CSE 2421 AU21 Assembler Execution Homework 22 points

Your name ___________________________________

Start in class, due midnight the next day. You may work in groups, but each person must turn in their own.

Given the partial machine state on the next page, walk through each instruction and determine what changes when each instruction is executed.

If a register changes:

- List the name of the register that changed
- Give the new value it has
- Change the value in the register state table (top of next page)

If a memory location changes:

- list the address in hex
- Give the new value
- Give the size in bytes of the new value
- Change the value at that location in the memory table (bottom of next page)

Numbers are decimal unless marked with a leading 0x to denote hex. You do not need to indicate changes to the condition codes such as zero and overflow in this homework. When done, answer the 3 questions on this side of the paper.

For example…

If the given machine state is:

%rax = 1
%rsp = 0xC0B0A090

You would add the text in italics in the comment

pushq %rax # ___*memory 0xC0B0A088 = 1 (8bytes), %rsp = 0xC0B0A088*

…end of example.

Answer these *after* doing the questions on the other side

Q1: What will happen if ret is the next instruction? Why?

Seg fault – the code overwrote the return address and the old base pointer, either of which will instantly or shortly be fatal

Q2: Does this code properly initialize the array of 2 longs to 0 and 1?

No, it overwrites other data.

Q3: If not, what single fix makes this code initialize the array of two longs to 0 and 1?

Move the leaq (%rsp), %rax down one line to below the subtract

For this homework, start with this partial machine state: (Keep old values by crossing them out and put new values in after the crossed out old value). Use the ~~strikethrough~~ font effect to cross out old values as shown for rax (current value is 1, old value was 256). This table is graded.

Register	Value
rax	~~256~~ ~~1~~ 0xC0B0A088
rdi	~~0~~ 1
rsp	~~0xC0B0A090~~ ~~0xC0B0A088~~ ~~0xC0B0A078~~ ~~0xC0B0A088~~ 0xC0B0A090
rbp	~~0xC0B0A0A0~~ ~~0xC0B0A088~~ 0

Assume the instruction "**call hw5**" was the last instruction that executed prior to this. Right away this tells you something about what is at the top of the stack – fill in the memory table appropriately. Execute each line of the following code and tell what changed:

Most are 1 point each unless marked (some have 2 responses) total of 10

hw5:

pushq %rbp # _ _%rsp = 0xC0B0A088, memory 0xC0B0A088 = 0xC0B0A0A0 (8 bytes) [2pts] _

movq %rsp, %rbp #__ _%rbp = 0xC0B0A088 ____

leaq (%rsp), %rax #__ %rax = 0xC0B0A088 ____

subq $16, %rsp #______%rsp = 0xC0B0A078 ____

movq %rdi, (%rax, %rdi, 8) #__ memory 0xC0B0A088 = 0 (8 bytes) [trashes old rbp in stack] _______

addq $1, %rdi #_______ %rdi = 1 ____

movq %rdi, (%rax, %rdi, 8) #_____ memory 0xC0B0A090 = 1 (8 bytes) [trashes return address in stack] __

leave #_[2pts] %rsp=0xC0B0A088 [optional answer], %rbp = 0, %rsp = 0xC0B0A090 [last two are required] __

This table is a scratch pad where you keep track of what is in memory. It will be graded. The addresses are on 8 byte boundaries. If you know the meaning of a location but not its value, write that meaning (such as "return address"). If you know both, tag the numeric value with the meaning, such as "old rbp." Use the ~~strikethrough~~ font effect on the old value if you change any value

Address	Value
0xC0B0A0A0	
0xC0B0A098	
0xC0B0A090	~~Return address~~ 1
0xC0B0A088	~~0xC0B0A0A0 Old rbp~~ 0
0xC0B0A080	
0xC0B0A078	
0xC0B0A070	

Table rows are 1 point per row