

# CS224 Practice Midterm 2

## Problem 1. (x points):

Consider the following Y86-64 program. The object code is shown on the left, and the assembly code on the right. Your task is to determine what the program state will be after this program is run to completion.

0x0000:	30f40001000000000000		irmovq	stack,	%rsp
0x000a:	30f70300000000000000		irmovq	0x3,	%rdi
0x0014:	30f60400000000000000		irmovq	0x4,	%rsi
0x001e:	80280000000000000000		call	loopy	
0x0027:	00		halt		
0x0028:			loopy:		
0x0028:	2072		rrmovq	%rdi,	%rdx
0x002a:	6222		andq	%rdx,	%rdx
0x002c:	71750000000000000000		jle	.L5	
0x0035:	2062		rrmovq	%rsi,	%rdx
0x0037:	6222		andq	%rdx,	%rdx
0x0039:	71750000000000000000		jle	.L5	
0x0042:	30f00000000000000000		irmovq	\$0,	%rax
0x004c:	30f80100000000000000		irmovq	\$1,	%r8
0x0056:	70630000000000000000		jmp	.L3	
0x005f:			.L4:		
0x005f:	6060		addq	%rsi,	%rax
0x0061:	2027		rrmovq	%rdx,	%rdi
0x0063:			.L3:		
0x0063:	2072		rrmovq	%rdi,	%rdx
0x0065:	6182		subq	%r8,	%rdx
0x0067:	2071		rrmovq	%rdi,	%rcx
0x0069:	6211		andq	%rcx,	%rcx
0x006b:	765f0000000000000000		jg	.L4	
0x0074:	90		ret		
0x0075:			.L5:		
0x0075:	30f00000000000000000		irmovq	\$0,	%rax
0x007f:	90		ret		
0x0080:			.pos	0x100	
0x0100:			stack:		

Please fill in the following details about the program state **after the program has run to completion**.

1. Condition Codes - Indicate the value of the three condition codes:

SF		ZF		OF	
----	--	----	--	----	--

2. Registers - Indicate the value in each of the following registers:

%rax		%rdx		%rcx	
%rsi		%rdi		%rbx	
%rsp		%r8		%r10	

3. What is the value of the program counter PC?
4. What does the function `loopy` do? What are its inputs? What is its output? What is the relationship of the inputs to the output?

## Problem 2. (x points):

### Decoding Y86-64

Consider the following sequence of bytes, shown in memory starting at address 0x000. Your task is to determine the Y86-64 instruction sequence that is encoded in these bytes. Memory is displayed as in the Y86-64 online simulator, that is, memory addresses increase as we move to the right and down. The address of the first byte shown on each row is given to the left of the row. Each cell of the memory table contains a single byte.

Address	Memory							
0x000	30	f2	03	00	00	00	00	00
0x008	00	00	10	30	f1	01	00	00
0x010	00	00	00	00	00	60	20	61
0x018	12	76	15	00	00	00	00	00
0x020	00	00	00					

What is the sequence of Y86-64 instructions that is encoded in these bytes?

After this program has been run to completion, what is the value in register `%rax`?

### Problem 3. (x points):

#### Encoding Y86-64 Instructions

Consider the following sequence of Y86-64 instructions. You will be determining the byte-level encoding for these instructions, assuming that they are placed in memory starting at memory location 0x000.

```
irmovq $5, %rax
irmovq $10, %rdx
subq %rax, %rdx
andq %rdx, %rax
halt
```

Please fill in following memory table, placing the encoded byte in each memory cell. Leave any cells blank that are not used by the given instruction sequence. Memory is displayed as in the Y86-64 online simulator, that is, memory addresses increase as we move to the right and down. The address of the first byte shown on each row is given to the left of the row. Each cell of the memory table should contain a single byte (if used in the encoding), or be blank (if not used in the encoding).

Address	Memory							
0x000								
0x008								
0x010								
0x018								

#### Problem 4. (x points):

In this problem you will determine what is needed to add another instruction to the Y86-64 instruction set. Create a computation table similar to those used to implement **Project 3: Y86-64** that defines what needs to happen in each stage of the sequential architecture to implement a `cmpq rA, rB` instruction that subtracts  $R[rA]$  from  $R[rB]$  but does **not** store the result of the subtraction in  $R[rB]$ ; it only updates the condition codes.

```
rrmovq %rdi, %rcx
subq   %rsi, %rcx
jle     .L4
```

With the new instruction, the above code is simplified as below.

```
cmpq %rsi, %rdi
jle  .L4
```

The byte format for `cmpq rA, rB`, where each box is a nibble, is below.

C	1	rA	rB
---	---	----	----

Stage	cmpq rA, rB
Fetch	
Decode	
Execute	
Memory	
Write back	
PC update	

## Problem 5. (x points):

Consider the following C functions and assembly code where x is in register %rdi, y is in register %rsi, z is in register %rdx, and the caller looks in %rax for the return value:

```
long fun0(long x, long y, long z) {
    long val = 0;
    if (y != z) {
        if (x > z) {
            val = y;
        } else {
            val = x;
        }
    } else if (x < y) {
        val = z;
    }
    return val;
}
```

```
long fun1(long x, long y, long z) {
    long val = 0;
    if (y <= z) {
        if (x > z) {
            val = x;
        } else {
            val = y;
        }
    } else if (z < x) {
        val = z;
    }
    return val;
}
```

```
long fun2(long x, long y, long z) {
    long val = 0;
    if (y == z) {
        if (x > z) {
            val = x;
        } else {
            val = z;
        }
    } else if (x < y) {
        val = y;
    }
    return val;
}
```

```
rrmovq    %rsi, %r10
subq      %rdx, %r10
je        .L2
rrmovq    %rdx, %r10
subq      %rdi, %r10
rrmovq    %rsi, %rax
cmovge    %rdi, %rax
ret
.L2:
xorq      %rax, %rax
rrmovq    %rdi, %r10
subq      %rsi, %r10
cmovl     %rdx, %rax
ret
```

Which of the functions compiled into the assembly code shown?

### Problem 6. (x points):

Consider the following C functions and assembly code where `ap` is in register `%rdi` and `bp` is in register `%rsi`, and the caller looks in `%rax` for the return value:

```
int fun0(long int *ap, long int *bp)
{
    long int a = *ap;
    long int b = *bp;
    *ap = b;
    return a&b;
}
```

```
int fun1(long int *ap, long int *bp)
{
    long int b = *bp;
    *bp = *ap;
    *ap = b&b;
    return b&b;
}
```

```
mrmovq    (%rdi), %rax
mrmovq    (%rsi), %rdx
andq      %rdx, %rax
rmmovq    %rax, (%rdi)
ret
```

```
int fun2(long int *ap, long int *bp)
{
    long int a = *ap;
    long int b = *bp;
    *ap = a&b
    return a&b;
}
```

Which of the three functions is implemented in the assembly code shown?