CSc 252: Computer Organization  
Fall 18 (Lewis)

Test 1  
Thu 6 Sep 2018

Name: ___________________________________________  NetID: _______________________

Person to your left: ___________________________  Person to your right: _______________________

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Answer</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Conversions and Approximations</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Binary Addition</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>MIPS Assembly</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1. (a) (4 points) What is the name of the process that we use to convert a 16-bit signed integer to the same value, but encoded as a 32-bit signed integer? 

Next, explain how we do this. What if the number is positive, does that matter? (NOTE: You are not required to explain why this works.)

(b) (4 points) Take the 2’s complement of the signed binary number 1000 1110. Show your work. (NOTE: You are not required to convert either number to decimal.)

(c) (4 points) Consider a single byte. If it is an unsigned integer, what are the maximum and minimum values that it can hold? If it is a signed integer, What are the maximum and minimum values that it can hold?
(d) (4 points) Consider a 12-bit unsigned integer. Of course, the LSB is worth 1, and the next bit is worth 2. What is the MSB worth?

If the 12-bit number is signed, what is the MSB worth?

(e) (4 points) How many registers are there in a MIPS processor, and how many bits are in each one?

2. (a) (5 points) Convert the following unsigned binary number to decimal: 0101 1101. Show your work.

(b) (5 points) Convert the following 16-bit number to both octal and hexadecimal:

1010 1100 0011 1000
(c) (5 points) Convert the decimal integer -427 to a 16-bit signed integer. Write the number in hexadecimal.

(d) (5 points) Convert each of the powers-of-2 below to an approximate value. (That is, your answer should be something like “32 thousand,” not 32,768.)

\[
\begin{align*}
2^{23} &= \underline{\quad} \\
2^{38} &= \underline{\quad} \\
2^{42} &= \underline{\quad} \\
2^{16} &= \underline{\quad} \\
2^{7} &= \underline{\quad}
\end{align*}
\]
3.  (a) (15 points) For the signed 16-bit numbers
   \[
   a = 1010\ 1110\ 1001\ 1001 \\
   b = 1110\ 1011\ 1111\ 0110
   \]
calculate \(a + b\). Do all of your work in binary; do not convert any number to decimal.
Then, state whether or not overflow occurred and explain your answer.

(b) (15 points) Now, calculate \(a - b\) for the following two numbers. You do not have to say anything about overflow.
   \[
   a = 0101\ 0000\ 0011\ 0010 \\
   b = 0110\ 0111\ 1000\ 1010
   \]
4. This question assumes some MIPS code (on the last page of this exam). The code sets up memory locations red, green, blue, black, white, dave. The code then loads the values of some of these variables into the indicated MIPS registers. In answering these questions, you can assume this code has already been executed, and that the value of some of the variables are already in the indicated registers.

**Special Limitations:**

- Each question is independent of the other questions - that is, assume that the program has started over from scratch each time.
- You may need to read from memory - but do not write to memory unless specifically instructed.

See the last page for the list of allowable instructions.

**NOTE:** You are not required to comment your code - but if you do, we may be able to offer more partial credit.

(a) (15 points) Calculate the value \((2 \times \text{red}) - \text{green} + \text{white}\), and store the result into \$s7.

(b) (15 points) First, read the variable black, add dave to it, and store it back to memory. Then do the same with white.
# values are hidden so that you can’t hardcode the answers!
.data
red: .word xxx
green: .word xxx
blue: .word xxx
black: .word xxx
white: .word xxx
dave: .word xxx

.text
main:
    # set $s1 = green
    la $s1, green
    lw $s1, 0($s1)

    # set $s2 = dave
    la $s2, dave
    lw $s2, 0($s2)

    # set $s6 = red
    la $s6, red
    lw $s6, 0($s6)

Allowable Instructions
When writing MIPS assembly, the only instructions that you are allowed to use (so far) are:

- add, addi, sub
- and, andi, or, ori, nor, nori, xor, xori
- lw, lh, lb, sw, sh, sb
- la
- syscall

(Later, we’ll add lots more.)

While MIPS has many other useful instructions (and the assembler recognizes many pseudo-instructions),
do not use them! We want you to learn the fundamentals of how assembly language works - you can use fancy tricks after this class is over.