Assignment 8

Q1

Given the following: %rdi: 0x3 %rsi: 0x7 Code:

leal (%rdi,%rsi,1),%eax

What will be the value of %eax after the instruction has executed? [eax]

WRITE YOUR ANSWER IN HEX and use CAPITAL LETTERS.

%eax = %rdi + (%rsi * 1) = %rdo + %rsi = 0x3 + 0x7 = 0xa

0x3 + 0x7 = 0xA

Q2

Following is code for a unknown function.

```
4005cb: 8b 06                     mov    (%rsi),%ebx
4005cd: 01 07                     add    %ebx,(%rdi)
4005cf: c3                        ret
```

Assuming the two parameters are called x and y; Pick the declaration for the function that you think is most fitting? [action]

- ~~int sum (int x, int y)~~
- ~~int* sum (int* x, int* y)~~
- void sum (int* x, int* y);

Q3

Given the following assembly code:

00000000004005c0 : 4005c0: 66 39 f7 cmp %si,%di 4005c3: 7f 03 jg 4005c8 <fun+0x8> 4005c5: 89 f0 mov %esi,%eax 4005c7: c3 ret
4005c8: 89 f8 mov %edi,%eax 4005ca: c3 ret

1) According to the Assembly, how many input parameters does this function take? 2
2) What is the datatype of the parameter(s)? This uses the %si and %di registers so this is 16 bit short?

Q4

Answer the following questions about the assembly code below: 1) If the value of %rsp is 0x00007FFFFFFFE108 at the start of the execution, what will the address of %rsp be after the first 4 lines (after the 4 push operations)? 0x00007FFFFFFFE108 - (8*4) = 0x7FFFFFFFE0E8

2) How many parameters does this function take? 8

3) The last parameters to this function is a pointer to a short (short *sp). In the assembly we can figure this out; What is the assembly command that tells us that the pointers points to a signed short? Just copy-paste the assembly command as your answer. movswl

4) What is the value of %rsp just before the ret command is executed? 0x00007FFFFFFFDB48

```
00000000004005d0 <sumsum>:
  4005d0: 41 55                  push   %r13
  4005d2: 41 54                  push   %r12
  4005d4: 55                     push   %rbp
  4005d5: 53                     push   %rbx
  4005d6: 41 89 d5               mov    %edx,%r13d       ; PARAMETER
  4005d9: 41 89 cc               mov    %ecx,%r12d       ; PARAMETER
  4005dc: 44 89 c5               mov    %r8d,%ebp        ; PARAMETER
  4005df: 44 89 cb               mov    %r9d,%ebx        ; PARAMETER
  4005e2: 8b 36                  mov    (%rsi),%esi      ; PARAMETER
  4005e4: e8 c7 ff ff ff         callq  4005b0 <sum>
  4005e9: 44 01 e8               add    %r13d,%eax
  4005ec: 44 01 e0               add    %r12d,%eax
  4005ef: 01 e8                  add    %ebp,%eax
  4005f1: 01 d8                  add    %ebx,%eax
  4005f3: 03 44 24 28            add    0x28(%rsp),%eax  ; PARAMETER ON STACK
  4005f7: 48 8b 54 24 30         mov    0x30(%rsp),%rdx  ; PARAMETER ON STACK
  4005fc: 0f bf 12               movswl (%rdx),%edx
  4005ff: 29 d0                  sub    %edx,%eax
  400601: 5b                     pop    %rbx
  400602: 5d                     pop    %rbp
  400603: 41 5c                  pop    %r12
  400605: 41 5d                  pop    %r13
  400607: c3                     ret
```

Q5

```c
int sum_ar( int* arr, int len ) {
    int ret = 0;
    int i = 0;

    while (i < len) {
        ret = ret + arr[ i ];
        i++;
    }

    return ret;
}
```

```
400608: ba 00 00 00 00 mov $0x0,%edx # i = 0 40060d: b8 00 00 00 00 mov
$0x0,%eax # ret = 0 400612: 39 f2 cmp %esi,%edx # i < len 400614: 7d 0b jge
400621 <sum_ar+0x19> # while () 400616: 48 63 ca movslq %edx,%rcx # ret
+ arr[i] 400619: 03 04 8f add (%rdi,%rcx,4),%eax # ret + arr[i] 40061c: 83 c2
01 add $0x1,%edx # i++ 40061f: eb f1 jmp 400612 <sum_ar+0xa> # while()
400621: c3 ret
```