Prerequisite Quiz
February 2, 2004
CS152 Computer Architecture and Engineering

This prerequisite quiz will be used in determining class admissions. The use of notes is not allowed during this quiz. Good Luck!

<table>
<thead>
<tr>
<th>Your Name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID Number:</td>
</tr>
<tr>
<td>Discussion Section:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>
3.141592653589793238462643383279502884197169399375105820974944
1) Assume that we have a 16 bit system that uses signed, two’s-complement integers. Perform the following conversions:

-21\text{\textsubscript{10}} to base 2:

FF2\text{\textsubscript{16}} to base 10:

65\text{\textsubscript{10}} to base 16:

(111111110110100)\text{\textsubscript{2}} to base 10:

FEC3\text{\textsubscript{16}} to base 8:

257\text{\textsubscript{10}} to base 2:
[ This page left for scratch ]
2) Suppose that we start with the following C program for fibonacci:

```c
int fib (int n) { /* n >= 0 */
    int temp;
    int sum;

    if (n <= 1) {
        return 1;
    } else {
        temp = fib (n-1);
        sum = temp + fib (n-2);
        return sum;
    }
}
```

Here is a translation of the above program into MIPS assembly language. Let’s assume that the stack pointer points at a full entry (i.e. 0($sp) is ok). Assume that we are using the RAW architecture (i.e. there are delay slots).

```
fib:  slti  $s0, $a0, 2  
      bne  $s0, $0, end

fib1: addi $sp, $sp, -8
      sw  $ra, 0($sp)
      sw  $a0, 4($sp)
      jal  fib
      addi $a0, $a0, -1
      add  $s0, $v0, $0
      lw  $a0, 4($sp)
      jal  fib
      addi $a0, $a0, -2
      add  $v0, $s0, $v0
      addi $sp, $sp, 12
end:  jr  $ra
      nop
```

There are 8 mistakes in this translation. Fix them in the above listing. To fix an error, you can either (1) add a single line of code or (2) change an existing line of code. Assume that MIPS register conventions must be maintained throughout the execution.

5
[ This page left for scratch ]
3) Please draw the waveform timing diagram for out1 and out2 below. Note that out1 and out2 start at 0. Clearly label any CLK-Q and combinational logic delays in your timing diagram!
4) Short Answer about MIPS:
   a) What is the distinction between \( x = y \) and \( \ast x = \ast y \) in C? Assume \( x \) is associated with register $s0, \ y \) with $s1. Here are 6 MIPS instructions, labeled L1 to L6:
      
      L1: add $s0, $s1, zero
      L2: add $s1, $s0, zero
      L3: lw $s0, 0($s1)
      L4: lw $t0, 0($s1)
      L5: sw $t0, 0($s0)
      L6: sw $s0, 0($s1)

      Which (if any) is true? (“L4; L5” means L4 then L5)
      A: Line 2 is \( x = y \); L4; L5 is \( \ast x = \ast y \)
      B: Line 1 is \( x = y \); L4; L5 is \( \ast x = \ast y \)
      C: Line 1 is \( x = y \); L6 is \( \ast x = \ast y \)
      D: Line 2 is \( x = y \); L3 is \( \ast x = \ast y \)
      E: L4; L5 is \( x = y \); L3 is \( \ast x = \ast y \)
      F: Line 2 is \( x = y \); L4 is \( \ast x = \ast y \)

   b) Suppose we have the following code sequence:
      
      loadvals: lb $s0, 100($zero) #byte@100= 0x0F
                  lb $s1, 200($zero) #byte@200= 0xFF

      What are the 32-bit values loaded into $s0 and $s1? Explain!
      A: 15   255
      B: 15    -1
      C: 15 -255
      D: -15   255
      E: -15    -1
      F: -15 -255

   c) Which of the codes below are pseudo-instructions in MIPS Assembly Language; that is, they are not found directly in the machine language? Explain!
      i.     addi $t0, $t1, 20000
      ii.    beq $s0, $r0, Exit
      iii.   sub $t0, $t1, 1

      A: Only i.
      B: Only ii.
      C: Only iii.
      D: Both i. and ii.
      E: Both ii. and iii.
      F: Both i. and iii.
      G: All of the above

   d) Name two reasons that the HI/LO registers were added to MIPS instead of placing multiplication/division results into the normal register file:
5) In this problem, you must design a finite state machine (FSM) for a “versatile” counter that either counts in normal binary (i.e. 0, 1, 2, 3, 0, 1, ...) or as a Grey code (0, 1, 3, 2, 0, 1, 3,...) depending on the “TYPE” signal (TYPE=1 ⇒ Grey code). Further, when “HOLD” is asserted, it will stop counting (freeze the count). When RESET is asserted, the counter will return to the zero state.

a) Complete the Following State Transition Diagram for the versatile counter. Include the “HOLD” and “TYPE” signals. Ignore RESET:

```
0 - 1
  |   |
  |   |
  |   |
 2 - 3
```
b) Construct a State Transition Table for this FSM. Encode the state as 2 bits, $S_1$ and $S_0$, where $S_1$ is the MSB (i.e. $S_1S_0=10$ for state 2). Ignore RESET.

c) Derive Next-State Logic Equations given the state transition table. Include the RESET signal in your equations. You will have 2 equations. Simplify these as much as possible (i.e. combine together terms as much as possible):

\[ S_0' = \]

\[ S_1' = \]
d) Implement the complete counter using rising-edge triggered flip flops, inverters, and 2, 3, or 4-input NAND gates. Minimize the number of gates that you use. Clearly label input and output signals. It should have as input: CLOCK, HOLD, TYPE, and RESET. It should have as output a 2-bit count value.
[ This page left for scratch ]
[ This page left for scratch ]

[ The End ]