**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:**

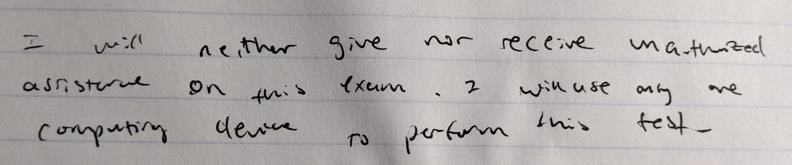
**NO CORRECTIONS ARE ALLOWED IN ANSWER CELLS!!!!!**

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:



Please hand write and sign here: **Chue Zhang**

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

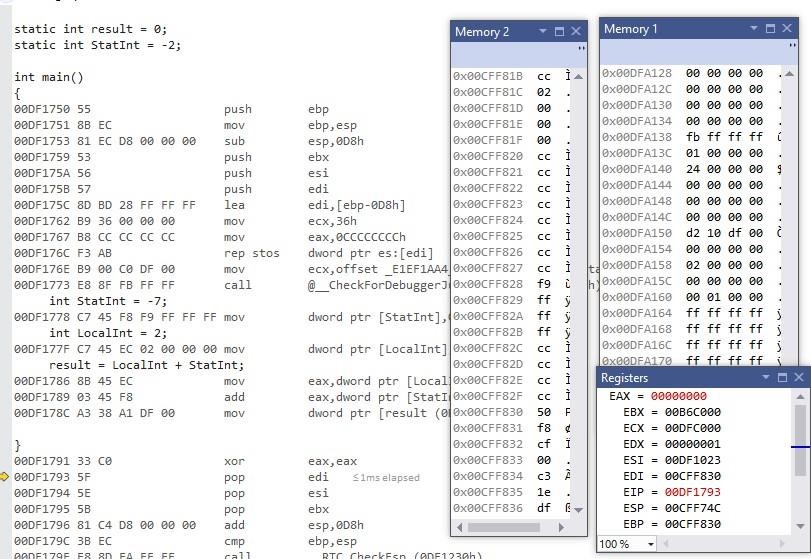
**Question 1.**

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? EIP is next address to be executed so 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.*

Yes, from the machine code, statINT is c7 45 f8 f9 ff ff ff. f9 we can see in memory window 2 so the address is 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.*

Yes, Localint variable is C7 45 EC 02 00 00 00, we can see 02 00 00 00 in memory window 2 so the address is 0x00cff81c

**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.*

Yes, Add -7 and 2 together you get -5 which when converted to hex, we get fbffffff and in memory window 1 we can see that written but you blurred our the addresses in mem window 1 for some reason. You can also find address in 3 lines below result = statint + local int which is df a1 38 83

**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.?

Yes, the first two values stored in stack with offset from rbp is 0xffffffff and 0x7fffffff. Assembly code says to +4 offset moved to edx and +8 offset moved to eax then add them together and store in eax. 0xffffffff and 0x7fffffff add together is 0x7ffffffe

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

To calculate the address of the static variable we have to add the offset of 0x200b0e to the base address to the register %rip. This will give us the address of the static variable.

***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.*

No, rbp always changes

***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.

At offset -12, the variable is 0x80000000, 0x07ffffff has -8 offset and minus 4 offset to that and you get 0x7fffffffdca4 which we see in dump.

**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

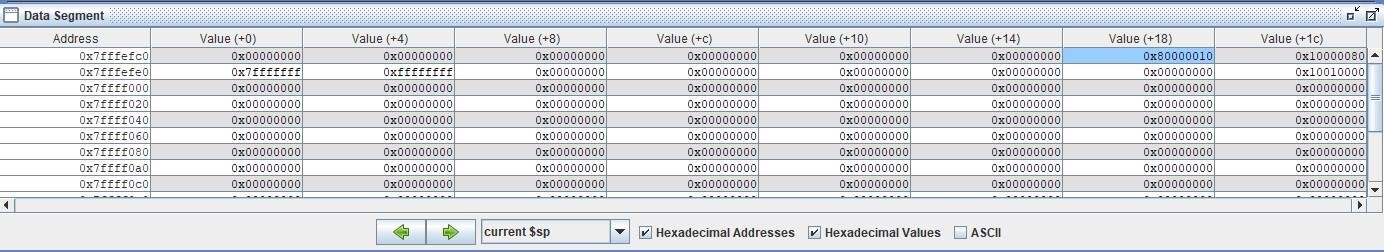
addi $sp,$sp,-4  sw

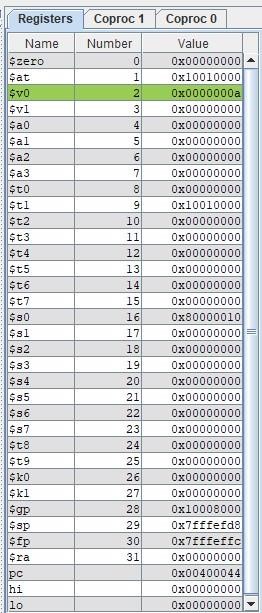
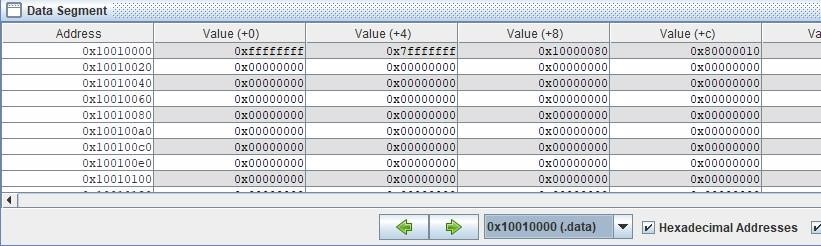
$s0,0(sp)

#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:





**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?

Address of integer first pushed is 0x7fffefe0 + 0x4 = 0x7fffefe4. Last value in stack is first value pushed.

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

First integer pushed onto stack is 0xffffffff which is -1.

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

0x7fffeffc – 0x7fffefe4 = 0x18

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

0x7fffefc0 + 0x18 = 0x7fffefd8

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

$sp = 0x80000010 = -2146483632

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

0x7fffeffc – 0x7fffefd8 = 0x24

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.

Stack pointer and frame pointer initially have same starting address therefore it points towards an address which is 0x777feffc