**Name:**

**Student ID:**

# Chapter 3

**3.1** [3] <§3.2> Convert $4096_{ten}$ into a 32-bit two's complement binary number.

**3.2** [3] <§3.2> Convert $-2047_{ten}$ into a 32-bit two's complement binary number.

**3.3** [5] <§3.2> Convert $-2{,}000{,}000_{ten}$ into a 32-bit two's complement binary number.

**3.4** [5] <§3.2> What decimal number does this two's complement binary number represent: $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 0000\ 0110_{two}$?

**3.5** [5] <§3.2> What decimal number does this two's complement binary number represent: $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110\ 1111_{two}$?

**3.6** [5] <§3.2> What decimal number does this two's complement binary number represent: $0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110\ 1111_{two}$?

**3.9** [10] <§3.2> If `A` is a 32-bit address, typically an instruction sequence such as

```
lui $t0, A_upper
ori $t0, $t0, A_lower
lw $s0, 0($t0)
```

can be used to load the word at `A` into a register (in this case, `$s0`). Consider the following alternative, which is more efficient:

```
lui $t0, A_upper_adjusted
lw $s0, A_lower($t0)
```

Describe how `A_upper` is adjusted to allow this simpler code to work. (Hint: `A_upper` needs to be adjusted because `A_lower` will be sign-extended.)

**3.10** [10] <§3.3> Find the shortest sequence of MIPS instructions to determine if there is a carry out from the addition of two registers, say, registers `$t3` and `$t4`. Place a 0 or 1 in register `$t2` if the carry out is 0 or 1, respectively. (Hint: It can be done in two instructions.)

**3.27** <§§3.3, 3.4, 3.5> With $x = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0101\ 1011_{two}$ and $y = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1101_{two}$ representing two's complement signed integers, perform, showing all work:

a. $x + y$

b. $x - y$

c. $x * y$

d. $x/y$

**3.28** [20] <§§3.3, 3.4, 3.5> Perform the same operations as Exercise 3.27, but with $x = 1111\ 1111\ 1111\ 1111\ 1011\ 0011\ 0101\ 0011$ and $y = 0000\ 0000\ 0000\ 0000\ 0000\ 0010\ 1101\ 0111_{two}$.

**3.30** [15] <§§3.2, 3.6> The Big Picture on page 216 mentions that bits have no inherent meaning. Given the bit pattern:

1010 1101 0001 0000 0000 0000 0000 0010

what does it represent, assuming that it is

a. a two's complement integer?

b. an unsigned integer?

c. a single precision floating-point number?

**3.35** [5] <§3.6> Add $2.85_{ten} \times 10^3$ to $9.84_{ten} \times 10^4$, assuming that you have only three significant digits, first with guard and round digits and then without them.

**3.36 (111011.01)B= ( ) D**

**(352)D=( )H**

**(72)O=( )B**

**(01011011)=( )H**