CS 210 PR Problem Set Part A

Jeong Yong Yang

TOTAL POINTS

35 / 39

QUESTION 1

Problem 1 6 pts

1.1 foo1 **1 / 1**

✓ **- 0 pts** **Correct**

- 1 pts Incorrect

1.2 foo2 **1 / 1**

✓ **- 0 pts** **Correct**

- 1 pts Incorrect

1.3 foo3 **1 / 1**

✓ **- 0 pts** **Correct**

- 1 pts Incorrect

1.4 foo4 **1 / 1**

✓ **- 0 pts** **Correct**

- 1 pts Incorrect

1.5 foo5 **1 / 1**

✓ **- 0 pts** **Correct**

- 1 pts Incorrect

1.6 foo6 **1 / 1**

✓ **- 0 pts** **Correct**

- 1 pts Incorrect

QUESTION 2

Problem 2 9 pts

2.1 A **3 / 3**

✓ **- 0 pts** **Correct**

- 3 pts Incorrect

2.2 B **0 / 3**

- 0 pts Correct

✓ **- 3 pts** **Click here to replace this description.**

2.3 C **3 / 3**

✓ **- 0 pts** **Correct**

- 3 pts Incorrect

QUESTION 3

Problem 3 10 pts

3.1 int val = ________________ **2 / 2**

✓ **- 0 pts** **Correct**

- 2 pts Incorrect

3.2 first blank of : for (___________; _________; i++) { **2 / 2**

✓ **- 0 pts** **Correct**

- 2 pts Incorrect

3.3 second blank of : for (___________; _________; i++) { **2 / 2**

✓ **- 0 pts** **Correct**

- 2 pts Incorrect

3.4 blank for first line of for loop: _____________________; **2 / 2**

✓ **- 0 pts** **Correct**

- 2 pts Incorrect

3.5 blank for second line of for loop: ___________________; **2 / 2**

✓ **- 0 pts** **Correct**

- 2 pts Incorrect

QUESTION 4

Problem 5 14 pts

4.1 A: 0x7fffffffe000: int hex value **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect. Expected "0x6c6c6548".

4.2 A: 0x7fffffffe000: Description **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect. Expected "buf[0-3]" or something similar.

4.3 A: 0x7fffffffe004: int hex value **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect. Expected "0x6f57206f".

4.4 A: 0x7fffffffe004: Description **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect.

4.5 A: 0x7fffffffe008: int hex value **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect. Expected "0x21646c72".

4.6 A: 0x7fffffffe008: Description **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect.

4.7 A: 0x7fffffffe00c: int hex value **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect. Expected "0x21212121".

4.8 A: 0x7fffffffe00c: Description **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect. Expected "i" to be mentioned.

4.9 A: 0x7fffffffe010: int hex value **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect

4.10 A: 0x7fffffffe010: Description **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect.

4.11 A: 0x7fffffffe014: int hex value **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect.

4.12 A: 0x7fffffffe014: Description **1 / 1**

✓ **- 0 pts Correct**

- 1 pts Incorrect.

4.13 Part B:output from the printf in the foo function: **1 / 2**

- 0 pts Correct

- 2 pts Incorrect. Expected "0xffffe00c 0x21212121"

✓ **- 1 pts Partially correct.**

Problem 1: 6 Points

Match each of the assembler routines on the left with the equivalent C function on the right.

```
foo1:
       lea    0xd(%rdi),%edx
        lea    0x10(%rdi),%eax
        test   %edx,%edx
        cmovns %edx,%eax
        sar    $0x2,%eax
        retq

foo2:
        lea    0xf(%rdi),%eax
        test   %edi,%edi
        cmovns %edi,%eax
        sar    $0x4,%eax
        retq

foo3:
        mov    %edi,%eax
        sar    $0x1f,%eax
        retq

foo4:
        mov    $0x0,%eax
        retq

foo5:
        imul   $0xe,%edi,%eax
        retq

foo6:
        mov    %edi,%eax
        shr    $0x1f,%eax
        retq
```

```
int choice1(int x)
{
  return x / 16;
}

int choice2(int x)
{
  return 14 * x;
}

int choice3(int x)
{
  return (x << 31) & 1;
}

int choice4(int x)
{
  return (x < 0);
}

int choice5(int x)
{
  return (x + 13) /4;
}

int choice6(int x)
{
  return (x >> 31);
}
```

Fill in your answers here:

foo1 corresponds to choice 5.
foo2 corresponds to choice 1.
foo3 corresponds to choice 6.
foo4 corresponds to choice 3.
foo5 corresponds to choice 2.
foo6 corresponds to choice 4.

1.1 foo1 **1 / 1**

- ✓ **- 0 pts** Correct
- **- 1 pts** Incorrect

Problem 1: 6 Points

Match each of the assembler routines on the left with the equivalent C function on the right.

```
foo1:
       lea    0xd(%rdi),%edx
        lea    0x10(%rdi),%eax
        test   %edx,%edx
        cmovns %edx,%eax
        sar    $0x2,%eax
        retq

foo2:
        lea    0xf(%rdi),%eax
        test   %edi,%edi
        cmovns %edi,%eax
        sar    $0x4,%eax
        retq

foo3:
        mov    %edi,%eax
        sar    $0x1f,%eax
        retq

foo4:
        mov    $0x0,%eax
        retq

foo5:
        imul   $0xe,%edi,%eax
        retq

foo6:
        mov    %edi,%eax
        shr    $0x1f,%eax
        retq
```

```
int choice1(int x)
{
  return x / 16;
}

int choice2(int x)
{
  return 14 * x;
}

int choice3(int x)
{
  return (x << 31) & 1;
}

int choice4(int x)
{
  return (x < 0);
}

int choice5(int x)
{
  return (x + 13) /4;
}

int choice6(int x)
{
  return (x >> 31);
}
```

Fill in your answers here:

foo1 corresponds to choice 5.
foo2 corresponds to choice 1.
foo3 corresponds to choice 6.
foo4 corresponds to choice 3.
foo5 corresponds to choice 2.
foo6 corresponds to choice 4.

1.2 foo2 **1 / 1**

✓ **- 0 pts** Correct

- 1 pts Incorrect

Problem 1: 6 Points

Match each of the assembler routines on the left with the equivalent C function on the right.

```
foo1:
       lea    0xd(%rdi),%edx
        lea    0x10(%rdi),%eax
        test   %edx,%edx
        cmovns %edx,%eax
        sar    $0x2,%eax
        retq

foo2:
        lea    0xf(%rdi),%eax
        test   %edi,%edi
        cmovns %edi,%eax
        sar    $0x4,%eax
        retq

foo3:
        mov    %edi,%eax
        sar    $0x1f,%eax
        retq

foo4:
        mov    $0x0,%eax
        retq

foo5:
        imul   $0xe,%edi,%eax
        retq

foo6:
        mov    %edi,%eax
        shr    $0x1f,%eax
        retq
```

```
int choice1(int x)
{
  return x / 16;
}

int choice2(int x)
{
  return 14 * x;
}

int choice3(int x)
{
  return (x << 31) & 1;
}

int choice4(int x)
{
  return (x < 0);
}

int choice5(int x)
{
  return (x + 13) /4;
}

int choice6(int x)
{
  return (x >> 31);
}
```

Fill in your answers here:

foo1 corresponds to choice 5.
foo2 corresponds to choice 1.
foo3 corresponds to choice 6.
foo4 corresponds to choice 3.
foo5 corresponds to choice 2.
foo6 corresponds to choice 4.

1.3 foo3 **1 / 1**

✓ **- 0 pts** Correct

- 1 pts Incorrect

Problem 1: 6 Points

Match each of the assembler routines on the left with the equivalent C function on the right.

```
foo1:
       lea    0xd(%rdi),%edx
        lea    0x10(%rdi),%eax
        test   %edx,%edx
        cmovns %edx,%eax
        sar    $0x2,%eax
        retq

foo2:
        lea    0xf(%rdi),%eax
        test   %edi,%edi
        cmovns %edi,%eax
        sar    $0x4,%eax
        retq

foo3:
        mov    %edi,%eax
        sar    $0x1f,%eax
        retq

foo4:
        mov    $0x0,%eax
        retq

foo5:
        imul   $0xe,%edi,%eax
        retq

foo6:
        mov    %edi,%eax
        shr    $0x1f,%eax
        retq
```

```
int choice1(int x)
{
  return x / 16;
}

int choice2(int x)
{
  return 14 * x;
}

int choice3(int x)
{
  return (x << 31) & 1;
}

int choice4(int x)
{
  return (x < 0);
}

int choice5(int x)
{
  return (x + 13) /4;
}

int choice6(int x)
{
  return (x >> 31);
}
```

Fill in your answers here:

foo1 corresponds to choice 5.
foo2 corresponds to choice 1.
foo3 corresponds to choice 6.
foo4 corresponds to choice 3.
foo5 corresponds to choice 2.
foo6 corresponds to choice 4.

1.4 foo4 **1 / 1**

✓ **- 0 pts** Correct

- 1 pts Incorrect

Problem 1: 6 Points

Match each of the assembler routines on the left with the equivalent C function on the right.

```
foo1:
       lea    0xd(%rdi),%edx
        lea    0x10(%rdi),%eax
        test   %edx,%edx
        cmovns %edx,%eax
        sar    $0x2,%eax
        retq

foo2:
        lea    0xf(%rdi),%eax
        test   %edi,%edi
        cmovns %edi,%eax
        sar    $0x4,%eax
        retq

foo3:
        mov    %edi,%eax
        sar    $0x1f,%eax
        retq

foo4:
        mov    $0x0,%eax
        retq

foo5:
        imul   $0xe,%edi,%eax
        retq

foo6:
        mov    %edi,%eax
        shr    $0x1f,%eax
        retq
```

```
int choice1(int x)
{
  return x / 16;
}

int choice2(int x)
{
  return 14 * x;
}

int choice3(int x)
{
  return (x << 31) & 1;
}

int choice4(int x)
{
  return (x < 0);
}

int choice5(int x)
{
  return (x + 13) /4;
}

int choice6(int x)
{
  return (x >> 31);
}
```

Fill in your answers here:

foo1 corresponds to choice 5.
foo2 corresponds to choice 1.
foo3 corresponds to choice 6.
foo4 corresponds to choice 3.
foo5 corresponds to choice 2.
foo6 corresponds to choice 4.

1.5 foo5 **1 / 1**

✓ **- 0 pts** Correct

- 1 pts Incorrect

Problem 1: 6 Points

Match each of the assembler routines on the left with the equivalent C function on the right.

```
foo1:
       lea    0xd(%rdi),%edx
        lea    0x10(%rdi),%eax
        test   %edx,%edx
        cmovns %edx,%eax
        sar    $0x2,%eax
        retq

foo2:
        lea    0xf(%rdi),%eax
        test   %edi,%edi
        cmovns %edi,%eax
        sar    $0x4,%eax
        retq

foo3:
        mov    %edi,%eax
        sar    $0x1f,%eax
        retq

foo4:
        mov    $0x0,%eax
        retq

foo5:
        imul   $0xe,%edi,%eax
        retq

foo6:
        mov    %edi,%eax
        shr    $0x1f,%eax
        retq
```

```
int choice1(int x)
{
  return x / 16;
}

int choice2(int x)
{
  return 14 * x;
}

int choice3(int x)
{
  return (x << 31) & 1;
}

int choice4(int x)
{
  return (x < 0);
}

int choice5(int x)
{
  return (x + 13) /4;
}

int choice6(int x)
{
  return (x >> 31);
}
```

Fill in your answers here:

foo1 corresponds to choice 5.
foo2 corresponds to choice 1.
foo3 corresponds to choice 6.
foo4 corresponds to choice 3.
foo5 corresponds to choice 2.
foo6 corresponds to choice 4.

1.6 foo6 1 / 1

✓ **- 0 pts** Correct

- 1 pts Incorrect

Problem 2: 9 Points

A: 3 Points

Consider the following C functions and assembly code:

```
int fun3(int a)
{
    return a * 128;
}

int fun12(int a)
{
    return a * 33;
}

int fun5(int a)
{
    return a * 65;
}
```

```
mov    %edi,%eax
shl    $0x5,%eax
add    %edi,%eax
retq
```

Which of the functions compiled into the assembly code shown?

Fun12 is the assembly code shown.

2.1 A 3 / 3

✓ **- 0 pts** Correct

- 3 pts Incorrect

B: 3 Points

Consider the following C functions and assembly code:

```
int fun3(int a, int b)
{
  if (a & b)
    return a;
  else
    return b;
}

int fun4(int a, int b)                        test   %esi,%edi
{                                             je     .L0
  if (a & b)                                  mov    %edi,%eax
    return b;                                 retq
  else                               .L0:
    return a;                                 mov    %esi,%eax
}                                             retq

int fun5(int a, int b)
{
    if (a < b)
        return b;
    else
        return a;
}
```

Which of the functions compiled into the assembly code shown?

fun4 is the assembly code shown.

2.2 B **0 / 3**

- **- 0 pts** Correct
- ✓ **- 3 pts Click here to replace this description.**

C: Points 3

Consider the following C functions and assembly code:

```
long funA(long *a, int idx, long *b)
{
   if (a[idx] > *b)
    *b = a[idx];
  else
    *b = 2 * *b;
  return *b;
}

long funB(long *a, int idx, long *b)
{
   if (b[idx] > *a)
     *a = b[idx];
  else
    *a = 2 * *a;
  return *a;
}

long funC(long *a, int idx, long *b)
{
  if (a[idx] > (long)b)
       b = (long *)a[idx];
  else
    b = (long *)(2L * (long)b);
  return (long)b;
}
```

```
        movslq %esi,%rsi
        mov    (%rdx,%rsi,8),%rdx
        mov    (%rdi),%rax
        cmp    %rax,%rdx
        jle    .L1
        mov    %rdx,(%rdi)
        jmp    .L2
.L1:
        add    %rax,%rax
        mov    %rax,(%rdi)
.L2:
        mov    (%rdi),%rax
        retq
```

Which of the functions compiled into the assembly code shown?

FunB is the assembly code shown.

2.3 C **3 / 3**

✓ **- 0 pts** Correct

- 3 pts Incorrect

Problem 3: 10 Points

Consider the following assembly representation of a function `foo` containing a `for` loop:

```
bar:
        lea     (%rdi,%rdi,1),%eax
        mov     $0x0,%edx
        jmp     .L2
.L3:
        lea     0x7(%rdx,%rax,1),%eax
        lea     0x5(%rdx),%ecx
        imul    %ecx,%eax
        add     $0x1,%edx
.L2:
        cmp     %edi,%edx
        jl      .L3
        retq
```

Fill in the blanks to provide the functionality of the loop:

```
int bar(int x)
{
    int i;
    int val = 2x;

    for( i = 0; i < x; i++ ) {

        val = val + i + 7;

        val = val * (5 + i);

    }
    return val;
}
```

3.1 int val = ________________ **2 / 2**

✓ **- 0 pts** Correct

- 2 pts Incorrect

Problem 3: 10 Points

Consider the following assembly representation of a function foo containing a for loop:

```
bar:
          lea     (%rdi,%rdi,1),%eax
          mov     $0x0,%edx
          jmp     .L2
.L3:
          lea     0x7(%rdx,%rax,1),%eax
          lea     0x5(%rdx),%ecx
          imul    %ecx,%eax
          add     $0x1,%edx
.L2:
          cmp     %edi,%edx
          jl      .L3
          retq
```

Fill in the blanks to provide the functionality of the loop:

```
int bar(int x)
{
    int i;
    int val = 2x;

    for( i = 0; i < x; i++ ) {

        val = val + i + 7;

        val = val * (5 + i);

    }
    return val;
}
```

3.2 first blank of : for (__________; _________; i++) { **2 / 2**

✓ **- 0 pts** Correct

- 2 pts Incorrect

Problem 3: 10 Points

Consider the following assembly representation of a function `foo` containing a `for` loop:

```
bar:
            lea     (%rdi,%rdi,1),%eax
            mov     $0x0,%edx
            jmp     .L2
.L3:
            lea     0x7(%rdx,%rax,1),%eax
            lea     0x5(%rdx),%ecx
            imul    %ecx,%eax
            add     $0x1,%edx
.L2:
            cmp     %edi,%edx
            jl      .L3
            retq
```

Fill in the blanks to provide the functionality of the loop:

```
int bar(int x)
{
    int i;
    int val = 2x;

    for( i = 0; i < x; i++ ) {

        val = val + i + 7;

        val = val * (5 + i);

    }
    return val;
}
```

3.3 second blank of : for (__________; _________; i++) { **2 / 2**

✓ **- 0 pts** Correct

- 2 pts Incorrect

Problem 3: 10 Points

Consider the following assembly representation of a function `foo` containing a `for` loop:

```
bar:
            lea     (%rdi,%rdi,1),%eax
            mov     $0x0,%edx
            jmp     .L2
.L3:
            lea     0x7(%rdx,%rax,1),%eax
            lea     0x5(%rdx),%ecx
            imul    %ecx,%eax
            add     $0x1,%edx
.L2:
            cmp     %edi,%edx
            jl      .L3
            retq
```

Fill in the blanks to provide the functionality of the loop:

```
int bar(int x)
{
    int i;
    int val = 2x;

    for( i = 0; i < x; i++ ) {

        val = val + i + 7;

        val = val * (5 + i);

    }
    return val;
}
```

3.4 blank for first line of for loop: ____________________; **2 / 2**

- ✓ **- 0 pts** Correct
- **- 2 pts** Incorrect

Problem 3: 10 Points

Consider the following assembly representation of a function `foo` containing a `for` loop:

```
bar:
            lea     (%rdi,%rdi,1),%eax
            mov     $0x0,%edx
            jmp     .L2
.L3:
            lea     0x7(%rdx,%rax,1),%eax
            lea     0x5(%rdx),%ecx
            imul    %ecx,%eax
            add     $0x1,%edx
.L2:
            cmp     %edi,%edx
            jl      .L3
            retq
```

Fill in the blanks to provide the functionality of the loop:

```
int bar(int x)
{
    int i;
    int val = 2x;

    for( i = 0; i < x; i++ ) {

        val = val + i + 7;

        val = val * (5 + i);

    }
    return val;
}
```

3.5 blank for second line of for loop: _________________; **2 / 2**

✓ **- 0 pts** Correct

- 2 pts Incorrect

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
 pc  = 0x0000000000400545
 rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as `??`.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the `foo` function:

0x7fffffffe00c 0x21212121

4.1 A: 0x7fffffffe000: int hex value **1 / 1**

✓ **- 0 pts** Correct

- 1 pts Incorrect. Expected "0x6c6c6548".

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
 pc  = 0x0000000000400545
 rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as `??`.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the `foo` function:

0x7fffffffe00c 0x21212121

4.2 A: 0x7fffffffe000: Description 1 / 1

✓ **- 0 pts** Correct

- 1 pts Incorrect. Expected "buf[0-3]" or something similar.

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
pc  = 0x0000000000400545
rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as ??.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the foo function:

0x7fffffffe00c 0x21212121

4.3 A: 0x7fffffffe004: int hex value 1 / 1

✓ **- 0 pts** **Correct**

- 1 pts Incorrect. Expected "0x6f57206f".

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
pc  = 0x0000000000400545
rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as ??.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the foo function:

0x7fffffffe00c 0x21212121

4.4 A: 0x7fffffffe004: Description 1 / 1

✓ **- 0 pts** Correct

- 1 pts Incorrect.

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
 pc  = 0x0000000000400545
 rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as ??.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the foo function:

0x7fffffffe00c 0x21212121

4.5 A: 0x7fffffffe008: int hex value 1 / 1

✓ - **0 pts** Correct

- **1 pts** Incorrect. Expected "0x21646c72".

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
 pc  = 0x0000000000400545
 rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as `??`.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the `foo` function:

0x7fffffffe00c 0x21212121

4.6 A: 0x7fffffffe008: Description 1 / 1

✓ **- 0 pts** Correct

- 1 pts Incorrect.

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
 pc  = 0x0000000000400545
 rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as ??.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the foo function:

0x7fffffffe00c 0x21212121

4.7 A: 0x7fffffffe00c: int hex value 1 / 1

✓ **- 0 pts** **Correct**

- 1 pts Incorrect. Expected "0x21212121".

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
 pc  = 0x0000000000400545
 rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as ??.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the foo function:

0x7fffffffe00c 0x21212121

4.8 A: 0x7fffffffe00c: Description 1 / 1

✓ - **0 pts** Correct

- **1 pts** Incorrect. Expected "i" to be mentioned.

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
 pc  = 0x0000000000400545
 rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as ??.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the foo function:

0x7fffffffe00c 0x21212121

4.9 A: 0x7fffffffe010: int hex value 1 / 1

✓ **- 0 pts** Correct

- 1 pts Incorrect

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
 pc  = 0x0000000000400545
 rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as ??.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the foo function:

0x7fffffffe00c 0x21212121

4.10 A: 0x7fffffffe010: Description 1 / 1

✓ - **0 pts** Correct

- **1 pts** Incorrect.

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
 pc  = 0x0000000000400545
 rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as ??.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the foo function:

0x7fffffffe00c 0x21212121

4.11 A: 0x7fffffffe014: int hex value 1 / 1

✓ **- 0 pts** Correct

- 1 pts Incorrect.

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
pc  = 0x0000000000400545
rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as ??.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the foo function:

0x7fffffffe00c 0x21212121

4.12 A: 0x7fffffffe014: Description 1 / 1

✓ **- 0 pts** **Correct**

- 1 pts Incorrect.

Part A

Given the code, the ascii chart on the previous page, and the following starting values, fill in the following memory diagram with execution proceeding up to 0x000000000040055e.

```
 pc  = 0x0000000000400545
 rsp = 0x00007fffffffe018
```

Memory values not updated may be left blank. Remember that an int value is 4 bytes located with the least significant byte at the address and the remaining 3 bytes in the successive byte addresses. Eg. If we know that six bytes starting at 0xbfffec10 is 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 then we would have to write down :

```
0x0000000bfffec10:   04030201
0x0000000bfffec14:   ????0605
```

Individual bytes of an int that whose value are unknown should be specifed as `??`.

Address	int hex value	Description
0x7fffffffe000	0x6c6c6548	buf[0-3]
0x7fffffffe004	0x6f57206f	
0x7fffffffe008	0x21646c72	
0x7fffffffe00c	0x21212121	i
0x7fffffffe010		
0x7fffffffe014		
0x7fffffffe018	0x00400584	low 32 bits of return address
0x7fffffffe01c	0x00000000	high 32 bits of return address

In the descriptions be sure to indicate if an address corresponds to a specific variable or argument and its value or if an address is a return address and its value.

Part B

Provide the output from the printf in the `foo` function:

0x7fffffffe00c 0x21212121

4.13 Part B:output from the printf in the foo function: 1 / 2

- **- 0 pts** Correct
- **- 2 pts** Incorrect. Expected "0xffffe00c 0x21212121"
- ✓ **- 1 pts Partially correct.**