COMP 122/L Practice Exam #1 (With Answers)

This is representative of the kinds of topics and kind of questions you may be asked on the midterm. In addition to this practice exam, you should also review:

- The handouts we did together in class (everything through introductory MIPS assembly)
- Labs 1-3

1.) In decimal, how much is a 8 in position 5 worth?

8 * 10^5

2.) In binary, how much is a 1 in position 7 worth?

1 * 2^7

3.) In hexadecimal, how much is a E in position 4 worth?

14 * 16^4

4.) Convert decimal 19 into 8-bit unsigned binary. Show all work, including value of each digit.

0001 0011

2^4 + 2^1 + 2^0 = 16 + 2 + 1

5.) Convert unsigned binary 1101 1101 into decimal. Show all work, including value of each digit.

2^0 + 2^2 + 2^3 + 2^4 + 2^6 + 2^7 = 1 + 4 + 8 + 16 + 64 + 128 = 221

6.) Convert **two's complement** binary 1101 1101 into decimal. Show all work, including value of each digit.

Negation:
0010 0010

Add one:
0010 0011

2^0 + 2^1 + 2^5 = 1 + 2 +32 = 35

-35

7.) Consider the following binary number:

1110 0110

Is it possible to tell if this number is in unsigned or two's complement representation? If yes, explain how. If not, explain why.

It's not possible to tell. Binary numbers are not self-describing. This could be an unsigned number, a two's complement number, a character, or any number of other things.

8.) Convert decimal 2028 to 4-digit hexadecimal. Show all work, including value of each digit.

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

2028 / 16^2 = 7, remainder 236
236 / 16^1 = 14 (E), remainder 12
12 / 16^0 = 12 (C), remainder 0

0x07EC

7 * 16^2 + 14 * 16^1 + 12 * 16^0 = 1792 + 224 + 12 = 2028

9.) Convert decimal -882 to 4-digit hexadecimal. Show all work, including value of each digit.

882 / 2^9 = 1, remainder 370
370 / 2^8 = 1, remainder 114
114 / 2^7 = 0, remainder 114
114 / 2^6 = 1, remainder 50
50 / 2^5 = 1, remainder 18
18 / 2^4 = 1, remainder 2
2 / 2^3 = 0, remainder 2
2 / 2^2 = 0, remainder 2
2 / 2^1 = 1, remainder 0
0 / 2^0 = 0, remainder 0

Unsigned magnitude: 0000 0011 0111 0010

(2^9 + 2^8 + 2^6 + 2^5 + 2^4 + 2^1 = 512 + 256 + 64 + 32 + 16 + 2 = 882)

Flip bits: 1111 1100 1000 1101
Add one: 1111 1100 1000 1110

To hex: 0xFC8E

10.) What is: 1111 1101 + 0100 0101? Specify if the result has a carry-out set and if the result sets the overflow bit. Show all work.

```
1 1  1  1  1  1  0  1  0

  1  1  1  1  1  1  0  1
+ 0  1  0  0  0  1  0  1
------------------------
  0  1  0  0  0  0  1  0
```

Carry set, overflow not set

11.) What is 1111 1100 + 1000 0000? Specify if the result has a carry-out set and if the result sets the overflow bit. Show all work.

```
1  0  0  0  0  0  0  0  0

  1  1  1  1  1  1  0  0
+ 1  0  0  0  0  0  0  0
------------------------
  0  1  1  1  1  1  0  0
```

Carry set, overflow set

12.) What is 1111 1100 - 1000 0000? Specify if the result has a carry-out set and if the result sets the overflow bit. Show all work.

Flip bits: 0111 1111
Adding one achieved by an initial carry-in of 1 below:

```
1 1  1  1  1  1  1  1  1
  1  1  1  1  1  1  0  0
+ 0  1  1  1  1  1  1  1
------------------------
  0  1  1  1  1  1  0  0
```

Carry set, overflow not set

13.) What is 0x3F & 0x5A? Provide the answer in two-digit hexadecimal. Show all work.

0x3F = 0011 1111
0x5A = 0101 1010

```
  0  0  1  1  1  1  1  1
& 0  1  0  1  1  0  1  0
------------------------
  0  0  0  1  1  0  1  0
```

0001 = 0x1
1010 = 0xA

0x1A

14.) What is 0x4E | 0xB2? Provide the answer in two-digit hexadecimal. Show all work.

0x4E = 0100 1110
0xB2 = 1011 0010

```
  0  1  0  0  1  1  1  0
| 1  0  1  1  0  0  1  0
------------------------
  1  1  1  1  1  1  1  0
```

1111 = 0xF
1110 = 0xE

0xFE

15.) What is 0x7A ^ 0x14? Provide the answer in two-digit hexadecimal. Show all work.

0x7A = 0111 1010
0x14 = 0001 0100

```
   0  1  1  1  1  0  1  0
^  0  0  0  1  0  1  0  0
------------------------
   0  1  1  0  1  1  1  0
```

0110 = 0x6
1110 = 0xE

0x6E

16.) What is ~0x87? Provide the answer in two-digit hexadecimal. Show all work.

0x8 = 1000
0x7 = 0111

Flip bits: 0111 1000

17.) What is 1101 0001 << 3? Express your answer in 8-bit binary.

1000 1000

18.) What is 1100 0101 >> 2 for **logical** shift right? Express your answer in 8-bit binary.

0011 0001

19.) What is 1100 0101 >> 2 for **arithmetic** shift right? Express your answer in 8-bit binary.

1111 0001

20.) What is 0100 0101 >> 2 for **arithmetic** shift right? Express your answer in 8-bit binary.

0001 0001

21.) Specify the mask and operation you would need to isolate bit 6 of an unknown 8-bit number. The result of the operation should be 0 (0x00) if bit 6 is 0, and non-zero if bit 6 is 1. The mask should be represented in 8-bit binary.

```
  XXXX XXXX
? ???? ????
-----------
  0X00 0000  (wanted)

X & 0 = 0
X & 1 = X
X | 0 = X
X | 1 = 1

  XXXX XXXX
& 0100 0000 (mask and operation)
-----------
  0X00 0000
```

22.) Specify the mask and operation you would need to set bits 1 and 4 of an unknown 8-bit number to 1. The result of this operation results in a new number, which the unknown number will be subsequently set to. The mask should be represented in 8-bit binary.

```
  XXXX XXXX
? ???? ????
-----------
  XXX1 XX1X  (wanted)

X & 0 = 0
X & 1 = X
X | 0 = X
X | 1 = 1

  XXXX XXXX
| 0001 0010 (mask and operation)
-----------
  XXX1 XX1X
```

23.) Consider the following 32-bit binary number:

1000 0110 1111 1101 0000 0010 0000 0000

What is the value of this number, if treated as a 32-bit floating point number according to the IEEE-754 standard? Show all work.

Sign bit: 1
Exponent: 0000 1101
Mantissa: 111 1101 0000 0010 0000 0000

Sign is 1 (negative)
Exponent: 8 + 4 + 1 = 13; 13 - 127 = -114
Mantissa: 2^-1 + 2^-2 + 2^-3 + 2^-4 +2^-5 + 2^-7 + 2^-14 = 0.97662353515625

Magnitude: (1 + 0.97662353515625) * 2^-114 = 9.517096137844945e-35

Overall value: -9.517096137844945e-35

24.) Consider the following floating point number: 12.609375

Convert this to a 32-bit floating point number according to the IEEE-754 standard. Show all work, including each of the 8 steps.

Step 1: positive number, sign bit 0

Step 2: Integral portion: 12 = 1100

Step 3: Fractional part: 0.609375

0.609375 * 2 = 1.21875; 1
0.21875 * 2 = 0.4375; 0
0.4375 * 2 = 0.875; 0
0.875 * 2 = 1.75; 1
0.75 * 2 = 1.5; 1
0.5 * 2 = 1.0; 1

Step 4: normalization

1100.100111

left 3 places (exponent of 3)

Step 5: add bias

3 + 127 = 130

Step 6: bias to binary

1000 0010

Step 7: mantissa bits

1001 0011 1000 0000 0000 000

Step 8: putting it together

0 1000 0010 1001 0011 1000 0000 0000 000

0100 0001 0100 1001 1100 0000 0000 0000

25.) What values (in decimal) will be in registers $t0, $t1, and $t2 after this program executes?

```
li $t0, 15
li $t1, 5
addu $t2, $t0, $t1
```

$t0: 15, $t1: 5, $t2: 20

26.) What values (in signed decimal) will be in registers $t0 and $t1 after this program executes?

```
li $t0, 7
li $t1, 11
subu $t0, $t0, $t1
```

$t0: -4, $t1: 11

27.) What value (in decimal) will be in registers $t0, $t1, and $t2 after this program executes?

```
li $t0, 6
li $t1, 5
nor $t2, $t0, $t1
```

$t0: 6, $t1: 5, $t2: 8 (6 = (28 0s) 0110, 5 = (28 0s) 0101, (28 0s) 0110 | (28 0s) 0101 = (28 0s) 0111, ~ (28 0s) 0111 = (28 1s) 1000, (28 1s) 1000 = 4294967288). Note This question isn't fair because of how big the decimal number is; I originally neglected the 28 0s to the left.

28.) What values (in decimal) will be in registers $t0, $t1, and $t2 after this program executes?

```
li $t0, 12
li $t1, 4
multu $t0, $t1
mflo $t2
```

$t0: 12, $t1: 4, $t2: 48

29.) What values (in decimal) will be in registers $t0, $t1, and $t2 after this program executes?

```
li $t0, 14
li $t1, 4
divu $t0, $t1
mflo $t2
```

$t0: 14, $t1: 4, $t2: 3, keep in mind this is integer division

30.) What will the following program print, if run with SPIM?

```
li $a0, 83
li $v0, 1
syscall
```

83

31.) What value (in decimal) will be in register $t0 after this program executes?

```
li $t0, 3
ori $t0, $t0, 8
```

$t0: 11 (3 = 0011, 8 = 1000, OR yields 1011 = 11)

32.) What value (in decimal) will be in register $t0 after this program executes?

```
li $t0, 7
andi $t0, $t0, 13
```

$t0: 5 (7 = 0111, 13 = 1101, AND yields 0101 = 5)

33.) What value (in decimal) will be in register $t0 after this program executes?

```
li $t0, 8
xori $t0, $t0, 11
```

$t0: 3 (8 = 1000, 11 = 1011, XOR yields 0011 = 3)

34.) What does the following program print, if run with SPIM?

```
li $a0, 15
li $v0, 1
syscall
li $a0, 'a'
li $v0, 11
syscall
li $a0, 4
li $v0, 1
syscall
```

15a4

35.) What does the following program print, if run with SPIM?

```
li $a0, 24
li $v0, 1
syscall
li $a0, 47
li $v0, 1
syscall
```

2447

36.) What does the following program print, if run with SPIM?

```
.data
foo:
    .asciiz "Some string\n"
bar:
    .asciiz "Some other string\n"
main:
  la $a0, bar
  li $v0, 4
  syscall
  li $v0, 10
  syscall
```

Some other string

37.) What does the following program print, if run with SPIM?

```
.data
foo:
    .ascii "alpha"
bar:
    .asciiz "beta"
main:
  la $a0, bar
  li $v0, 4
  syscall
  li $v0, 10
  syscall
```

beta

38.) What does the following program print, if run with SPIM, and 4 is input by the user?

```
li $v0, 5
syscall
addiu $a0, $v0, 3
li $v0, 1
syscall
```

7

39.) Convert the following C-like code into MIPS assembly. The names of the variables reflect which registers must be used for the MIPS assembly. Do not assume any initial values for the registers. You may use additional registers.

```
$t0 = 3;
$t1 = 7;
$t2 = ($t0 * $t1) + 8;
```

```
li $t0, 3
li $t1, 7
mult $t0, $t1
mflo $t2
addiu $t2, $t2, 8
```

40.) Convert the following C-like code into MIPS assembly. The names of the variables reflect which registers must be used for the MIPS assembly. Do not assume any initial values for the registers. You may use additional registers.

```
int s0 = 82;
int s1 = s0 << 2;
int s2 = s1 * 20;
int s3 = s2 + 7;
int s4 = s3 - 24;
int s5 = s4 / 3;
```

```
main:
  li $s0, 82            # int s0 = 82;

  sll $s1, $s0, 2       # int s1 = s0 << 2;

  li $t0, 20            # int s2 = s1 * 20  (part 1 of 3)
  mult $s1, $t0         #                   (part 2 of 3)
  mflo $s2              #                   (part 3 of 3)

  addi $s3, $s2, 7      # int s3 = s2 + 7

  li $t1, 24            # int s4 = s3 - 24  (part 1 of 2)
  sub $s4, $s3, $t1     #                   (part 2 of 2)

  li $t2, 3             # int s5 = s4 / 3  (part 1 of 3)
  div $s4, $t2          #                  (part 2 of 3)
  mflo $s5              #                  (part 3 of 3)
```

41.) Convert the following C-like code into MIPS assembly. The names of the variables reflect which registers must be used for the MIPS assembly. Do not assume any initial values for the registers. You may use additional registers. The portions in <<>> will require you to use QtSpim functionality. You do not need to exit the program properly.

```
int s0 = <<read integer from the user>>;
int s1 = s0 + 3;
<<print integer s1>>
```

```
main:
  li $v0, 5
  syscall
  move $s0, $v0
  addi $s1, $s0, 3

print:
  li $v0, 1
  move $a0, $s1
  syscall
```