EECS 100 *Final Exam*

Fall 2010, Engineering Part

Name: ___________________________________ unique name: _______________

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

____

Scores:

Page #	Points
2	/10
3	/10
4	/15
5	/15
6	/10
7	/10
Total	**/70**

NOTES:

- **Closed notes.** You may have a calculator and a writing implement. That's it.
- Be aware that there are two reference sheets at the end of the exam you may wish to rip out.
- **Be sure to show work and explain what you've done when asked to do so.** Getting partial credit without showing work will be rare.

For purposes of this section you are to assume '+' is "OR", '*' is "AND" and '!' is NOT.

1) Logic
   a) Fill in the following truth table for the function !(A+B)*!C. **[3]**

A	**B**	**C**	**!(A+B)*!C**
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

   b) Write a logic equation which corresponds to the following truth table. **[4]**

A	**B**	**C**	**OUT**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

   c) Draw gates which correspond to the logic equation !(A+B)*(A+C)*D. **[3]**

2) The following is code that is supposed to interface with an I/O device called BOB.

```
BOB_driver      in   200           BOB_a
                bne  BOB_driver    BOB_a      BOB_one
                in   201           BOB_c
                out  202           BOB_one
BOB_wait        in   200           BOB_a
                bne  BOB_wait      BOB_a      BOB_zero
                out  202           BOB_zero
                ret  BOB_ra
BOB_a           .data 10
BOB_b           .data 20
BOB_c           .data 30
BOB_one         .data 1
BOB_zero        .data 0
BOB_ra          .data 0
```

a. Is this an input device (receiving data from the e100 program) or an output device (sending data to the e100 program)? **[3]**

b. Which ports in the above code are associated with valid, ack, and data? **[3]**

```
valid =
ack =
data =
```

c. Depending on whether BOB is an input/output device, write a sequence of no more than three instructions to properly call BOB_driver and to send/retrieve data to/from the driver. If BOB is an input device, copy the data into a variable called "data_bob". If BOB is an output device, copy the data from "data_bob". **[4]**

3) The unit step function u(x) returns a 1 if x is zero or more and otherwise returns a 0. Write an E100 assembly function which implements this unit step function. The function is to be named “U” and it takes one argument, U_x, and returns one value, U_rv. The return address should be placed in U_ra. *<u>All variables used by this function should be declared in the function and should follow our standard naming conventions</u>*. **[15]**

4) Write an e100 assembly *program* which turns all of the green LEDs on if dipswitch 4 (DPDT_SW[4]) is a “1” and turns them off that switch is a ”0”. As long as the program is running any change in dipswitch 4 should cause the green LEDs to update appropriately. The other dipswitches could have any value and should be ignored. **[15]**

5) Using no more than 12 transistors draw a three-input OR gate in CMOS logic. **[5]**

6) Using only AND, OR and NOT gates, draw a circuit which implements a three-input XOR gate. **[5]**

7) Short answer

a) One debate in the e-waste field is if we should use lead-based solder in our electronics. Provide one good reason to use lead-based solder and one good reason not to use it. **[2]**

Reason to use it:

Reason not to use it:

b) When programming the E100, all values are kept in memory. In a real computer we instead use registers, loading and storing values from memory into and out of the registers. What are some advantages of doing this? **[3]**

8) Provide the 6-bit two-complement representation for the following values. If the value can't be represented write "no such representation" instead. **[5]**

a) -12 ____________________________

b) 55 ____________________________

c) -31 ____________________________

d) 16 ____________________________

e) -4 ____________________________

<table>
<tr><th>nstruction name</th><th>Opcode</th><th>Effect</th></tr>
<tr><td>halt</td><td>0</td><td>PC = PC+4
stop executing instructions</td></tr>
<tr><td>add</td><td>1</td><td>PC = PC+4
memory[addr0] = memory[addr1] + memory[addr2]</td></tr>
<tr><td>sub</td><td>2</td><td>PC = PC+4
memory[addr0] = memory[addr1] - memory[addr2]</td></tr>
<tr><td>mult</td><td>3</td><td>PC = PC+4
memory[addr0] = memory[addr1] * memory[addr2]</td></tr>
<tr><td>div</td><td>4</td><td>PC = PC+4
memory[addr0] = memory[addr1] / memory[addr2]</td></tr>
<tr><td>cp</td><td>5</td><td>PC = PC+4
memory[addr0] = memory[addr1]</td></tr>
<tr><td>and</td><td>6</td><td>PC = PC+4
memory[addr0] = memory[addr1] & memory[addr2]</td></tr>
<tr><td>or</td><td>7</td><td>PC = PC+4
memory[addr0] = memory[addr1] | memory[addr2]</td></tr>
<tr><td>not</td><td>8</td><td>PC = PC+4
memory[addr0] = ~memory[addr1]</td></tr>
<tr><td>sl</td><td>9</td><td>PC = PC+4
memory[addr0] = memory[addr1] << memory[addr2]</td></tr>
<tr><td>sr</td><td>10</td><td>PC = PC+4
memory[addr0] = memory[addr1] >> memory[addr2]</td></tr>
<tr><td>cpfa</td><td>11</td><td>PC = PC+4
memory[addr0] = memory[addr1 + memory[addr2]]</td></tr>
<tr><td>cpta</td><td>12</td><td>PC = PC+4
memory[addr1 + memory[addr2]] = memory[addr0]</td></tr>
<tr><td>be</td><td>13</td><td>if (memory[addr1] == memory[addr2]) {
    PC = addr0
} else {
    PC = PC+4
}</td></tr>
<tr><td>bne</td><td>14</td><td>if (memory[addr1] != memory[addr2]) {
    PC = addr0
} else {
    PC = PC+4
}</td></tr>
<tr><td>blt</td><td>15</td><td>if (memory[addr1] < memory[addr2]) {
    PC = addr0
} else {
    PC = PC+4
}

Comparisons take into account the sign of the number. E.g., 16'hffff (-1) is less than 16'h0000 (0).</td></tr>
<tr><td>call</td><td>16</td><td>memory[addr1] = PC+4
PC = addr0</td></tr>
<tr><td>ret</td><td>17</td><td>PC = memory[addr0]</td></tr>
<tr><td>in</td><td>18</td><td>PC = PC + 4
memory[addr1] = data from I/O port addr0</td></tr>
<tr><td>out</td><td>19</td><td>PC = PC + 4
I/O port addr0 = memory[addr1]</td></tr>
</table>

Port number	Port type	Definition	Use
0	in	bits 15-0: DPDT_SW[15:0]	binary input
1	out	bits 15-0: LED_RED[15:0]	binary output
2	out	bits 7-0: LED_GREEN[7:0]	binary output
3	out	bits 15-0: displayed on HEX3-HEX0	hexadecimal output
4	out	bits 15-0: displayed on HEX7-HEX4	hexadecimal output
5	in	bits 15-0: real-time clock	measure time
10	out	bit 0: lcd_valid	LCD display
11	in	bit 0: lcd_ack	
12	out	bits 3-0: lcd_x[3:0]	
13	out	bit 0: lcd_y	
14	out	bit 7-0: lcd_ascii[7:0]	
20	in	bit 0: ps2_valid	PS/2 keyboard
21	out	bit 0: ps2_ack	
22	in	bit 0: ps2_pressed	
23	in	bits 7-0: ps2_ascii[7:0]	
30	out	bit 0: sdram_valid	SDRAM memory
31	in	bit 0: sdram_ack	
32	out	bit 0: sdram_write	
33	out	bits 10-0: sdram_x[10:0]	
34	out	bits 10-0: sdram_y[10:0]	
35	out	bit 15-0: sdram_data_write[15:0]	
36	in	bit 15-0: sdram_data_read[15:0]	
40	out	bit 0: speaker_valid	speaker
41	in	bit 0: speaker_ack	
42	out	bits 15-0: speaker_sample[15:0]	
50	in	bit 0: microphone_valid	microphone
51	out	bit 0: microphone_ack	
52	in	bits 15-0: microphone_sample[15:0]	
60	out	bit 0: vga_valid	VGA monitor
61	in	bit 0: vga_ack	
62	out	bit 0: vga_write	
63	out	bits 9-0: vga_x1[9:0]	
64	out	bits 8-0: vga_y1[8:0]	
65	out	bits 9-0: vga_x2[9:0]	
66	out	bits 8-0: vga_y2[8:0]	
67	out	bit 7-0: vga_color_write[7:0]	
68	in	bit 7-0: vga_color_read[7:0]	
70	in	bit 0: mouse_valid	USB mouse
71	out	bit 0: mouse_ack	
72	in	bits 15-0: mouse_deltax	
73	in	bits 15-0: mouse_deltay	
74	in	bit 0: mouse_button1	
75	in	bit 0: mouse_button2	
76	in	bit 0: mouse_button3	
80	in	bit 0: sd_valid	SD card
81	out	bit 0: sd_ack	
82	in	bits 15-0: sd_data[15:0]	