Computer Science
C.Sc. 342

Quiz No.2 To be performed

5:00-6:15 PM on March 23, 2022

Submit by 6:15 PM 03/23/2022on Slack to Instructor **Please write your Last Name on every page:**

NO CORRECTIONS ARE ALLOWED IN ANSWER CELLS!!!!!

You may use the back page for computations.
Please answer all questions. **Not all questions are of equal difficulty.**

Please review the entire quiz first and then budget your time carefully.

Please hand write and sign statements affirming that you will not cheat:

"I will neither give nor receive unauthorized assistance on this exam. I will use only one computing device to perform this test"

Please hand write and sign here:

This quiz has 6 pages.

Question	Your Grade	Max Grade
1.1		5
1.2		10
1.3		10
1.4		10
2.1.1		15
2.1.2		15
2.1.3		15
2.2.1		5
2.2.2		5
2.2.3		5
2.3		5

Total: 100

Question 1.

A student, while debugging his program, unintentionally displayed partially corrupted DISSASSEMBLY windows in MS Visual Studio Debug environment.

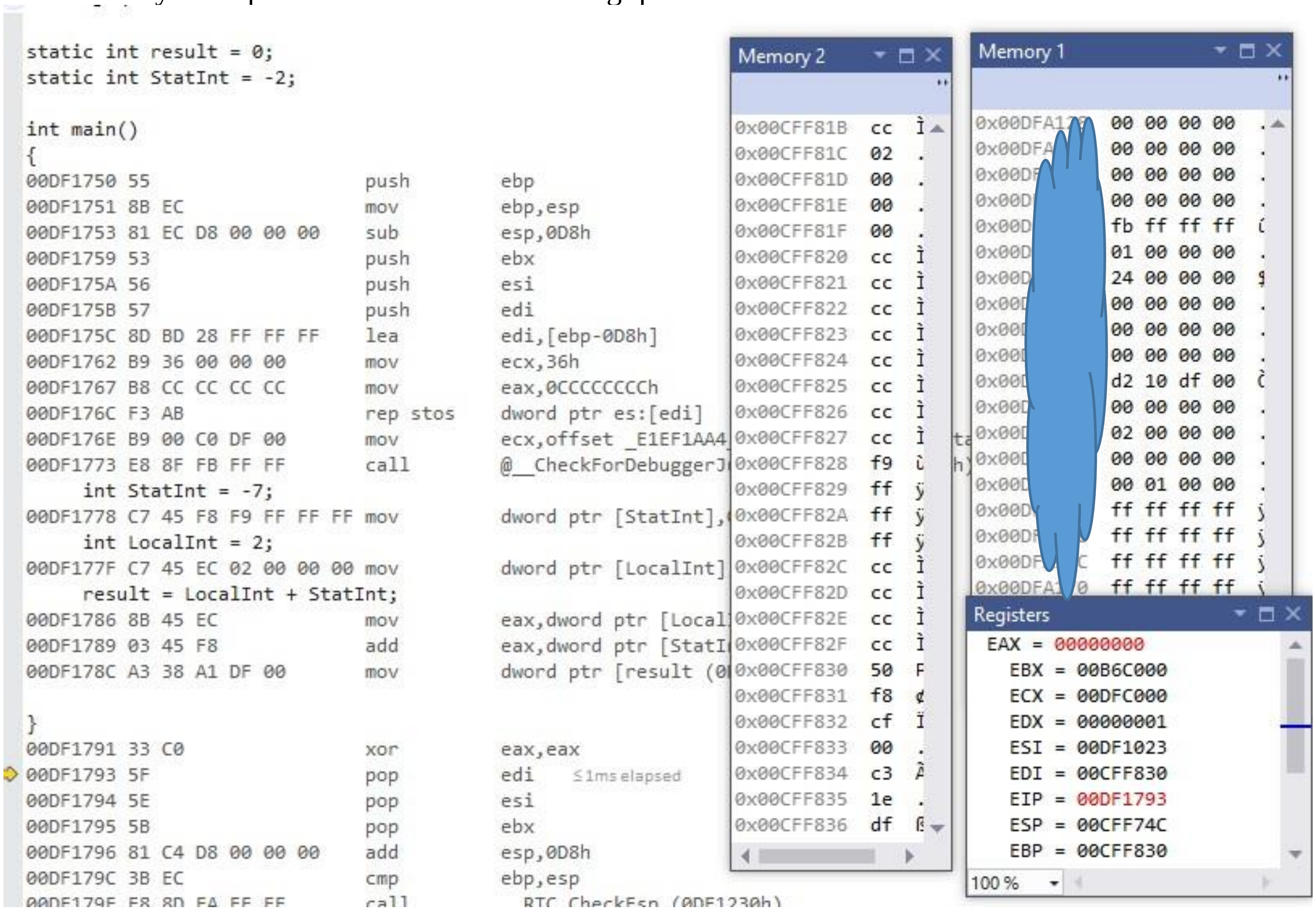
He was able to display correctly Register window, and two Memory windows.

His task was to determine addresses of variables in the expression

result = LocalInt + StatInt in Memory at the instance of the snapshot.

He is not allowed to restart the debug session.

Can you help him to answer the following questions:



```
static int result = 0;
static int StatInt = -2;

int main()
{
00DF1750 55                   push       ebp
00DF1751 8B EC                mov        ebp,esp
00DF1753 81 EC D8 00 00 00    sub        esp,0D8h
00DF1759 53                   push       ebx
00DF175A 56                   push       esi
00DF175B 57                   push       edi
00DF175C 8D BD 28 FF FF FF    lea        edi,[ebp-0D8h]
00DF1762 B9 36 00 00 00       mov        ecx,36h
00DF1767 B8 CC CC CC CC       mov        eax,0CCCCCCCCh
00DF176C F3 AB                rep stos   dword ptr es:[edi]
00DF176E B9 00 C0 DF 00       mov        ecx,offset _E1EF1AA4_
00DF1773 E8 8F FB FF FF       call       @__CheckForDebuggerJ
    int StatInt = -7;
00DF1778 C7 45 F8 F9 FF FF FF mov        dword ptr [StatInt],
    int LocalInt = 2;
00DF177F C7 45 EC 02 00 00 00 mov        dword ptr [LocalInt]
    result = LocalInt + StatInt;
00DF1786 8B 45 EC             mov        eax,dword ptr [Local
00DF1789 03 45 F8             add        eax,dword ptr [StatI
00DF178C A3 38 A1 DF 00       mov        dword ptr [result (0

}
00DF1791 33 C0                xor        eax,eax
00DF1793 5F                   pop        edi    ≤1ms elapsed
00DF1794 5E                   pop        esi
00DF1795 5B                   pop        ebx
00DF1796 81 C4 D8 00 00 00    add        esp,0D8h
00DF179C 3B EC                cmp        ebp,esp
00DF179E E8 8D FA FF FF       call         RTC_CheckEsp (0DF1230h)
```

Memory 2

```
0x00CFF81B  cc  Ì
0x00CFF81C  02  .
0x00CFF81D  00  .
0x00CFF81E  00  .
0x00CFF81F  00  .
0x00CFF820  cc  Ì
0x00CFF821  cc  Ì
0x00CFF822  cc  Ì
0x00CFF823  cc  Ì
0x00CFF824  cc  Ì
0x00CFF825  cc  Ì
0x00CFF826  cc  Ì
0x00CFF827  cc  Ì
0x00CFF828  f9  ù
0x00CFF829  ff  ÿ
0x00CFF82A  ff  ÿ
0x00CFF82B  ff  ÿ
0x00CFF82C  cc  Ì
0x00CFF82D  cc  Ì
0x00CFF82E  cc  Ì
0x00CFF82F  cc  Ì
0x00CFF830  50  P
0x00CFF831  f8  ¢
0x00CFF832  cf  Ï
0x00CFF833  00  .
0x00CFF834  c3  Ã
0x00CFF835  1e  .
0x00CFF836  df  ß
```

Memory 1

```
0x00DFA1     00 00 00 00  .
0x00DFA      00 00 00 00  .
0x00DF       00 00 00 00  .
0x00D        00 00 00 00  .
0x00D        fb ff ff ff  û
0x00D        01 00 00 00  .
0x00D        24 00 00 00  $
0x00D        00 00 00 00  .
0x00D        00 00 00 00  .
0x00D        00 00 00 00  .
0x00D        d2 10 df 00  Ò
0x00D        00 00 00 00  .
0x00D        02 00 00 00  .
0x00D        00 00 00 00  .
0x00D        00 01 00 00  .
0x00D        ff ff ff ff  ÿ
0x00DF       ff ff ff ff  ÿ
0x00DF   C   ff ff ff ff  ÿ
0x00DFA1 0   ff ff ff ff  ÿ
```

Registers

```
EAX = 00000000
  EBX = 00B6C000
  ECX = 00DFC000
  EDX = 00000001
  ESI = 00DF1023
  EDI = 00CFF830
  EIP = 00DF1793
  ESP = 00CFF74C
  EBP = 00CFF830
```

1.1 [5 points] What is the address of the instruction that will be executed next instance?

ANSWER: This is shown on the EIP register. If we look closely at EIP register, it shown that the address is 0x00DF1793.

1.2 [10 points] Can you determine the address of variable **StatInt** in the expression? [✓ **YES**] **or NO.**
*Please circle around your answer***. IF** *No is your answer, then go to the next question*
ELSE *Please compute the address of variable* ***StatInt*** *in memory , and determine*
the value of variable StatInt you can read from memory:
Address of ***StatInt*** *is ……. Value*
of ***StatInt*** *in memory is Please*
justify your answers.

ANSWER:
The address of the variable StatInt is 0x00CFF828.
Reason: The value of -7 in hexadecimal is FFF9, therefore, we can see in the memory windows 2 that the value of F9 FF FF is at address 0x00CFF828.

1.3 [10points] Can you determine the address of variable **LocalInt** in the expression? [✓ **YES**] **or NO.**
*Please circle around your answer***. IF** *No is your answer, then go to the next question*
ELSE *Please compute the address of variable* ***LocalInt*** *in memory , and determine*
the value of variable ***LocalInt*** *you can read from memory:*
Address of ***LocalInt*** *is …….*
Value of ***LocalInt*** *in memory is….*
Please justify your answers.

ANSWER:
The memory address is 0x00CFF81C
Reason: If we look at the memory window 2, it holds the value of 2 in hexadecimal.

1.4 [10 points] Can you determine the address of variable **result** in the expression? [✓ **YES**] **or NO.**
*Please circle around your answer***. IF** *No is your answer, then go to the next question*
ELSE *Please compute the address of variable* **result** *in memory , and determine the*
value of variable **result** *you can read from memory:*
Address of **result** *is ……. Value*
of **result** *in memory is Please*
justify your answers.

ANSWER:
The memory address is 0x00DFA138.
Reason: If we look at the memory window 1, it holds value of -5 in hexadecimal.

Question 2.

A student wrote MIPS assembly program and executed it in MARS simulator.

```
    .data
array1:  .word                    -1,0x7fffffff,0x10000080,0x80000010
.text
```

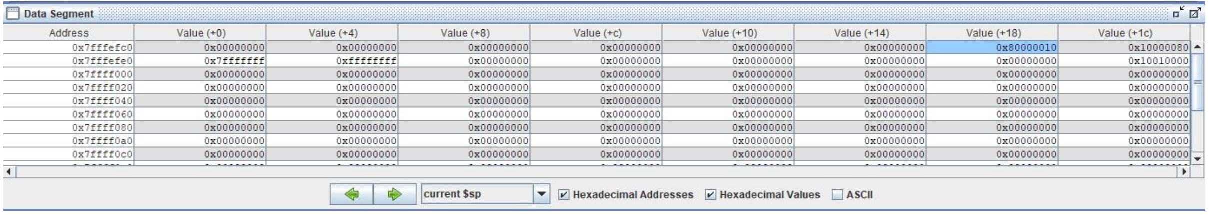
```
            main:
                                    la $t1,array1
# create Frame pointer

                                        add $fp,$zero,$sp
   #Store the address of the first element on stack  using frame pointer
                      sw $t1,0($fp) #allocate memory on Stack for 6
   integers
    addi $sp,$sp,-24
   #load       FIRST element from array1[0] to register $s0
                lw  $s0,0($t1)
   # push $s0 (NO PUSH!)i.e. store  register $s0 on #top of the stack
                 sw  $s0,0($sp)
   #load       SECOND element from array1[1] to register $s0
     lw  $s0,4($t1)  #create new top of the stack                    addi $sp,$sp,-4
               sw  $s0,0($sp)
                       #
     #load third element from array1[2] to register  $s0              lw
    $s0,8($t1)  #create new top of the stack

               addi      $sp,$sp,-4
    sw  $s0,0(sp)

     #load forth element from array1[3] to register  $s0
                lw  $s0,12($t1)
  #create  new  top  of  the  stack
addi  $sp,$sp,-4                 sw
$s0,0($sp)
```

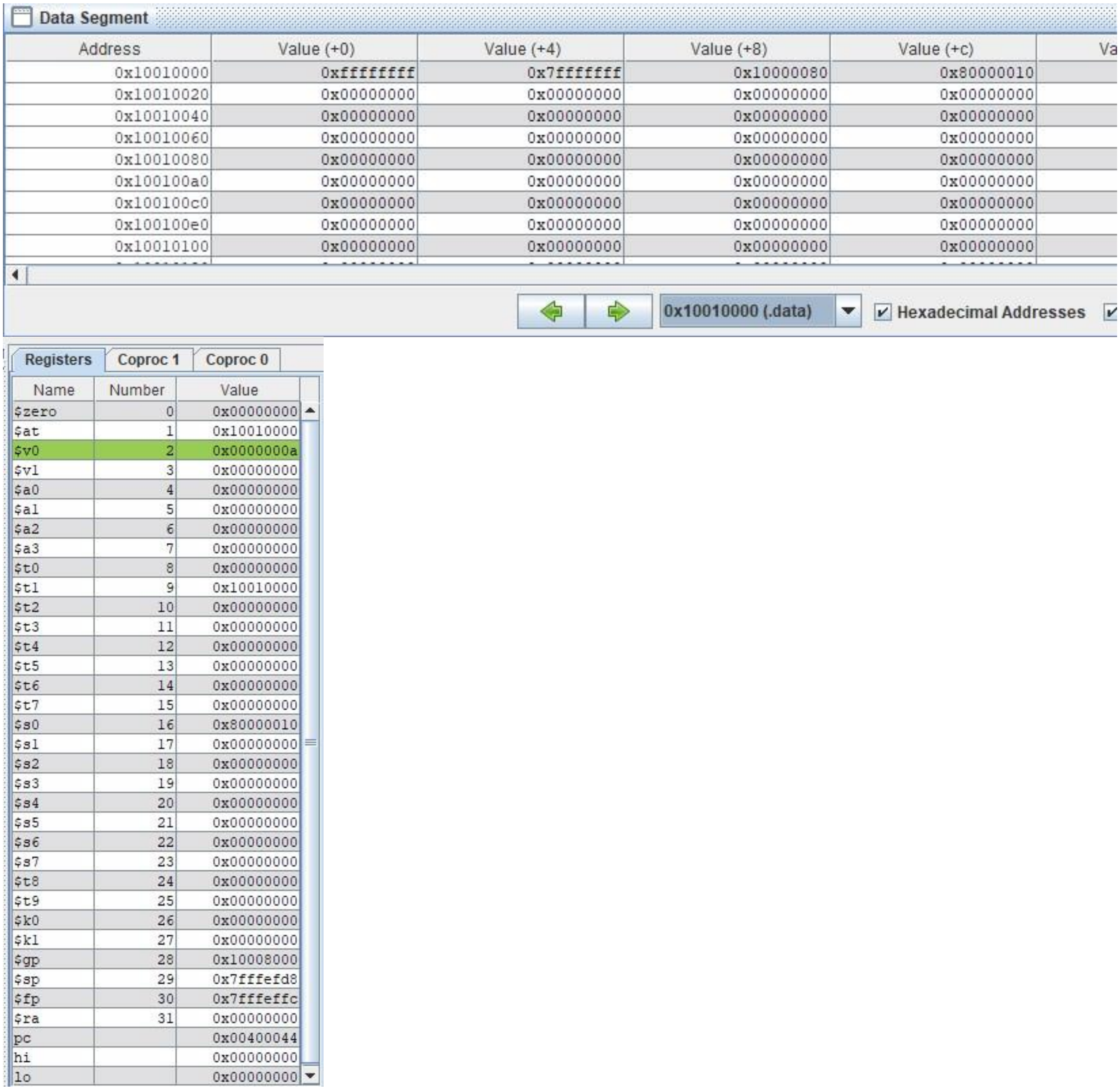
After execution of the program in MARS simulator, he displayed the following memory windows and register file:



Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7fffefc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x80000010	0x10000080
0x7fffefe0	0x7fffffff	0xffffffff	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x10010000
0x7ffff000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff0a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff0c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

current $sp ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Va
0x10010000	0xffffffff	0x7fffffff	0x10000080	0x80000010	
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x10010000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x80000010
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffefd8
$fp	30	0x7fffeffc
$ra	31	0x00000000
pc		0x00400044
hi		0x00000000
lo		0x00000000

Figure 2. Register file and memory windows in MARS simulator.

Based on the information displayed in **Figure 2.** memory windows and register file above, please answer the following questions

2.1.1 [15 points] What is the address of an integer that was **first** pushed on to stack?

ANSWER:

The address of an integer that was first pushed on to stack is 0x7fffefc4.

2.1.2 [15 points] What is the value in Hex and signed decimal of an integer that was **first** pushed on to stack?

ANSWER:

The value in hex of the integer is 0xffffffff. This is in signed decimal the equivalent of -1.

2.1.3 [15 points] What is the offset from FRAME POINTER to an integer that was **first** pushed on to stack?

ANSWER:

The offset from FRAME POINTER to an integer that was **first** pushed on to stack is the same as the frame pointer.
This means, 0x7FFFEFFC – 0x7FFFEFE4 = 0x18

2.2.1 [5 points] What is the address of an integer that was **Last** pushed on to stack?

ANSWER:

The address of the last value pushed on the stack is 0x7fffefc0 + 0x18 offset. This equals to 0x7fffeffdc. Therefore, the address of the last value on pushed on stack is 0x7fffeffdc.

2.2.2 [5 points] What is the value in Hex and signed decimal of an integer that was **Last** pushed on to stack?

ANSWER:

The hex value of the last value pushed on the stack is 0x80000010 and the decimal signed value is -2147483632.

2.2.3 [5 points] What is the offset from FRAME POINTER to an integer that was **Last** pushed on to stack?

ANSWER:

The offset from the frame pointer of the integer that was last pushed on the stack is 0x7FFFEFFC– 0x7fffefd8= 0x24 in hex.

2.3 [5 points] Based on the data shown Figure 2.,Can you determine if Frame pointer points to an **address** *or a* **value?** Please circle around your answer. Please explain.

ANSWER:

The frame pointer points the starting address of the stack. This is different from the stack which has the same value as the frame pointer in the beginning, but changes as things are pushed onto the stack. As shown from figure 2, we see that the address of the frame pointer is 0x7FFFEFFC.