Engineering 100 *Final Exam*

Microprocessors and Music

Fall 2007

Name: _____ unique name: _____

Sign the honor code:

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

_____

Scores:

#	Points
1	/25
2	/25
3	/25
4	/25
5	/20
Total	**/120**
Extra Credit	/10

NOTES:

- Closed book, closed notes
- There are **9** pages including this one ***The last 2 are handouts, you may rip them out!***
- Calculators are allowed, but no PDAs, Portables, Cell phones, etc. You may not store any information in your calculator relevant to this class.
- You have 60 minutes for this exam.
- **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.
- **The extra credit is very hard.** We'd suggest you don't start on it unless you have everything else done and checked over.

Color merge: 25 points

1. The VGA controller specification includes the following description:

   > The VGA controller maintains a 2-dimensional array of 8-bit values in **video memory**. The width of the array is 640 (0-639), and the height of the array is 480 (0-479). Each value represents the color of a pixel: bits 5-4 specify the amount of red; bits 3-2 specify the amount of green; and bits 1-0 specify the amount of blue (bits 7-6 have no effect on the color displayed on the screen).

   You are to write an E100 assembly function, called mergeC where you are passed three arguments mergeC_red, mergeC_green, and mergeC_blue. These will each be values between 0 and 3. Your function is to merge these three colors into one variable for use as the VGA_color_write value (port number 67). The return value should be called mergeC_rv. The call(s) to this function look like this:

   ```
   call mergeC mergeC_ra
   ```

SD card driver: 25 points

2. Say in version 2.0 of the E100 the SD card functionality were improved to allow the specification of an address (though it remains read only).

80	in	bit 0: sd_valid	SDRAM memory
81	out	bit 0: sd_ack	
82	out	bits 15-0: sd_x[15:0]	
83	out	bits 15-0: sd_y[15:0]	
84	in	bit 15-0: sd_data [15:0]	

Which SD card word is being read is specified by an (x,y) coordinate. x and y are each 16 bits. The SD card controller handles the low-level details of communicating with SD card. E100 programs interact with the SD card controller via I/O ports 80-84. These ports provide the following signals listed above.

sd_valid and sd_ack implement a protocol similar to the standard input protocol. The data being transferred from the SD card controller to the E100 program is sd_read and the location is specified a unique sd_x and sd_y.

You are to write a device driver function in the E100 assembly language for this device. The caller will pass in two arguments SDriver_X and SDriver_Y and you are to return the value located at that xy location in a variable called SD_rv. The call(s) to this function look like this:

```
call SDriver SDriver_ra
```

Digital Logic: 25 points

3. Digital logic

   a. To the left of this truth table, write a logic equation which generates the same output as this truth table. **[10 points]**

A	B	C	Out
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Out=

   b. Draw a CMOS circuit that implements the equation A&B&C using 12 or fewer transistors. **[15 points]**

Verilog: 25 points

4. Say I want a device that counts to 3 (from 0) 2 bits and after it gets to 3 it goes back to 1 (so 0,1,2,3,1,2,3,1,2,3, etc.) Fill in the following templates to complete a Verilog module called "counter" which accomplishes this task. The top module takes a clock as in input and outputs the current count. Where a line is only partly completed a blank is specifically drawn, in addition you need to fill in any other needed code.

```
module add(
   input wire [1:0] in,
   output reg [1:0] out);
   always @* begin









   end
endmodule


module register(
   input wire clock,
   input wire write,
   input wire [1:0] data_in,
   output reg [1:0] data_out);

   always _____________




endmodule
```

```
module counter(
   input wire clock,
   output _________________
);









endmodule
```

Short answer: 20 points

5. Answer the following questions

   a. For a large server farm (where 1000s of computers might be housed) the power consumption of the processors not only matters because of the electric bill and wiring for those servers, but also because of the cost to

      _____ **[4]**

   b. For a single high-performance computer (say a high-end gaming machine where cost isn't too much of an issue) the power consumption of the processor matters mainly

      because _____ **[4]**

   c. In no more than 50 words, explain how power consumption issues have led to multi-core processors (more than one processor on a single chip) becoming common. **[6]**

   d. Someone has proposed that the speaker driver should play at a 40 KHz sampling rate rather than a 8 KHz sampling rate. In no more than 50 words, explain the advantages and disadvantages of doing this. **[6]**

Notes:

1. Type in verilog code and test
2. Change transistor question to fill-in-the-table and get points right.

Extra Credit: 10 points

```
Main      call fact fact_ra
          out 3 fact_rv
          halt


fact      blt done fact_arg two

          cp push_arg fact_ra
          call push push_ra

          sub fact_arg fact_arg one
          cp push_arg fact_arg
          call push push_ra
          call fact fact_ra

          call pop pop_ra
          cp push_arg fact_rv
          call push push_ra
          sub fact_arg pop_rv two
          call fact fact_ra

          call pop pop_ra
          add fact_rv pop_rv fact_rv
          call pop pop_ra
          cp fact_ra pop_rv
          ret fact_ra

done      cp fact_rv one
          ret fact_ra
fact_ra   .data 0
fact_rv   .data 0
fact_arg  .data 5

push      cpta push_arg stack index
          add index index one
          ret push_ra
push_ra   .data 0
push_rv   .data 0
push_arg  .data 0

pop       sub index index one
          cpfa pop_rv stack index
          ret pop_ra
pop_ra    .data 0
pop_rv    .data 0
pop_arg   .data 0

one       .data 1
two       .data 2
index     .data 0
stack     .data
```

a. What's with all the calls to push and pop? Why are they needed? **[2 points]**

b. What value is displayed to the HEX display? You must explain your answer. **[4 points]**

c. The program on the side is supposed to compute the Fibonacci sequence (that's where f(n)=f(n-1)+f(n-2), and f(0)=1 and f(1)=1). It doesn't. To fix it you may change up to three lines of code, Explain why you made those changes. **[4 points]**

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]	

nstruction name	Opcode	Effect
halt	0	PC = PC+4 stop executing instructions
add	1	PC = PC+4 memory[addr0] = memory[addr1] + memory[addr2]
sub	2	PC = PC+4 memory[addr0] = memory[addr1] - memory[addr2]
mult	3	PC = PC+4 memory[addr0] = memory[addr1] * memory[addr2]
div	4	PC = PC+4 memory[addr0] = memory[addr1] / memory[addr2]
cp	5	PC = PC+4 memory[addr0] = memory[addr1]
and	6	PC = PC+4 memory[addr0] = memory[addr1] & memory[addr2]
or	7	PC = PC+4 memory[addr0] = memory[addr1] | memory[addr2]
not	8	PC = PC+4 memory[addr0] = ~memory[addr1]
sl	9	PC = PC+4 memory[addr0] = memory[addr1] << memory[addr2]
sr	10	PC = PC+4 memory[addr0] = memory[addr1] >> memory[addr2]
cpfa	11	PC = PC+4 memory[addr0] = memory[addr1 + memory[addr2]]
cpta	12	PC = PC+4 memory[addr1 + memory[addr2]] = memory[addr0]

be	13	if (memory[addr1] == memory[addr2]) {
		PC = addr0
		} else {
		PC = PC+4
		}

bne	14	if (memory[addr1] != memory[addr2]) {
		PC = addr0
		} else {
		PC = PC+4
		}

blt	15	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).

call	16	memory[addr1] = PC+4 PC = addr0
ret	17	PC = memory[addr0]
in	18	PC = PC + 4 memory[addr1] = data from I/O port addr0
out	19	PC = PC + 4 I/O port addr0 = memory[addr1]