CS33: Intro Computer Organization

Name: [redacted]

UID:

IMPORTANT INSTRUCTIONS: You must write your name on the back page of the exam (And above). You may do so now. Do not open the exam.

This is an open book, open notes exam, but you cannot share books/notes. Please follow the university guidelines in reporting academic misconduct.

Note that there is an ASCII Table at the end of this exam. You will need it at some point. Do NOT detach the ASCII table.

Please wait until everyone has their exam to begin. We will let you know when to start.

Good luck!

1) GDB Lies (10, 1pt each)

Suppose we debug the following program (assignments omitted), and break at the `printf`:

```
void main(int argc, char* argv) {
  int i=...
  unsigned u=...
  float f=...
  double d=...
  printf("...",...)
}
```

List any outputs from gdb that *must* have been tampered with. (ie. if it might not have been tampered with, then don't list it) For example, the output of the first command has been tampered with, because the return value is not expected:

```
(gdb) print sizeof(double)
$0 = 37
```

```
(gdb) print sizeof(short)
$1 = 2

(gdb) print sizeof(0)
$2 = 4

(gdb) print (unsigned) -1 > 1
$3 = 0
```

-1

```
(gdb) print 0 - 1
$4 = -1
```

✓

```
(gdb) print 1U - 2
$5 = 4294967295
```

✓ $2^{32} - 1 - 2$

✓

```
(gdb) p (int)(float)i == i
$6 = 1

(gdb) p (int)(double)i == i
$7 = 0

(gdb) p (unsigned)(int) u == u
$8 = 0
```

253 -

```
(gdb) p/f 0xC2040000
$9 = 33
```

04 C2

253

27

```
(gdb) p/x (int)3.14159
$10 = 0x40490fd0
```

List GDB Outputs Tampered With: $0, $3, ~~$7, $8~~ 9 10

8

0100 0100 | 0100 1001 | 0000 | 1111 | 1101 [illegible]

0 | 000 1101 | 1111 0000 | 1001 0100 / 0000 0100

2) One of a kind! (10, 1 pt each)

A. For each instruction below, write *one* alternate instruction which performs the same operation. The alternate should not use the same instruction type (known as an opcode). It is acceptable if the flags do not match.

1. `leaq (%rbx, %rbp), %rbp` addq %rbp, %rbp

2. `leaq (, %rdi, 2), %rdi` imulq $2, %rdi

3. `mov %rax, %rax` leaq (%rax), %rax

clear out bits

4. `add $0, %eax` mov %eax, %eax

5. `xor %rbx, %rbx` movq $0, %rbx

~~6. `cltq`~~ movslq

B. Rewrite the following with *one* instruction:
(assume x is in `%rax`, y is in `%rbx`, array a (declared as `int a[256]`) is in address in `%rcx`, and that array b (declared as `char b[100][4]`) is at address 0x100.

7. `x = (x < 0) ? -1 : 0` sarq $31, %rax

8. `x = x+2*y+17` leal 17(%eax, %ebx, 2), %eax

9. `a[x]++` addl $1 (0x100, %rax, 4)

rcx

OK

10. `x = b[x][y]` movl 0x100(%ebx, %eax, 4), %eax

9

3) Bitwise Number Classification (10 pts, 2 pts each)

Match the following datalab implementations to their descriptions.

```
int func1(int x) {
    return (x>>31) & 0x1;
}

int func2(int x) {
  return (!!x) & (!(x+x));
}

int func3(int x) {
  return !x;
}

int func4(int x) {
    int nx = ~x;
    int nxnz = !!nx;
    int nxovf = !(nx+nx);
    return nxnz & nxovf;
}

int func5(int x) {
  int minus_x = ~x+1;
  return ((minus_x|x) >> 31) & 1;
}
```

1. isTmin: Returns 1 if x == Tmin, 0 otherwise func 2
2. isTmax: Returns 1 if x == Tmax, 0 otherwise func 4
3. isNegative: Returns 1 if x < 0, 0 otherwise func 1
4. isNonZero: Returns 1 if x!=0, 0 otherwise func 5
5. isZero: Returns 1 if x == 0, and 0 otherwise func 3

10

Q4) Array of hope (10 pts). Consider the following code on the left, and answer the questions on the right:

```
typedef struct {
  char g ;
  short n[10];
  int o;
  double w;
  float r;
} struct_elem;

typedef union {
  char g;
  short n[10];
  int o;
  double w;
  float r;
} union_elem;

struct_elem struct_array[10];
union_elem union_array[10];

____ get_val(int x, int y) {
  ...
}

int main(int argc, char** argv) {
  int a[16][16];
  printf("%ld\n", sizeof(struct_elem));
  printf("%ld\n", sizeof(union_array));
  union_array[0].o=0x42040000;
  printf("%f\n", union_array[0].r);
}
```

42 04 00 00

00 00 04 42

0100 000 0100 0000 1000 0100

E=0

0100 0010

1. What is printed? **(2 pts each, 6 pts total)**

48 ✓ +2

240 ✓ +2

1.00001 × 2^3 ×

2. Notice that get_val is missing a definition. The following is the disassembly from gdb:

```
Dump of assembler code for function get_val:
   0x0000066a <+0>:     movslq %esi,%rsi
   0x0000066d <+3>:     movslq %edi,%rdi
   0x00000670 <+6>:     lea    (%rdi,%rdi,2),%rax
   0x00000674 <+10>:    lea    (%rsi,%rax,8),%rdx
   0x00000678 <+14>:    lea    0x2009c1(%rip),%rax
   0x0000067f <+21>:    movzwl 0x2(%rax,%rdx,2),%eax
   0x00000684 <+26>:    retq
```

32 +2

What is the definition of get_val? **(3pts)**
(Hint: 0x2009c1(%rip) is the address of either struct_array or union_array)

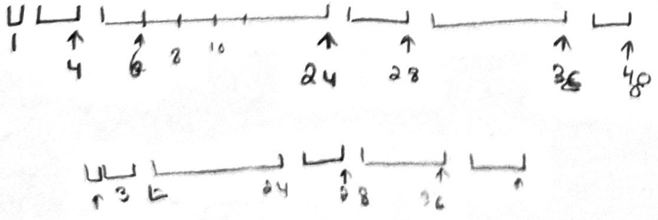
```
int  get_val(int x, int y) {
   return  (int)unsigned ( struct-aray [x]. n[y])
                                          +2.5.
}
```

3. Which of the following orders minimizes the size of the struct? **(1pts)**

a. g r o w n

b. n w o r g

[✓ c.] w r o n g +1


1
4
6
8
10
24
28
36
48
1
3
24
28
36