 a2 00 46 a0 a2 00 00 00 00 00 ec f9 8f 00 01 00 00 00 e0 a9 e8 00 ec f9 8f 00 4f 8b 9a 00 00 00 00 00  .¡¢.F ¢.....ìù......à©è.ìù..O.š.....
0x008FF9EC  3c fa 8f 00 13 8a 9a 00 00 a0 a2 00 ad 8d 9a 00 01 00 00 00 e0 a9 e8 00 08 ce e8 00 8f e5 6a 30 72 11 9a 00  <ú...Šš.. ¢...š.....à©è..Îè..åj0r.š.
0x008FFA10  72 11 9a 00 00 d0 76 00 00 00 00 00 00 00 00 00 00 00 00 00 08 fa 8f 00 00 00 00 00 94 fa 8f 00 70 bb 9a 00  r.š..Ðv.............ú......”ú..p»š.
0x008FFA34  43 9b 47 30 00 00 00 00 4c fa 8f 00 a9 7b 01 75 00 d0 76 00 90 7b 01 75 a4 fa 8f 00 3b be ee 76 00 d0 76 00  C.G0....Lú..©{.u.Ðv..{.u¤ú..;.îv.Ðv.
0x008FFA58  08 4d 11 c7 00 00 00 00 00 00 00 00 00 d0 76 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .M.Ç.........Ðv.....................
0x008FFA7C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 58 fa 8f 00 00 00 00 00 ac fa 8f 00 a0 ef ef 76 0c f4 67 b1  ................Xú......¬ú.. ïïv.ôg±
0x008FFAA0  00 00 00 00 b4 fa 8f 00 bf bd ee 76 ff ff ff ff f7 93 f1 76 00 00 00 00 00 00 00 00 72 11 9a 00 00 d0 76 00  ....´ú..¿.îvÿÿÿÿ÷“ñv........r.š..Ðv.
```

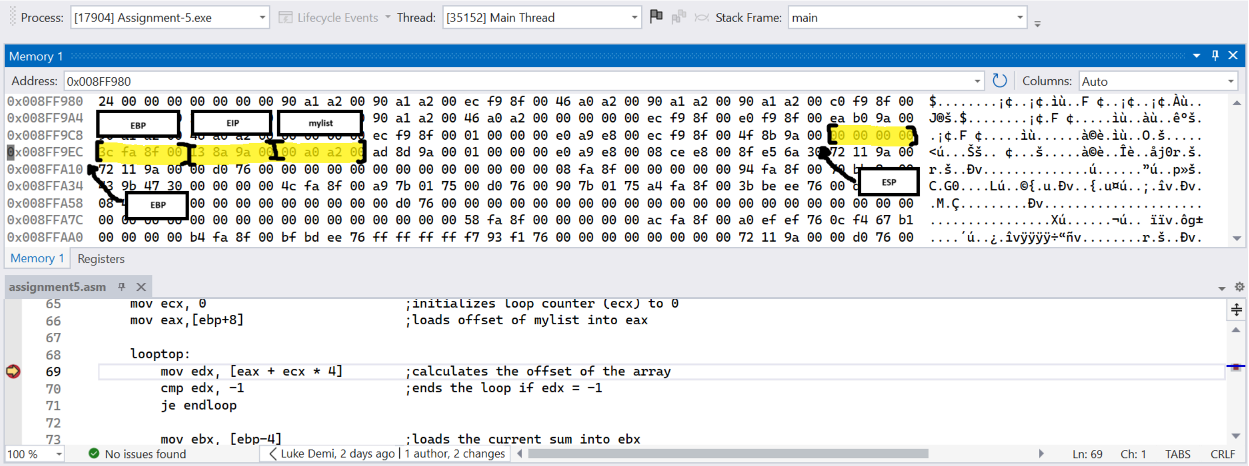
Memory 1 Registers

assignment5.asm

```
        mov ecx, 0                              ;initializes loop counter (ecx) to 0
        mov eax,[ebp+8]                         ;loads offset of mylist into eax

        looptop:
            mov edx, [eax + ecx * 4]            ;calculates the offset of the array
            cmp edx, -1                         ;ends the loop if edx = -1
            je endloop

            mov ebx, [ebp-4]                    ;loads the current sum into ebx
```

100 % No issues found Luke Demi, 2 days ago | 1 author, 2 changes Ln: 69 Ch: 1 TABS CRLF

8) Edit the screenshot taken in step #7 to show where on the stack each item you documented in step #4 is.



Process: [17904] Assignment-5.exe Lifecycle Events Thread: [35152] Main Thread Stack Frame: main

Memory 1

Address: 0x008FF980 Columns: Auto

```
0x008FF980  24 00 00 00 00 00 00 00 90 a1 a2 00 90 a1 a2 00 ec f9 8f 00 46 a0 a2 00 90 a1 a2 00 90 a1 a2 00 c0 f9 8f 00  $........¡¢..¡¢.ìù..F ¢..¡¢..¡¢.Àù..
0x008FF9A4  4a ae 9a 00 24 00 00 00 00 00 00 00 90 a1 a2 00 46 a0 a2 00 00 00 00 00 ec f9 8f 00 e0 f9 8f 00 ea b0 9a 00  J®š.$........¡¢.F ¢.....ìù..àù..ê°š.
0x008FF9C8  90 a1 a2 00 46 a0 a2 00 00 00 00 00 ec f9 8f 00 01 00 00 00 e0 a9 e8 00 ec f9 8f 00 4f 8b 9a 00 00 00 00 00  .¡¢.F ¢.....ìù......à©è.ìù..O.š.....
0x008FF9EC  3c fa 8f 00 13 8a 9a 00 00 a0 a2 00 ad 8d 9a 00 01 00 00 00 e0 a9 e8 00 08 ce e8 00 8f e5 6a 30 72 11 9a 00  <ú...Šš.. ¢...š.....à©è..Îè..åj0r.š.
0x008FFA10  72 11 9a 00 00 d0 76 00 00 00 00 00 00 00 00 00 00 00 00 00 08 fa 8f 00 00 00 00 00 94 fa 8f 00 70 bb 9a 00  r.š..Ðv.............ú......”ú..p»š.
0x008FFA34  43 9b 47 30 00 00 00 00 4c fa 8f 00 a9 7b 01 75 00 d0 76 00 90 7b 01 75 a4 fa 8f 00 3b be ee 76 00 d0 76 00  C.G0....Lú..©{.u.Ðv..{.u¤ú..;.îv.Ðv.
0x008FFA58  08 4d 11 c7 00 00 00 00 00 00 00 00 00 d0 76 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .M.Ç.........Ðv.....................
0x008FFA7C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 58 fa 8f 00 00 00 00 00 ac fa 8f 00 a0 ef ef 76 0c f4 67 b1  ................Xú......¬ú.. ïïv.ôg±
0x008FFAA0  00 00 00 00 b4 fa 8f 00 bf bd ee 76 ff ff ff ff f7 93 f1 76 00 00 00 00 00 00 00 00 72 11 9a 00 00 d0 76 00  ....´ú..¿.îvÿÿÿÿ÷“ñv........r.š..Ðv.
```

Memory 1 Registers

assignment5.asm

```
        mov ecx, 0                              ;initializes loop counter (ecx) to 0
        mov eax,[ebp+8]                         ;loads offset of mylist into eax

        looptop:
            mov edx, [eax + ecx * 4]            ;calculates the offset of the array
            cmp edx, -1                         ;ends the loop if edx = -1
            je endloop

            mov ebx, [ebp-4]                    ;loads the current sum into ebx
```

100 % No issues found Luke Demi, 2 days ago | 1 author, 2 changes Ln: 69 Ch: 1 TABS CRLF

9) What would be the impact if a programmer left off the second-to-last line of code (ie: the line of code before 'ret') in each of these sections of code?

The second-to-last line of code in each of these sections of code is pop ebp, which restores the base pointer before returning from the function. If this portion of code was left off, the base pointer would not be reset to where it was supposed to be outside of the function, which would lead to stack corruption issues and memory leaks. The code would end up executing differently than original intended.

10) Briefly compare and contrast how fun1 and fun2 declare variables.

Fun1 declared a variable by subtracting 4 from esp and assigned these open bytes to 0. Fun2 on the other hand declared the variable by pushing 0 directly onto the stack. Both fun1 and fun2 accomplish the same thing, but have different methods, and both are used for declaring variables.

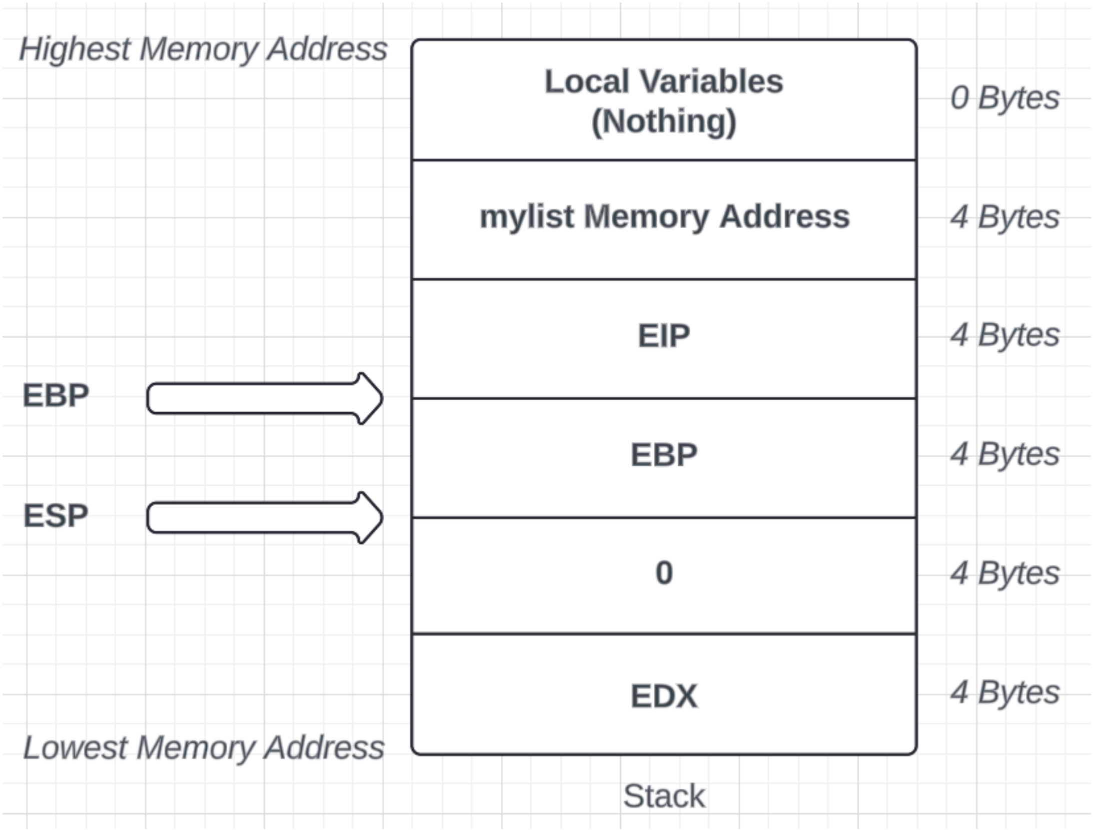
11) Line #33 is ‘push eax’. Which line of code sets the value of eax prior to line #33 being executed? What C keyword is being implemented with these lines of code?

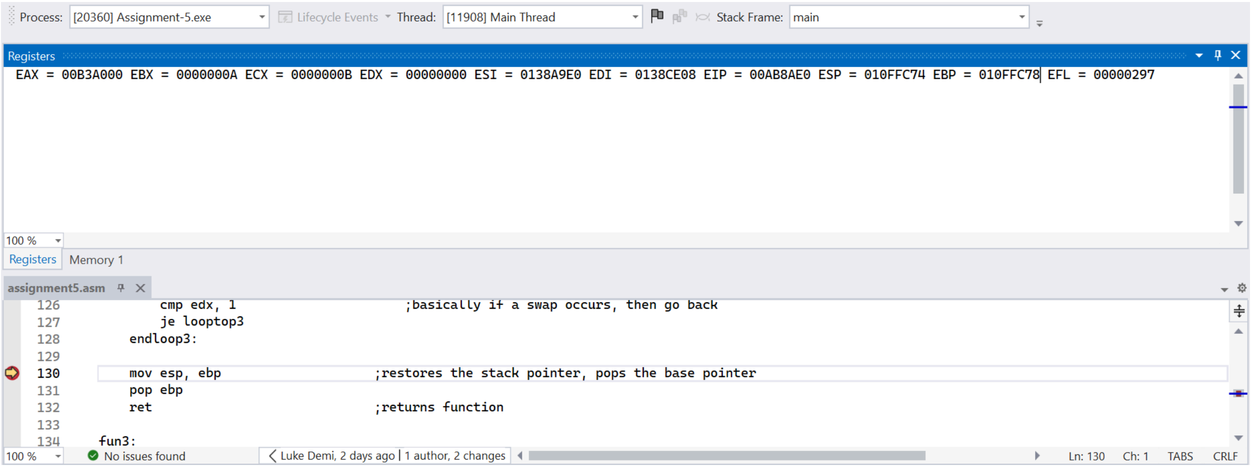
Line #30 sets the value of eax prior to line #33 being executed, because line #30 calls the fun1 function which then stores the accumulated sum of the elements of mylist into the eax register, setting the value of eax (line #81). The C keyword being implemented here is return, since the combination of these lines of code serves the purpose of returning a value from the function.

12) Does fun2 have a return value? If so, what line of code sets the return value for fun2? Provide evidence of your answer.

Fun2 does not have an explicit return value, it simply alters the contents of the mylist array through a bubble sort algorithm.

13) [5pts] Repeat steps #4, 5, 6, 7, and 8 for line #130




Process: [20360] Assignment-5.exe Thread: [11908] Main Thread Stack Frame: main

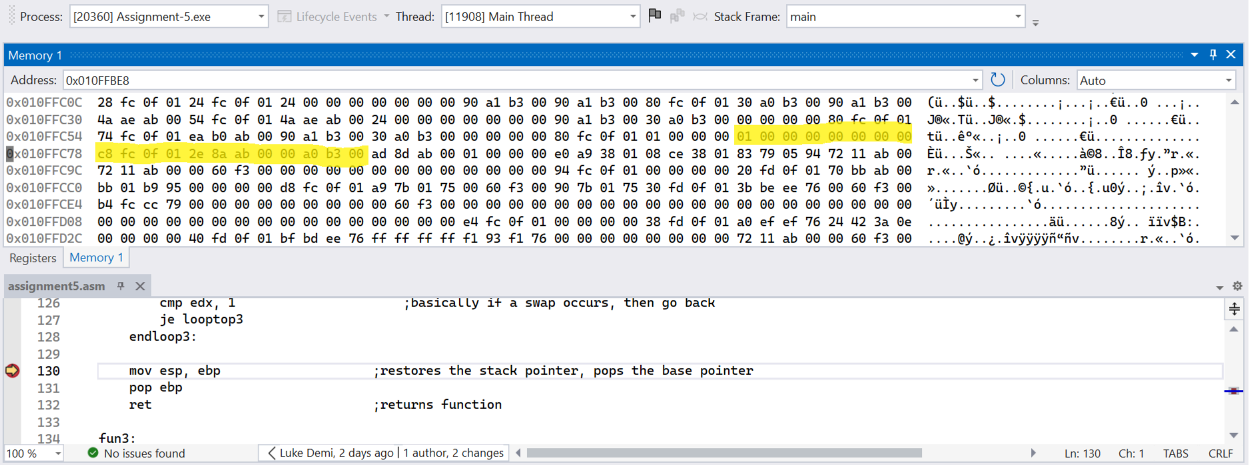
```
EAX = 00B3A000 EBX = 0000000A ECX = 0000000B EDX = 00000000 ESI = 0138A9E0 EDI = 0138CE08 EIP = 00AB8AE0 ESP = 010FFC74 EBP = 010FFC78 EFL = 00000297
```

```
            cmp edx, 1                          ;basically if a swap occurs, then go back
            je looptop3
        endloop3:

        mov esp, ebp                        ;restores the stack pointer, pops the base pointer
        pop ebp
        ret                                 ;returns function

    fun3:
```

No issues found Luke Demi, 2 days ago | 1 author, 2 changes Ln: 130 Ch: 1 TABS CRLF

EBP = 010FFC78; ESP = 010FFC74

4 bytes is currently being used by fun2’s stack because the difference between the base pointer and stack pointer is 4 bytes.



Process: [20360] Assignment-5.exe Thread: [11908] Main Thread Stack Frame: main

Memory 1

Address: 0x010FFBE8 Columns: Auto

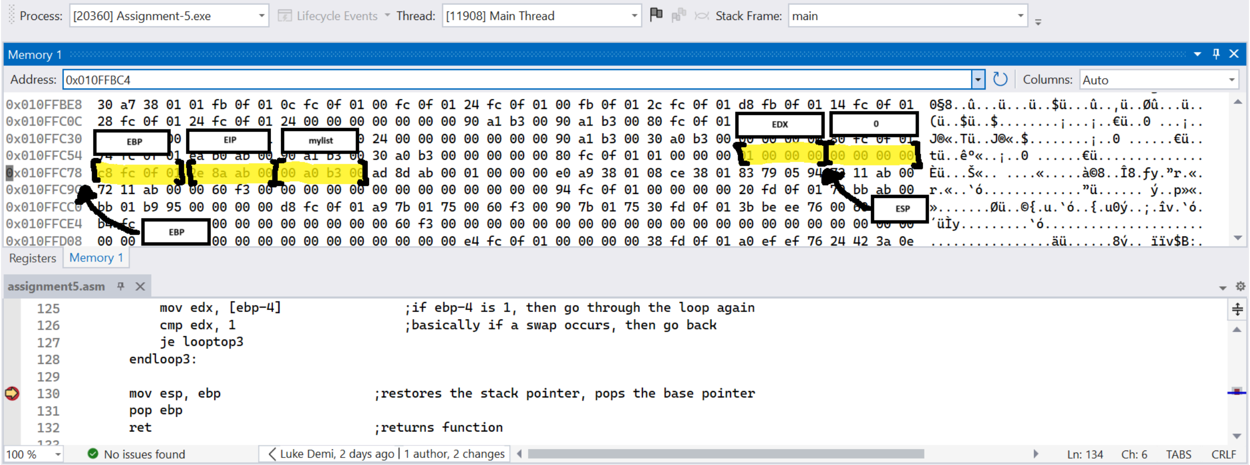
```
0x010FFC0C  28 fc 0f 01 24 fc 0f 01 24 00 00 00 00 00 00 00 90 a1 b3 00 90 a1 b3 00 80 fc 0f 01 30 a0 b3 00 90 a1 b3 00  (ü..$ü..$.......¡...¡..€ü..0 ...¡..
0x010FFC30  4a ae ab 00 54 fc 0f 01 4a ae ab 00 24 00 00 00 00 00 00 00 90 a1 b3 00 30 a0 b3 00 00 00 00 00 80 fc 0f 01  J®«.Tü..J®«.$.......¡..0 ......€ü..
0x010FFC54  74 fc 0f 01 ea b0 ab 00 90 a1 b3 00 30 a0 b3 00 00 00 00 00 80 fc 0f 01 01 00 00 00 01 00 00 00 00 00 00 00  tü..ê°«..¡..0 ......€ü............
0x010FFC78  c8 fc 0f 01 2e 8a ab 00 00 a0 b3 00 ad 8d ab 00 01 00 00 00 e0 a9 38 01 08 ce 38 01 83 79 05 94 72 11 ab 00  Èü...Š«.. ....«.....à©8..Î8.ƒy.”r.«.
0x010FFC9C  72 11 ab 00 00 60 f3 00 00 00 00 00 00 00 00 00 00 00 00 00 94 fc 0f 01 00 00 00 00 20 fd 0f 01 70 bb ab 00  r.«..`ó............”ü...... ý..p»«.
0x010FFCC0  bb 01 b9 95 00 00 00 00 d8 fc 0f 01 a9 7b 01 75 00 60 f3 00 90 7b 01 75 30 fd 0f 01 3b be ee 76 00 60 f3 00  ».......Øü..©{.u.`ó..{.u0ý..;.îv.`ó.
0x010FFCE4  b4 fc cc 79 00 00 00 00 00 00 00 00 00 60 f3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ´üÌy.........`ó.....................
0x010FFD08  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 e4 fc 0f 01 00 00 00 00 38 fd 0f 01 a0 ef ef 76 24 42 3a 0e  ................äü......8ý.. ïïv$B:.
0x010FFD2C  00 00 00 00 40 fd 0f 01 bf bd ee 76 ff ff ff ff f1 93 f1 76 00 00 00 00 00 00 00 00 72 11 ab 00 00 60 f3 00  ....@ý..¿.îvÿÿÿÿñ“ñv........r.«..`ó.
```

Registers Memory 1

assignment5.asm

```
            cmp edx, 1                      ;basically if a swap occurs, then go back
            je looptop3
        endloop3:

        mov esp, ebp                    ;restores the stack pointer, pops the base pointer
        pop ebp
        ret                             ;returns function

    fun3:
```

100 % No issues found Luke Demi, 2 days ago | 1 author, 2 changes Ln: 130 Ch: 1 TABS CRLF



Process: [20360] Assignment-5.exe Thread: [11908] Main Thread Stack Frame: main

Memory 1

Address: 0x010FFBC4 Columns: Auto

```
0x010FFBE8  30 a7 38 01 01 fb 0f 01 0c fc 0f 01 00 fc 0f 01 24 fc 0f 01 00 fb 0f 01 2c fc 0f 01 d8 fb 0f 01 14 fc 0f 01  0§8..û...ü...ü..$ü...û..,ü..Øû...ü..
0x010FFC0C  28 fc 0f 01 24 fc 0f 01 24 00 00 00 00 00 00 00 90 a1 b3 00 90 a1 b3 00 80 fc 0f 01                          (ü..$ü..$.......¡...¡..€ü..0 ...¡..
0x010FFC30              0                                 0 24 00 00 00 00 00 00 00 90 a1 b3 00 30 a0 b3 00             J®«.Tü..J®«.$.......¡..0 ......€ü..
0x010FFC54  74 fc 0f 01 ea b0 ab 00 90 a1 b3 00 30 a0 b3 00 00 00 00 00 80 fc 0f 01 01 00 00 00 01 00 00 00 00 00 00 00  tü..ê°«..¡..0 ......€ü............
0x010FFC78  c8 fc 0f 01 2e 8a ab 00 00 a0 b3 00 ad 8d ab 00 01 00 00 00 e0 a9 38 01 08 ce 38 01 83 79 05 94    11 ab 00  Èü...Š«.. ....«.....à©8..Î8.ƒy.”r.«.
0x010FFC9C  72 11 ab 00 00 60 f3 00 00 00 00 00 00 00 00 00 00 00 00 00 94 fc 0f 01 00 00 00 00 20 fd 0f 01 7  bb ab 00  r.«..`ó............”ü...... ý..p»«.
0x010FFCC0  bb 01 b9 95 00 00 00 00 d8 fc 0f 01 a9 7b 01 75 00 60 f3 00 90 7b 01 75 30 fd 0f 01 3b be ee 76 00 0        ».......Øü..©{.u.`ó..{.u0ý..;.îv.`ó.
0x010FFCE4  b4 fc             00 00 00 00 00 00 00 60 f3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ´üÌy.........`ó.....................
0x010FFD08  00 00             00 00 00 00 00 00 00 00 00 00 e4 fc 0f 01 00 00 00 00 38 fd 0f 01 a0 ef ef 76 24 42 3a 0e  ................äü......8ý.. ïïv$B:.
```

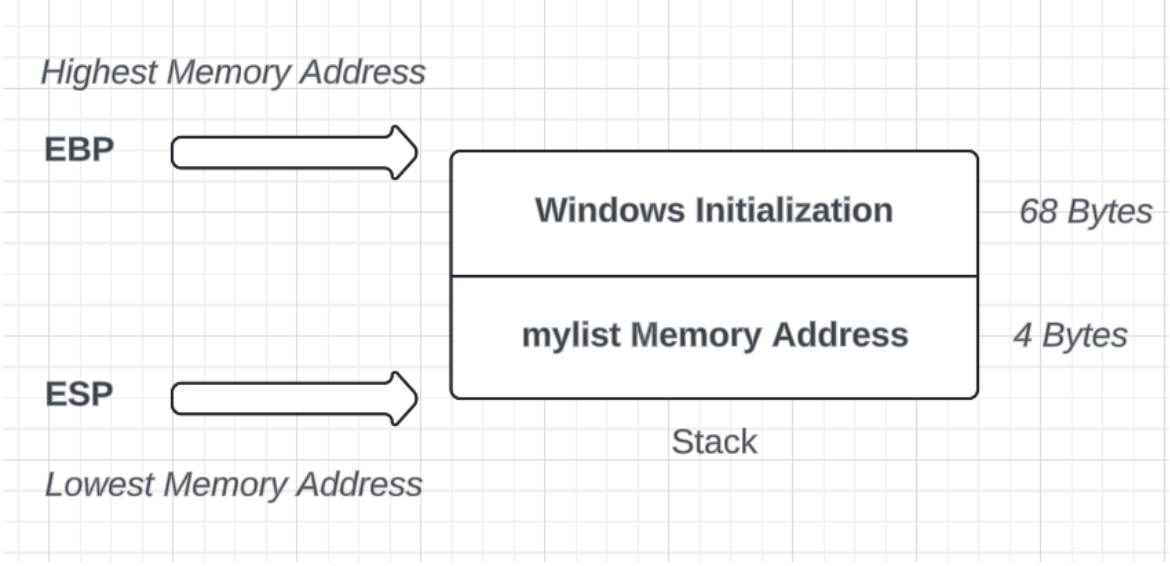
Registers Memory 1

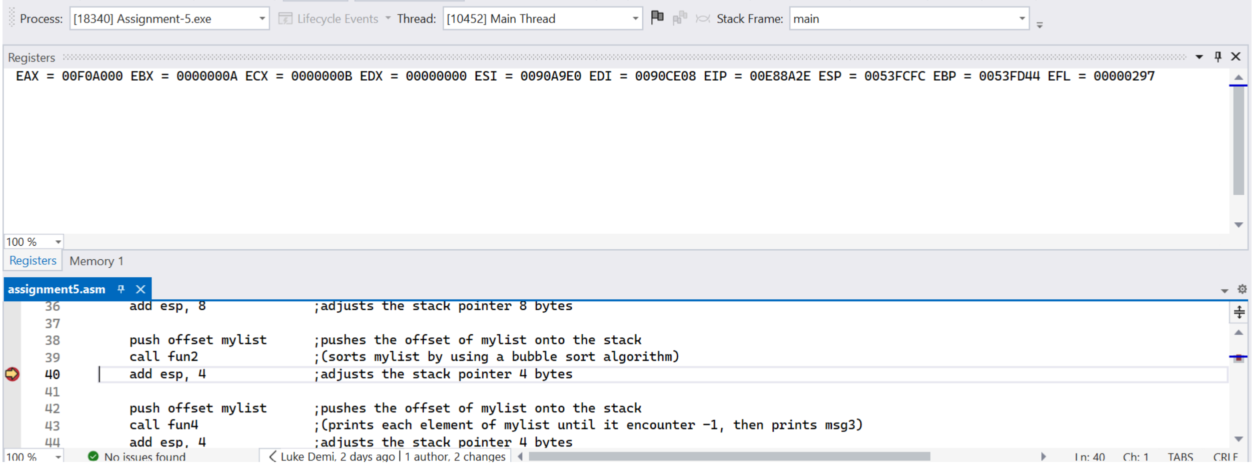
assignment5.asm

```
            mov edx, [ebp-4]                ;if ebp-4 is 1, then go through the loop again
            cmp edx, 1                      ;basically if a swap occurs, then go back
            je looptop3
        endloop3:

        mov esp, ebp                    ;restores the stack pointer, pops the base pointer
        pop ebp
        ret                             ;returns function
```

100 % No issues found Luke Demi, 2 days ago | 1 author, 2 changes Ln: 134 Ch: 6 TABS CRLF

14) [5pts] Repeat steps #4, 5, 6, 7, and 8 for line #40



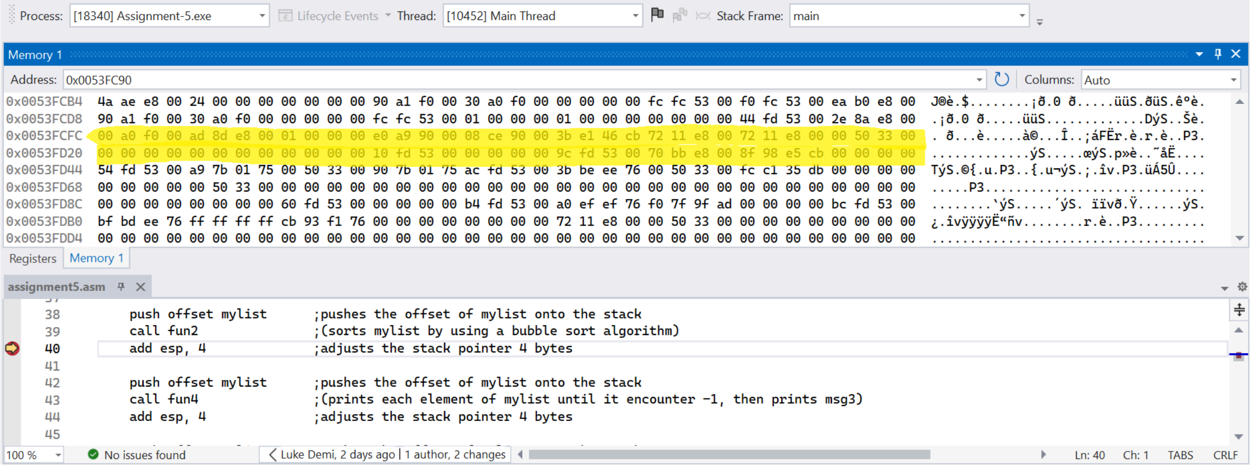


```
EAX = 00F0A000 EBX = 0000000A ECX = 0000000B EDX = 00000000 ESI = 0090A9E0 EDI = 0090CE08 EIP = 00E88A2E ESP = 0053FCFC EBP = 0053FD44 EFL = 00000297
```

```
        add esp, 8              ;adjusts the stack pointer 8 bytes

        push offset mylist      ;pushes the offset of mylist onto the stack
        call fun2               ;(sorts mylist by using a bubble sort algorithm)
        add esp, 4              ;adjusts the stack pointer 4 bytes

        push offset mylist      ;pushes the offset of mylist onto the stack
        call fun4               ;(prints each element of mylist until it encounter -1, then prints msg3)
        add esp, 4              ;adjusts the stack pointer 4 bytes
```

EBP = 0053FD44; ESP = 0053FCFC

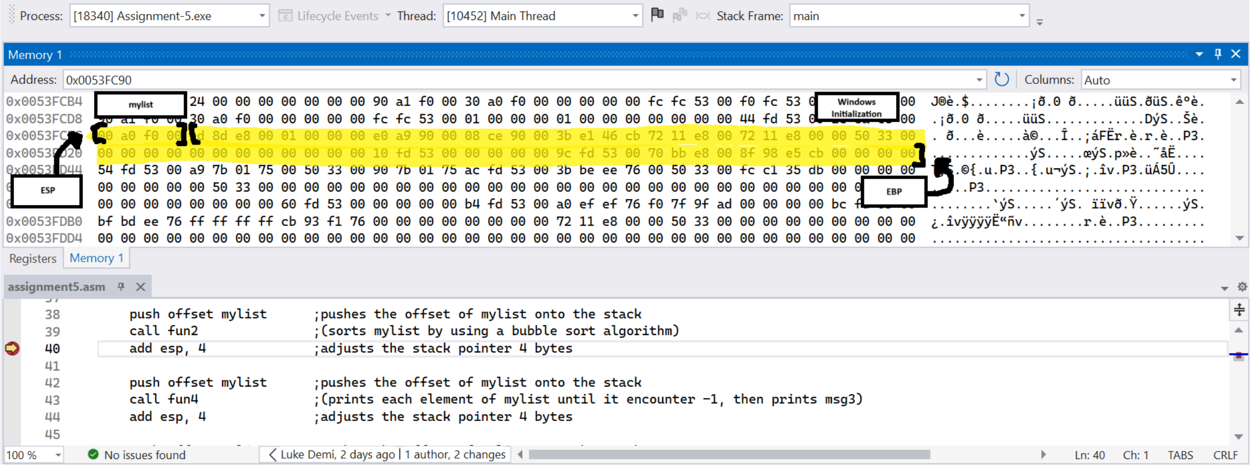
72 bytes is currently being used by fun2's stack because the difference between the base pointer and stack pointer is 72 bytes.



Address: 0x0053FC90

```
0x0053FCB4  4a ae e8 00 24 00 00 00 00 00 00 00 90 a1 f0 00 30 a0 f0 00 00 00 00 00 fc fc 53 00 f0 fc 53 00 ea b0 e8 00  J®è.$........¡ð.0 ð.....üüS.ðüS.êºè.
0x0053FCD8  90 a1 f0 00 30 a0 f0 00 00 00 00 00 fc fc 53 00 01 00 00 00 01 00 00 00 00 00 00 00 44 fd 53 00 2e 8a e8 00  .¡ð.0 ð.....üüS.............DýS..Šè.
0x0053FCFC  00 a0 f0 00 ad 8d e8 00 01 00 00 00 e0 a9 90 00 08 ce 90 00 3b e1 46 cb 72 11 e8 00 72 11 e8 00 00 50 33 00  . ð...è.....à©...Î..;áFËr.è.r.è..P3.
0x0053FD20  00 00 00 00 00 00 00 00 00 00 00 00 10 fd 53 00 00 00 00 00 9c fd 53 00 70 bb e8 00 8f 98 e5 cb 00 00 00 00  .............ýS.....œýS.p»è..˜åË....
0x0053FD44  54 fd 53 00 a9 7b 01 75 00 50 33 00 90 7b 01 75 ac fd 53 00 3b be ee 76 00 50 33 00 fc c1 35 db 00 00 00 00  TýS.©{.u.P3..{.u¬ýS.;.îv.P3.üÁ5Û....
0x0053FD68  00 00 00 00 00 50 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....P3.............................
0x0053FD8C  00 00 00 00 00 00 00 00 60 fd 53 00 00 00 00 00 b4 fd 53 00 a0 ef ef 76 f0 7f 9f ad 00 00 00 00 bc fd 53 00  ........`ýS.....´ýS. ïïvð.Ÿ.....ýS.
0x0053FDB0  bf bd ee 76 ff ff ff ff cb 93 f1 76 00 00 00 00 00 00 00 00 72 11 e8 00 00 50 33 00 00 00 00 00 00 00 00 00  ¿.îvÿÿÿÿË“ñv........r.è..P3.........
0x0053FDD4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....................................
```

```
        push offset mylist      ;pushes the offset of mylist onto the stack
        call fun2               ;(sorts mylist by using a bubble sort algorithm)
        add esp, 4              ;adjusts the stack pointer 4 bytes

        push offset mylist      ;pushes the offset of mylist onto the stack
        call fun4               ;(prints each element of mylist until it encounter -1, then prints msg3)
        add esp, 4              ;adjusts the stack pointer 4 bytes

```



Address: 0x0053FC90

```
        push offset mylist      ;pushes the offset of mylist onto the stack
        call fun2               ;(sorts mylist by using a bubble sort algorithm)
        add esp, 4              ;adjusts the stack pointer 4 bytes

        push offset mylist      ;pushes the offset of mylist onto the stack
        call fun4               ;(prints each element of mylist until it encounter -1, then prints msg3)
        add esp, 4              ;adjusts the stack pointer 4 bytes

```

15) What is the primary difference between your answers to #13 and #14?

For #14, line #40 is a part of the main function which means the items pushed onto the stack in the fun2 function are no longer present, while for #13, line #130 is a part of the fun2 function which means all the items pushed onto the stack in the fun2 function are still present and on the stack. Also, #14 includes 68 bytes of windows initialization data on the stack because it is in the main function, and this initialization data is not present in the fun2 stack.