Last NAME: *Quiz 3 , November 8, 2021* *First Name:*

Computer Science
C.Sc. 342

Quiz No.3 To be performed

12:00-1:40PM AND 5:00-6:15 PM ***on*** November 8, 2021

Submit by 6:15 PM 11/08/2021 on 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 8 pages.

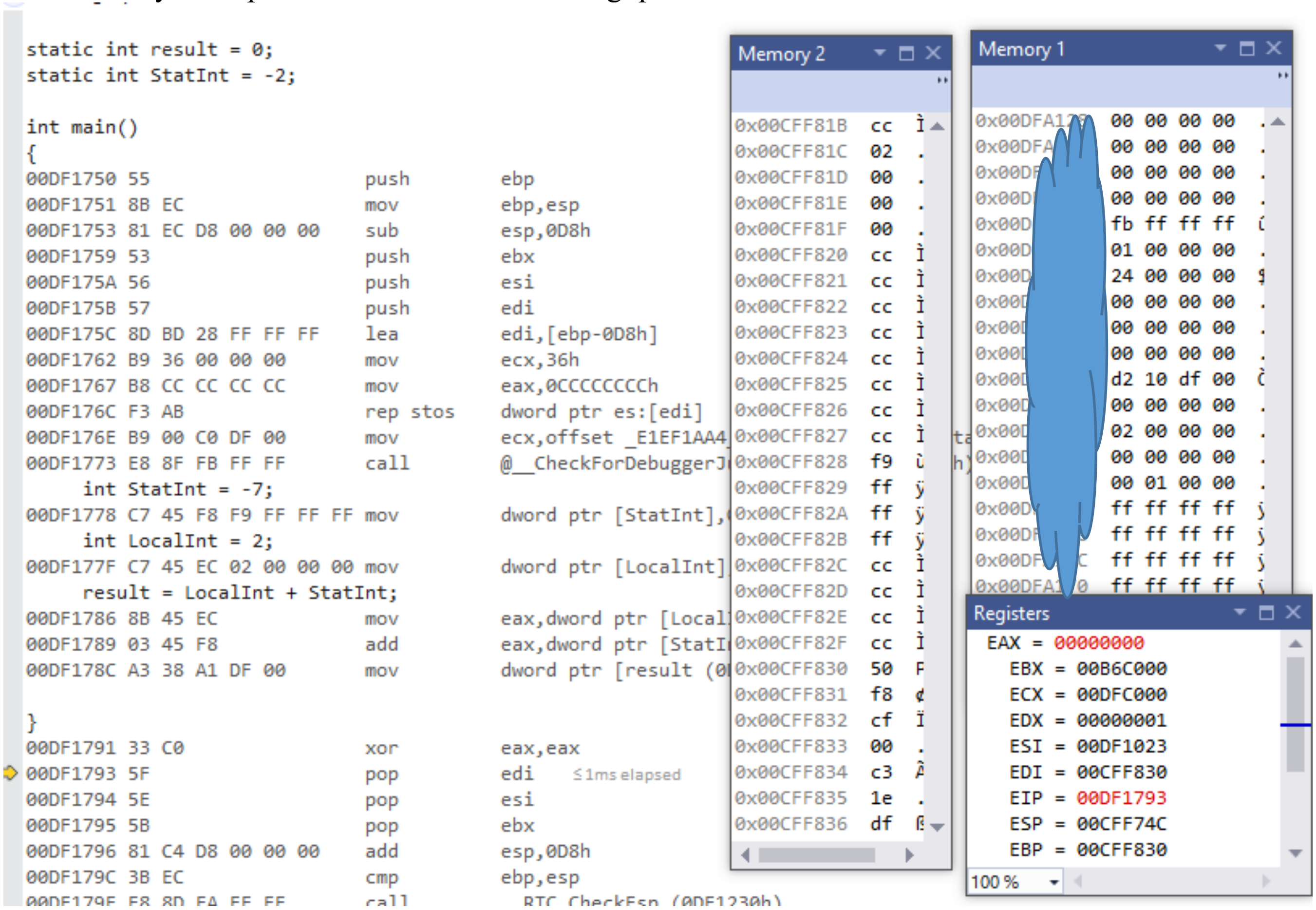
Question	Your Grade	Max Grade
1.1		5
1.2		10
1.3		10
1.4		10
2.1		5
2.2		5
2.3		10
2.4		10
3.1.1		5
3.1.2		5
3.1.3		5
3.2.1		5
3.2.2		5
3.2.3		5
3.3		5

Total: 100

Last NAME: *Quiz 3 , November 8, 2021* *First Name:*

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  ø
0x00CFF832  cf  Ï
0x00CFF833  00  .
0x00CFF834  c3  Ã
0x00CFF835  1e  .
0x00CFF836  df  ß
```

Memory 1

```
0x00DFA12     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
```

Last NAME: *Quiz 3 , November 8, 2021* *First Name:*

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

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.

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.

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.

Question 2.

A student compiled his C code using compiler:

"GCC: (GNU) 4.8.5 20150623 (Red Hat 4.8.5-11)"
Target processor: x64, i7

Figure 1. Dump of assembly code in GDB:

```
(gdb) disassemble
Dump of assembler code for function main:
   0x00000000004004ed <+0>:     push   %rbp
   0x00000000004004ee <+1>:     mov    %rsp,%rbp
=> 0x00000000004004f1 <+4>:     movl   $0xffffffff,-0x4(%rbp)
   0x00000000004004f8 <+11>:    movl   $0x7ffffff,-0x8(%rbp)
   0x00000000004004ff <+18>:    movl   $0x8000000,-0xc(%rbp)
   0x0000000000400506 <+25>:    movl   $0x0,-0x10(%rbp)
   0x000000000040050d <+32>:    mov    -0x8(%rbp),%eax
   0x0000000000400510 <+35>:    mov    -0x4(%rbp),%edx
   0x0000000000400513 <+38>:    add    %edx,%eax
   0x0000000000400515 <+40>:    mov    %eax,-0x10(%rbp)
   0x0000000000400518 <+43>:    mov    0x200b0e(%rip),%eax
   0x000000000040051e <+49>:    mov    -0x8(%rbp),%edx
   0x0000000000400521 <+52>:    sub    %eax,%edx
   0x0000000000400523 <+54>:    mov    %edx,%eax
   0x0000000000400525 <+56>:    mov    %eax,-0x14(%rbp)
   0x0000000000400528 <+59>:    mov    $0x0,%eax
   0x000000000040052d <+64>:    pop    %rbp
   0x000000000040052e <+65>:    retq
End of assembler dump.
```

Question 2.1 [5 points] Do you have enough information to determine the content of register %eax after executing instruction at offset +40 in the dump of assembly code shown in Figure 1.?

Question 2.2 [5 points] Please compute the address of the static variable referenced in this dump of assembly code show in Figure 1.?

Question 2.3 [10 points] In GDB environment you typed the following commands:

```
(gdb) x $rbp - 4
0x7fffffffdcac:  0xffffffff
(gdb) x $rbp - 8
0x7fffffffdca8: 0x07ffffff
```

Can you determine the content of register %rbp. ***YES or NO***?

If *No* go to next question **ELSE** Please determine the content of register *%rbp.*

Question 2.4 [10 points] Shown below partial stack memory for dump of assembly code shown in Figure 1?

```
0x7fffffffdca4:  0x00    0x00    0x00    0x08    0xff    0xff    0xff    0x07
0x7fffffffdcac:  0xff    0xff    0xff    0xff    0x00    0x00    0x00    0x00
0x7fffffffdcb4:  0x00    0x00    0x00    0x00    0x35    0xcb    0xa3    0xf7
```

Please determine the value of variable on stack at offset -12 decimal from base pointer %rbp. Use the value for Register %rbp you obtained in question 2.3.

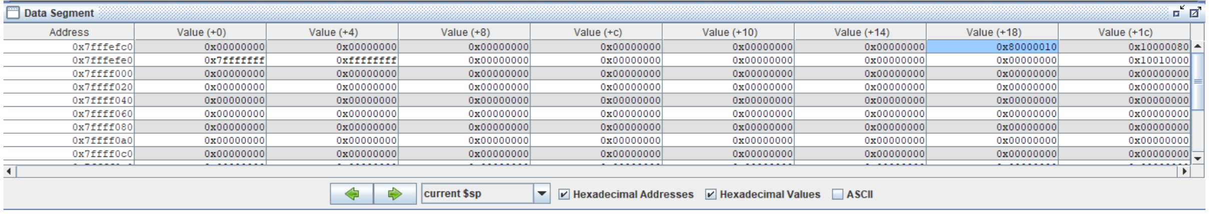
Last NAME: Quiz 3 , November 8, 2021   First Name:

Question 3.

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

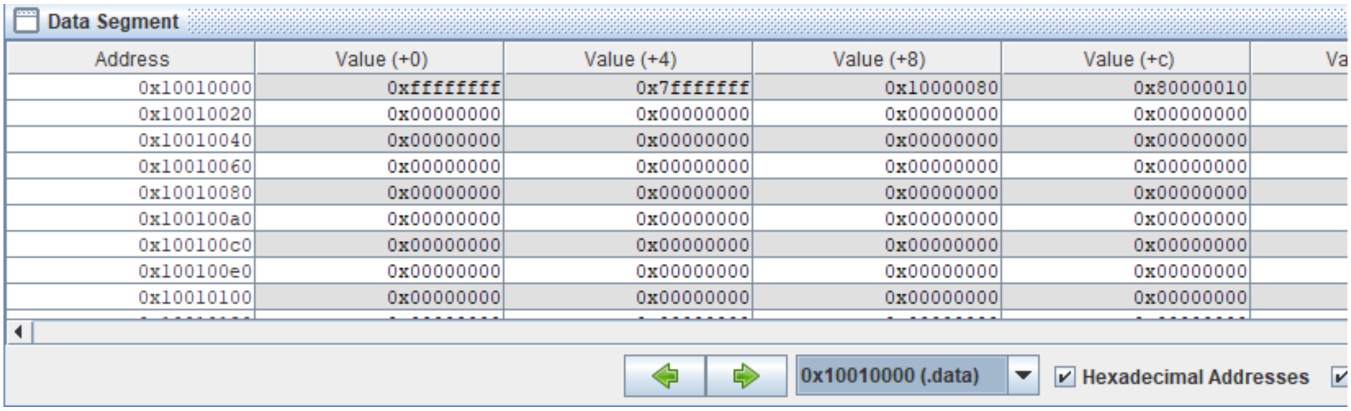
```
 .data
array1:  .word -1,0x7fffffff,0x10000080,0x80000010
.text
    main:
               la $t1,array1
# create Frame pointer
                    add $fp,$zero,$sp
#Store the address of the first element on stack
using frame pointer
                    sw $t1,0($fp)
#allocate memory on Stack for 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:



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

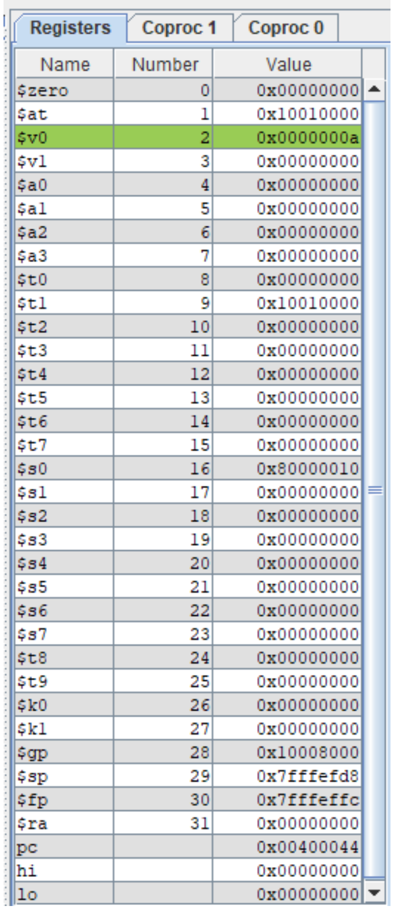
current $sp ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
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

0x10010000 (.data) ☑ Hexadecimal Addresses

Last NAME: *Quiz 3 , November 8, 2021* *First Name:*



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

3.1.1 [5 points] What is the address of an integer that was **first** pushed on to stack?

3.1.2 [5 points] What is the value in Hex and signed decimal of an integer that was **first** pushed on to stack?

3.1.3 [5 points] What is the offset from FRAME POINTER to an integer that was **first** pushed on to stack?

3.2.1 [5 points] What is the address of an integer that was **Last** pushed on to stack?

3.2.2 [5 points] What is the value in Hex and signed decimal of an integer that was **Last** pushed on to stack?

3.2.3 [5 points] What is the offset from FRAME POINTER to an integer that was **Last** pushed on to stack?

3.3 [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.