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:

I will neither give nor recieve unathorized assistance on this exam. I will only use one computing device to perform this test

Azwad Shameem

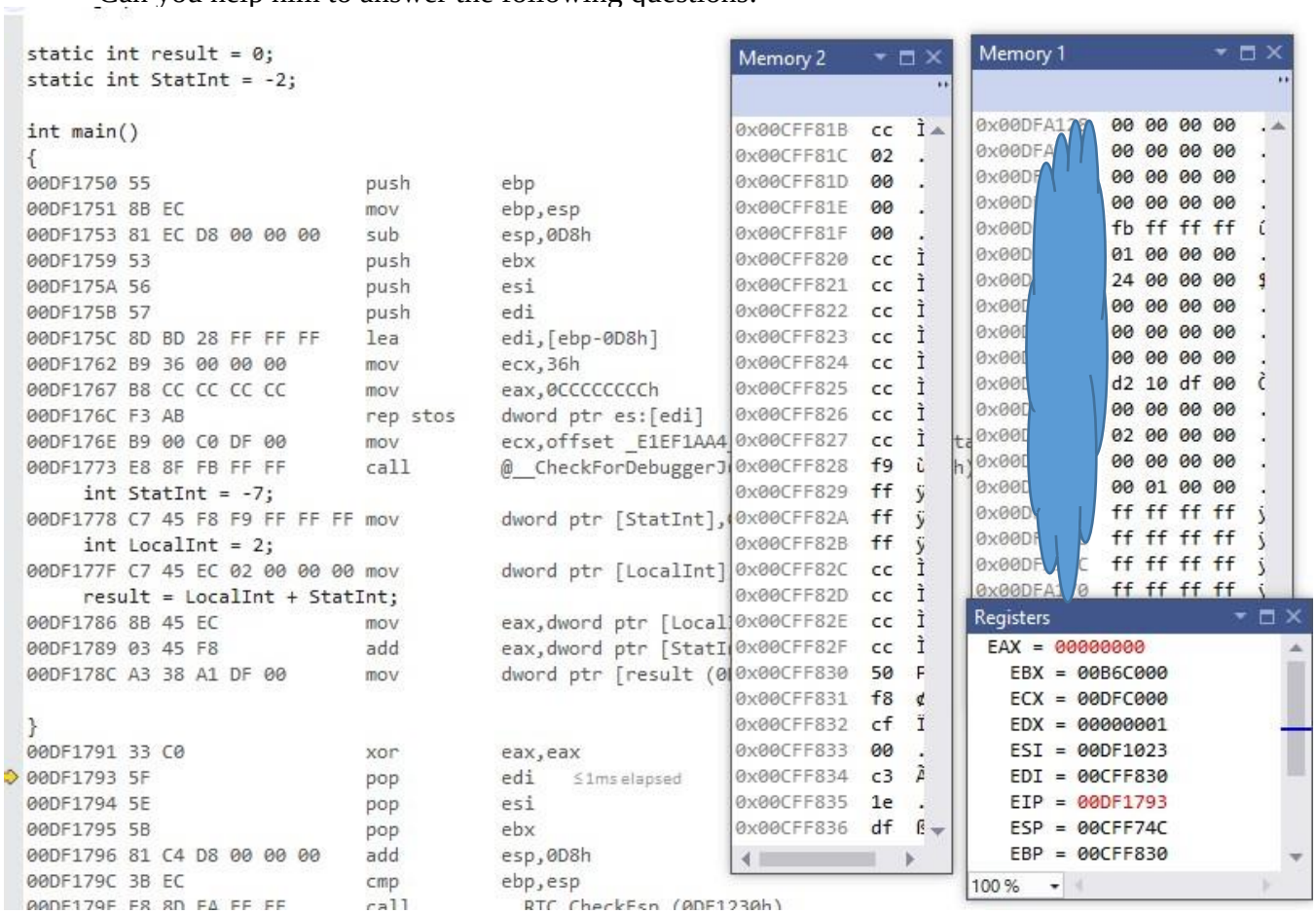
This quiz has 6 pages.

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:



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

The address of the instruction that will be executed in the next instance is the address of the register EIP, which is **0x00DF1793**. In addition, the yellow arrow marker in the image above also tells us the address of the next executed instruction.

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 ……. 0x00CFF828*
Value of ***StatInt*** *in memory is 0xFFFFFFF9= -7(dec)*
Please justify your answers.

To obtain the address of StatInt we need to find EBP and the offset. To get EBP, we can look in memory window 2 and to get the offset we can look at the machine instruction at C7 45 F8 F9 FF FF FF, in which the third value in this instruction represents the offset.

Now that we have EBP and offset we can find the address of StatInt.
0x00CFF830 + F8= 0x00CFF828

To find the value of StatInt in memory we can use the last four values in machine instruction C7 45 F8 F9 FF FF FF.

Value: FF FF FF F9 = 1111 1111 1111 1111 1111 1111 1111 1001
2's complement 0000 0000 0000 0000 0000 0000 0000 0110 +1 = -7 (dec)

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 ……. 0x00CFF81C*
Value of ***LocalInt*** *in memory is…. 0x00000002 = 2 (dec)*
Please justify your answers.

To find the address of LocalInt we need to find EBP and offset. EBP is given in memory window 2 and offset is the third value in the machine instruction for the LocalInt variable C7 45 EC 00 00 00 02.

Now that we have EBP and offset.
0x00CFF830 + EC = 0x00CFF81C

To find the value of StatInt in memory we can use the last four values in machine instruction for the LocalInt variable, C7 45 EC 00 00 00 02.
Value: 00 00 00 02 = 0010 0000 0000 0000 0000 0000 0000 0000
2's complement 1101 1111 1111 1111 1111 1111 1111 1111 +1 = 2 (dec)

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 ……. 0x00DFA138*
of ***result*** *in memory is 0xFFFFFFFB*
Please justify your answers.

To get the address of result, we can look at the instruction where the result of adding the result of adding is stored into static variable result.

We know that by adding -7 and 2, we get -5 and -5 in hex is FF FF FF FB. Furthermore, we can also see this value in the code after the mov instruction which is highlighted being stored in little endian at the address 0x00DFA138.

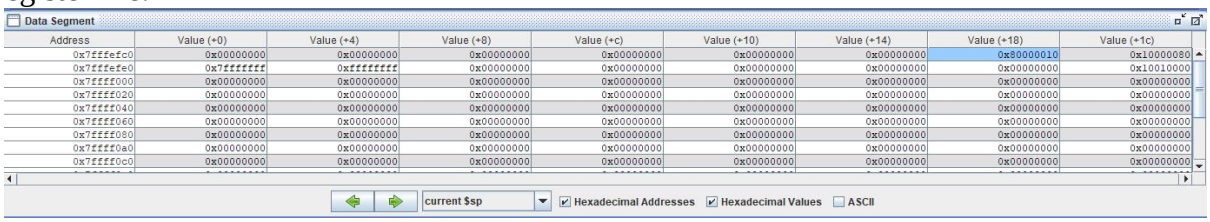
5: 0000 0000 0000 0000 0000 0000 0000 0101
2's complement of 5: 1111 1111 1111 1111 1111 1111 1111 1010 + 1
1111 1111 1111 1111 1111 1111 1111 1011 (B: 11)
-5 (dec): 0xFFFFFFFB

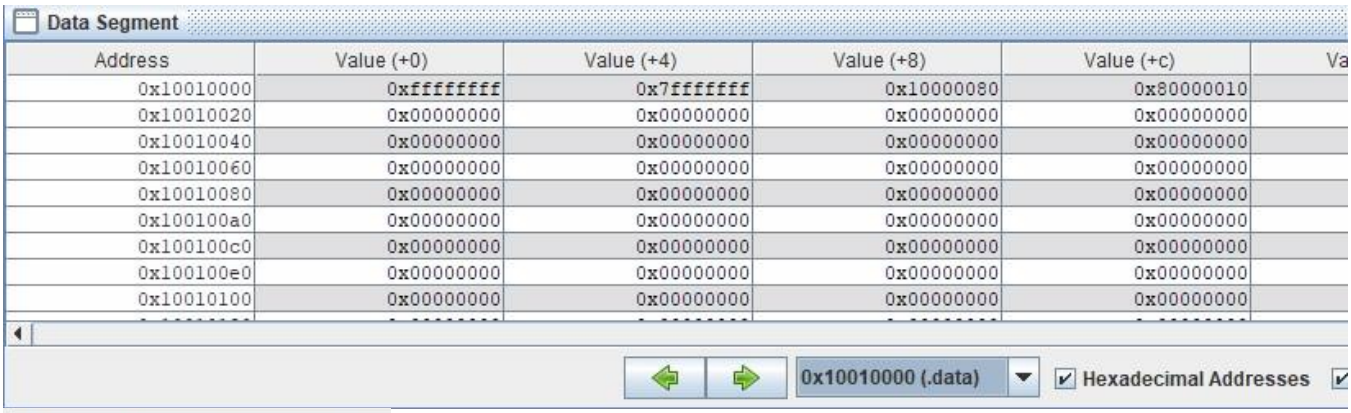
Question 2.

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:





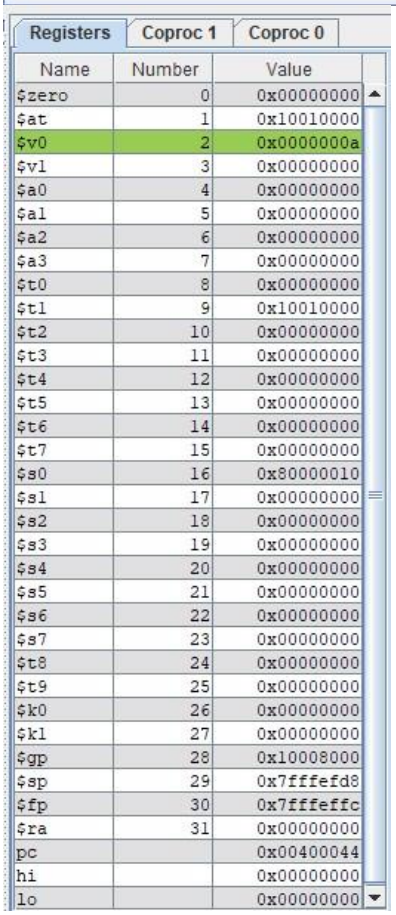


Figure 2. Register file and memory windows in MARS simulator.

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?

To find the number that was first pushed to the stack you need to look at the value in the $fp register. We also know that the first value is at the bottom of the stack due to the nature of a stack which is LIFO.

Address of the first integer that is pushed on to stack is
0x7fffefec + 0x4 (offset) **= 0x7fffefe4**

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

To get the hex value we can look at the value being stored in the address we determined in the previous question. In addition, since MIPS uses BIG endian notation the value stored is in the original hex format, so we can use two's complement to get the signed decimal.
Hex value: **0xffffffff**
2's complement of 0xffffffff: 1111 1111 1111 1111 1111 1111 1111 1111
0000 0000 0000 0000 0000 0000 0000 0000 + 1 = -1 (dec)
Signed dec: **-1**

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

To find the offset from the frame pointer to an integer that was first pushed to the stack we can use the frame pointer minus the address of the first integer pushed to the stack.
0x7fffeffc - 0x7fffefe4 = **-24** (dec)
1 * 16 + 8 = 24

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

The address of the integer that was last pushed to stack can be found by looking at the current $sp which is 0x7fffefd8. Also, since we know stack follows the LIFO structure, we know that the current $sp register holds the address of the value on the top of the stack, which is also the one that was last pushed to the stack. As a result, the address of the integer last pushed to the stack is **0x7fffefd8**.

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

By using the address from the previous question, we can look at the corresponding address in the data segment figure to get the value of the last pushed integer to the stack.
Hex: **0x80000010**
0x80000010 = 1000 0000 0000 0000 0000 0000 0001 0000 = $-2^{31} + 2^4$
Signed dec: $-2^{31} + 2^4$ = **-2147483632**

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

To find the offset from the frame pointer to an integer we can take the frame pointer and subtract it from the address of the integer last pushed to the stack.
0x7fffeffc – 0x7fffefd8 = -36 (dec)
Offset: **-36** (dec)
This can be affirmed by reading the code and noticing that it actually results in -24 – 4 - 4 – 4 = -36.

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.

To determine if a frame pointer points to an address we can look at $fp which stores the address 0x7fffeffc and then look at the data segment figure where this address is located. Then we can tell at this address that the value is stored in 0x10010000, which is the address of an array. Therefore, we know that the frame pointer is pointing to an **address** because the address of $fp takes us to an array instead of a value.