Assignment #2

Problem 2.1: **riscv stack frame**

a)

1. main: ; function entry label

2. addi sp,sp,-48
   - 48-byte allocation stack frame
   - sp := sp - 48

3. sd s0,40(sp)
   - sd = store doubleword (64-bits)
   - save the old frame pointer (s0) into the stack at address sp + 40
   - s0 is placed at sp+40

4. addi s0,sp,48
   - set up a frame pointer at s0 := sp + 48
   - Since sp was decremented earlier by 48, s0 takes its old value

5. mv a5,a0
   - mv is a pseudoinstruction;
   - mv a5, a0 = addi a5, a0, 0
   - copy the incoming a0 (argc) into temporary register a5

6. sd a1,-48(s0)
   - sd = Store Doubleword
   - Store the incoming a1 (argv pointer) at the address (s0 - 48)
   - Because s0 = old_sp, s0 - 48 = new_so, this stores the argv at the bottom of the frame

7. sw a5,-36(s0)
   - sw = store word (32 bits)
   - stores the 32 bit value in a5 (argc) at s0 - 36
   - s0 - 36 is a slot reserved for local variables / saved arg

8. sw zero,-20(s0)

- store 0 (register) into s0 - 20
- The local variable ‘rc’ is initialized to zero in memory

9. lw a5,-20(s0)
   - lw = load word
   - load the word from s0 - 20 into the register a5
   - This loads 0 into the variable a5

10. mv a0,a5
    - Move the value in a5 into the a0 register
    - a0 is the standard register to hold a functions return value

11. ld s0,40(sp)
    - Restore old s0 from the stack at s0 + 40 (64 bits)

12. addi sp,sp,48
    - deallocate stack frame sp := sp + 48

13. jr ra
    - jalr x0, ra, 0
    - jump to the address in ra; return

b)

Higher Addresses	
(above-frame / caller area)	
saved s0 (sd s0,40(sp))	<--- Address = s0 - 8 = sp + 40 (8 bytes)
(padding / alignment)	
rc (sw zero,-20(s0))	<--- Address = s0 - 20 = sp + 28 (4 bytes)
argc (sw a5,-36(s0))	<--- Address = s0 - 36 = sp + 12 (4 bytes)
argv (sd a1,-48(s0))	<--- Address = s0 - 48 = sp + 0 (bottom)
(pointer, 8 bytes, 64-bit)	<--- Address = sp + 0 (8 bytes)
Lower addresses	