

8

Question 1. C Puzzles (8pts)

You are running the following program on the cs33.seas.ucla.edu machine (ISA is x86-64).

```
// Create some random values
```

```
int x = random();
```

```
int y = random();
```

```
int z = random();
```

```
/* convert to other forms */
```

```
unsigned ux = (unsigned) x;
```

```
unsigned uy = (unsigned) y;
```

```
double dx = (double) x;
```

```
double dy = (double) y;
```

```
double dz = (double) z;
```

For each of the following C-puzzles, in the column marked answer, either mark true if the expression always holds (ie. always yields 1), or give a counterexample (eg. TMIN) which breaks the rule.

Expression	Answer ("True" or describe a counterexample)
$(x < y) == (-x > -y)$	(example) $x = \text{Tmin}, y = 0$
$ux - uy == -(y - x)$	(a) True
$(x \geq 0) \parallel (x < ux)$	(b) $x = -1$
$ux \& \sim(1 \ll 31) < 0$	(c) $ux = U_{\max}$
$\sim x + \sim y + 1 == \sim(x + y)$	(d) True
$dx * dy == x * y$	(e) $x = \text{Tmax}, y = 5$
$dx * y == x * dy$	(f) True
$dx + dy + dz == dz + dy + dx$	(g) True
$((x \gg 31) \ll 31) \leq x$	(h) True

4 Question 2. Multiple Choice (10pts)

For the following multiple choice questions, select all that apply.

1. Which of the following registers are guaranteed to have the same value before and after a call instruction in x86-64?

- (a) rax
- (b) rbx
- (c) rdi
- (d) rbp
- (e) rsp

2. Which of the following instructions read memory?

- (a) `movq %rbx, %rbp`
- (b) `cvtsi2ssl %rdi, %xmm0`
- (c) `leaq 4(%rax, %rbx, 2), %rcx`
- (d) `cmove %rbx, %rcx`
- (e) `subq %rax, (%rbx)`

3. Assuming our ISA is x86-64, which of the following operations could we identify as modifying the ***values on*** the program stack?

- (a) `call <func>` *Doesn't MODIFY values, but adds to it*
- (b) `addq $8, %rsp`
- (c) `movq %rax, (%rbp)`
- (d) `movq 20(%rsp), %rax`
- (e) `pushq %rbp` *Modifies stack, not values on it*
- (f) `addq %rax, 8(%rsp)`

4. What hexadecimal bit pattern would be found in memory in an x86-64 machine, for the number negative 33, when the corresponding datatype is an "int" in C?

- (a) `0x80 0x00 0x00 0x33`
- (b) `0x80 0x00 0x00 0x21`
- (c) `0x21 0x00 0x00 0x80`
- (d) `0x33 0x00 0x00 0x80`
- (e) `0xFF 0xFF 0xFF 0x21`
- (f) `0xDF 0xFF 0xFF 0xFF`
- (g) `0xFF 0x21`
- (h) `0xDF 0xFF`

5. If a, b, c are n -bit signed integers, and c is the result of $a+b$, under what conditions can we be *guaranteed* that c is not the true result under full precision arithmetic?

- (a) $a \geq 2^{n-2} \ \&\& \ b \geq 2^{n-2}$
- (b) $a \leq -2^{n-2} \ \&\& \ b \leq -2^{n-2}$
- (c) $a - b > 0$
- (d) $a + b > 0$
- (e) $a > 0 \ \&\& \ b > 0 \ \&\& \ c < 0$
- (f) $a < 0 \ \&\& \ b < 0 \ \&\& \ c > 0$

$n=4$
 2^{n-2}
 2^2

0100
1011
1100
1111
1000

6 + 1
Question 3. This Bytes (8pts)

For this question, either interpret the value as a bit pattern, or write down the corresponding value.

For floating point questions, use the following 8-bit floating point representation based on the IEEE floating point format:

- There is a sign bit in the most significant bit.
- The next 4 bits are the exponent. The exponent bias is: $2^{4-1}-1=7$
- The last 3 bits are the fraction.
- The representation encodes numbers of the form: $V = (-1)^s \times M \times 2^E$, where M is the significand and E is the biased exponent.

Bit Pattern	Value Description
(a) 00000000	Negative of smallest possible signed integer (ie. -TMin)
(b) 10000000	Largest signed integer that is a power-of-2
(c) 11111111	TMin + Tmax
(d) 01100000 ✓	Floating Point value: 32
00100001	(e) 0×21 (interpret as "char"-sized integer)
11011110	(f) $0 \times cd$ (interpret as "char"-sized integer)
00111000	(g) 1 (interpret as "8-bit float")
10111111	(h) $(-1)(1\frac{1}{2} + \frac{1}{4} + \frac{1}{8})(2^{(7-7)})$ (interpret as "8-bit float")

5

Question 4. Be the compiler! (6 pts)

Suppose we have the following C code:

```
if(a>b) { a+=b; }
```

Also assume that a and b are "int", a is in %eax, b is in %ebx. You can use other registers as temporaries.

- (a) Write an x86-64 assembly snippet that is equivalent to this statement in C, while making sure to use a jump (aka branch) instruction. Please use a label (.eg L1) as the target of the jump. (4 pts)

3

```

cmp %ebx,%eax
jg .L1
.L1:
addq %ebx,%eax

```

- (b) Write an x86-64 assembly snippet that is equivalent to this statement in C, while making sure ***NOT*** to use a jump (aka branch) instruction. (2 pts)

2

```

leaq (%ebx,%eax),%edi
cmp %ebx,%eax
cmovg %edi,%eax

```


Question 5: Interpreting Assembly (6 pts)

For each of the functions in x86-64 assembly below, convert them into a plausible version of the C code.

Assembly of function	Write a plausible C-code for the function
<pre>movq %rdi,%rax salq \$4, %rax addq %rdi,%rax addq %rax,%rax ret</pre>	<pre>long fun(long x) { long t = x; t *= 16; t += x; t += t; return t; }</pre> <p style="text-align: center;">(3)</p>
<pre>movl (%rdi),%edx addl %edx,(%rsi) movl %edx,%eax ret</pre>	<pre>int fun(int x, int y) { int w = x; y += w; return w; }</pre> <p style="text-align: center;">(0)</p>

Question 6: (6 pts)

The C code and assembly is given below for a function, but without values of M and N.

<pre>#define M __ #define N __ int array1[M][N]; int array2[N][M]; int copy(int i, int j) { array1[i][j] = array2[j][i]; }</pre>	<pre>movslq %edi,%rdi i movslq %esi,%rax j lea (%rax,%rax,4),%rdx rdx = 5j add %rdi,%rdx rdx = 5j + i mov array2(,%rdx,4),%edx edx = 4(5j + i) lea 0x0(,%rdi,8),%rsi rsi = 8i sub %rdi,%rsi rsi = 7i add %rax,%rsi rsi = 7i + j mov %edx,array1(,%rsi,4) retq array2[4*5j + 4*i] -> array1[4*7i + 4*j]</pre>
---	---

What are the values of M and N?

M = 5 ✓

(6)

N = 7 ✓

Question 7. ISA Design (4 pts)

In one or two sentences only, why have 32-bit ISAs become less popular for personal computers (laptops/desktops/cell-phones) over the last two decades?

They are less precise than 64 bit ISAs

Question 8: Stack Structures (12pts)

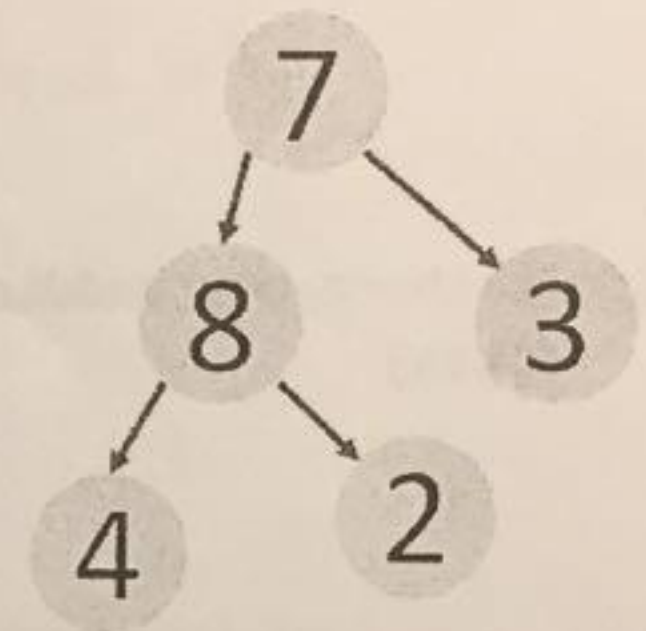
Considered an *unordered* tree represented with this struct. The function "smallest" will retrieve the smallest element from the tree.

```

struct node {
    int value;
    struct node* left, *right;
} node;

int smallest(node * n) {
    int temp, ret = n->value;
    if(n->left) {
        temp=smallest(n->left);
        if(temp<ret) ret = temp;
    }
    if(n->right) {
        temp=smallest(n->right);
        if(temp<ret) ret = temp;
    }
    return ret;
}
    
```

Assume this is the input data-structure stored using node structs:



```

smallest:
0x4004ed <+0>: push    %rbp
0x4004ee <+1>: push    %rbx
0x4004ef <+2>: sub     $0x8,%rsp
0x4004f3 <+6>: mov     %rdi,%rbp
0x4004f6 <+9>: mov     (%rdi),%ebx
0x4004f8 <+11>: mov    0x8(%rdi),%rdi
0x4004fc <+15>: test   %rdi,%rdi
0x4004ff <+18>: je     0x40050b <smallest+30>
0x400501 <+20>: callq  0x4004ed <smallest>
0x400506 <+25>: cmp    %eax,%ebx
0x400508 <+27>: cmovg %eax,%ebx
0x40050b <+30>: mov    0x10(%rbp),%rdi
0x40050f <+34>: test   %rdi,%rdi
0x400512 <+37>: je     0x40051e <smallest+49>
0x400514 <+39>: callq  0x4004ed <smallest>
0x400519 <+44>: cmp    %eax,%ebx
0x40051b <+46>: cmovg %eax,%ebx
0x40051e <+49>: mov    %ebx,%eax
0x400520 <+51>: add    $0x8,%rsp
0x400524 <+55>: pop    %rbx
0x400525 <+56>: pop    %rbp
0x400526 <+57>: retq
    
```

(note, cmov is conditional move)

Caller Return Address
old rbp
old rbx
Unused
? rbp = 8
rbx = 7
Unused
0x400519

(assume 8 bytes wide!!)

5-1

(a) Draw what is on the stack, provided in the space above, when smallest(2) is entered (ie. when it is called, and just before the instruction at 0x4004ed is executed) . Assume the root of the pictured tree is the input.

Note: If you don't know what a register value is, just mark it as "old rbp" etc. If you know what a register value is, write the corresponding value. **(6pts)**

(b) What is the size of the node struct? **(2pts)**

24 bytes

(c) Is there any padding in the struct "node" due to alignment rules? **(1pts)**

yes

(d) Can you rearrange the elements of "node" to reduce its size? **(1pts)**

No

(e) Which of the following are possible starting addresses for a node: **(2pts)**

(i) 0x7ffe4d3be87c

(ii) 0x7ffe4d3be874

(iii) 0x000444444440

(iv) 0xfffff1234568

Question 9 (Bonus): Your points overfloweth! (5pts)

Consider the following code (a variation on a hopefully-familiar example).

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int a[2];
    double d;
} struct_t;

double fun(int i, int j) {
    volatile struct_t s;
    s.d = 12345.0;
    s.a[i] = j;
    return s.d;
}
```

What input arguments to function "fun" would return the value 33.0?

$$i = 2$$

$$j = 33$$

Question 10 (bonus): (5pts)

The following is a student's submission from a previous year's question on the datalab.

```
int function(int x) {  
    int m1 = 0x11 | (0x11 << 8);  
    int mask = m1 | (m1 << 16);  
    int s = x & mask;  
    s += x>>1 & mask;  
    s += x>>2 & mask;  
    s += x>>3 & mask;  
    s = s + (s >> 16);  
    mask = 0xF | (0xF << 8);  
    s = (s & mask) + ((s >> 4) & mask);  
    return (s + (s>>8)) & 0x3F;  
}
```

What does this function do?

~~IT DON'T KNOW~~

IT WORKS