CS 3410 Spring 2014 Homework 1

Due: Monday, Feb. 24th, 2014, 11:59pm

Name: ______________________________ NetID: _________________

Part 1: Logic, Gates, Numbers, Arithmetic

1. Consider the following truth table:

a	**b**	**c**	**d**	**out**
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

a. Write this truth table in a sum-of-products from.

b. Fill in the Karnaugh map.

cd \ ab	00	01	11	10
00				
01				
11				
10				

c. Use the Karnaugh Map to rewrite the sum-of-products form with a minimal number of terms. Shade with different colors the groupings on the Karnaugh Map.

2. Answer the following questions:

a. Fill out the following chart. Assume 16-bits and two's complement.

Decimal	Hexadecimal	Octal	Binary
-1330			
	FFD6		
		052525	
			0000 1010 1100 1010

b. The following two numbers are in 8-bit *unsigned* binary form. Subtract them and leave the answer in 2's complement signed binary form.

10001000
—10101001

c. What are the largest and smallest values that can be represented by a number in 16-bit 2's complement? How about 8-bit 2's complement? How do these values change when the number is unsigned?

	2's Complement		Unsigned	
	Smallest	Largest	Smallest	Largest
8-bit				
16-bit				

d. Consider the following additions between 4-bit 2's complement binary numbers. Indicate whether an overflow happens and explain why.

1) 0111 + 1111

2) 0011 + 0110

3) 1110 + 1011

4) 1110 + 1101

3. Do exercise B.5 in Appendix B of the text book (5th edition by Patterson and Hennessy). Draw the circuit here.

Part 2: FSMs, Memory

4. You have been asked to design a simple spam filter for emails. The filter parses the words in the email in a stream and classifies each words as one of three types: *regular*, *nonsense* or *spam*. Out of any 2 consecutive words, your filter should output a “SPAM” alert if the two consecutive words are both *nonsense* or if at least one of them were *spam*. After a SPAM alert has been displayed the filter should continue processing. If it is not outputting a “SPAM” notification, the filter should output “Okay”.

   Call the input that the filter is going to process x_0, the previous input x_1, and the previous-previous input x_2. Let *regular*=0, *nonsense*=1 and *spam*=2. The state of the filter is defined by the value of x_1 and x_2 (For example, if the two previous inputs were *nonsense* the current state is $x_2 = 1, x_1 = 1$). Use a Moore machine for this problem.

a. What is the number of *distinct* states? Distinct means that no two states have the same transitions and output.

b. Draw the finite state machine. Place states which output “Okay” on the left and states which output “SPAM” on the right. Draw labeled directed transition arrows. Indicate the initial state.

5. Answer the following questions:

a. Register files are used for many processor operations because they have very few gate delays. Why aren't registers used for ALL memory then?

b. What allows DRAM to be so dense (in terms of the amount of data it can store in a specific amount of space)? Explain why this is both good and bad.

c. What does a Tri-State buffer allow us to do with things like D-latches?

d. What "kind" of memory are SRAM and DRAM typically used for?

e. Why is a clock needed when memory is involved?

Part 3: CPU, CPU performance & pipeline

6. While more pipeline stages generally mean higher throughput, give two reasons why having many additional pipeline stages will not always improve the speed of the processor.
(**Note: this question will NOT be graded.**)

7. A processor has a clock rate of 400MHz and one program is executing on the computer. The instruction types, instruction number for each type and the cycles for each instruction type of the program are shown below.

Instruction Type	Instruction Count	Cycles
Integers	45000	1
Load/Store	75000	2
Float	8000	4
Branch	1500	2

a. Calculate the CPI of the program when executed on this processor.

b. How much time will it take to execute the program on the processor?

8. Do exercise 4.3 in session 4 of the text book (5th edition by Patterson and Hennessy).

Part 4: MIPS, MIPS Pipeline

9. Write the MIPS equivalent to the following C code which takes a constant integer and adds it to another from an array. You must include for each variable the register in which it is contained. Do not include or use delay slots.
   a = b + myArray[42]

   a -- __________, b -- __________, myArray (base address) -- $s4

10. Write the MIPS equivalent to the following C code which is a simple loop. Again, include the registers in which your variables are stored. Do not include or use delay slots.
    int i = 0;
    int n = 16;
    while(i < n){
    i++;
    }
    i -- __________, n -- __________

11. The selection sort algorithm is one of the simplest sorting algorithms. For every index in the list, it simply iterate over every item after it to find the smallest, swapping them.

a. Write this algorithm (for ascending order) in C. You may assume that array nums is a valid pointer and has exactly n elements. **Addendum: This is now provided for you.**

```
sort(int* nums, int n)
    int minIndex, minValue, i, j, temp;
    for (i=0; i<n; i++){
        minIndex = i;
        minValue = nums [minIndex];
        for (j=i+1; j<n; j++){
                if (nums [j]<minValue){
                        minIndex = j;
                        minValue = nums[j];
                }
        }
        temp = nums [i];
        nums [i] = minValue;
        nums [minIndex] = temp;
    }
```

b. Now convert the **above** code to MIPS assembly. Assume that the based address of array nums is stored in register $a0 (it is passed in) and n in $a1. **Also include registers corresponding to your variables as well as comments for where an assembly instruction corresponds to your C code.** You do not need to include instructions for saving registers and allocating new frames. Do not include or use delay slots. **Addendum: The skeleton is now provided for you. Fill in the blanks.**

```
$t0: array offset temp
$t1: i
$t2: minIndex
$t3: minValue
$t4: j
$t9: branching test temp
…Saving registers, allocating new frame omitted…

1)      ____________________      // i=0

LP1:

2)      ____________________      // check i<n

3)      ____________________      //Outer for loop test statement
ADDI $t2, $t1, 0                  //minIndex = i
SLL $t0, $t2, 2                   //minIndex*4
ADD $t0, $a0, $t0                 //actual address of minIndex
LW $t3, 0($t0)                    //minValue = nums[minIndex]

4)      ____________________      //j=i+1
```

```
LP2:
5)    ____________________      //j<n

6)    ____________________      //Inner loop test
SLL $t0, $t4, 2
ADD $t0, $a0, $t0
LW $t5, 0($t0)                  //nums[j]
SLT $t9, $t5, $t3               //nums[j]<minValue
BEQ $t9, $zero, INNEREXIT
ADDI $t2, $t4, 0                //minIndex = j
ADDI $t3, $t5, 0                // minValue = nums[j]

INNEREXIT:
ADDI $t4, $t4, 1                //j++
J LP2

LP2EXIT:
SLL $t0, t1, 2
ADD $t0, $t0, $a0

7)    ____________________      //temp = nums[i]

8)    ____________________      //nums[i] = minValue
SLL $t0, $t2, 2
ADD $t0, $t0, $a0
SW $t5, 0$(t0)                  // nums [minIndex] = temp
ADDI $t1, $t1, 1                //i++

9)    ____________________      //exit loop

EXIT:
..restore registers, return…
```

12. We have two MIPS processors. One is a five stage pipelined processor and the other is an unpipelined multi-cycle processor.

    For the unpipelined processor:

    Branches and Jumps take 100 ns
    Math and Logical Operations take 125 ns
    Memory Ops: 175 ns
    For the pipelined processor:

    Critical Path for Fetch 125 ns
    Critical Path for Decode 110 ns
    Critical Path for Execute 115 ns
    Critical Path for Memory 125 ns
    Critical Path for Write Back 75 ns
    Our program contains 25% branches, %50 Math/Logical, and %25 Memory Ops.

    Which processor would we choose strictly based on performance?

Part 5: C-Programming

NOTE: On all problem sets in CS3410, submit code which adheres to the C99 standard (for gcc, compile with -std=c99).

The instruction following requires that you are in the CSUG computers. Use your favorite SSH client to connect to csugXX.csuglab.cornell.edu, where XX is a number between 01 and 08. Login with your NetID and password. Alternatively, you may want to use the CSUG virtual machine. Check out the course webpage for the installation direction.

If you have a problem or question, the first place you should look is C: A Reference Manual. Many common questions are also the top result on your favorite search engine.

Overview

For this assignment, we will start with the proverbial first program—Hello World. It simply prints out "Hello World". Then you will be asked to implement and submit a simple interactive program.

Writing the C Program

Input the source code (which should be very short) using your favorite text editor and save as "hello.c". Many people like to use `vim` or `emacs`. You may wish to start with `nano`, which provides a more friendly interface.

Here is one way to do this using the `nano` text editor. On the console, type:

```
$ nano hello.c
```

After the editor comes up in the terminal window, type in the "Hello World" program.

Then, save and exit out of editor by pressing [ctrl]+[x]. It asks whether you want to save your modification. Type [y] and confirm the filename by pressing [Enter].

When you see the terminal prompt again, verify that you have the "hello.c" file in your current directory by typing:

```
$ ls

hello.c
```

We will now proceed to compile this source file.

Compiling the C File

Still remaining in the same directory as your "hello.c" file, compile "hello.c" by typing the following into the terminal on your console:

```
$ gcc -Wall -std=c99 hello.c -o hello
```

This command compiles your code into an executable named "hello". Now run your program with

```
$ ./hello
```

You should now see the fruits of your labor!

What to Submit

Write a C program that asks for your name, then prints a personalized greeting (your choice, but it must include both the input AND your NetID somehow). For example, if your NetID is "cs321", for the input "Paul" on the command line:

```
What is your name? Paul

Hello, Paul. I am cs321. Would you like to play a game?
```

Submit both your source code and your compiled binary to CMS.

Part 6 (study questions): Data and control hazards

Note: this part will NOT be graded as it covers lectures after the due date. But you are recommended to study it for a warm-up before the prelim.

13. Consider the following MIPS assembly sequence:

```
sub $3, $7, $8
add $6, $4, $8
add $4, $4, $7
and $12, $3, $5
sub $2, $4, $4
sw $13, 12($2)
```

a. Identify the data hazards in the sequence by circling them and drawing an arrow to the dependenc(ies). For example:

```
add $3, $4, $5
add $6, $3, $7
```

b. Notice that the above data hazards are all RAW hazards.
   1) Describe the other two less-common types of data hazards.

   2) Would you expect to see these other two types of data hazards (as mentioned in part a) in the generic 5-stage pipeline for MIPS? Why or why not?

14. Fill in the following pipeline execution chart

a. Assuming a *non-bypassed* 5-stage pipeline. (Hint: data hazards are resolved by stalling).

	Cycle																	
Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
sub $3, $7, $8																		
add $6, $4, $8																		
add $4, $4, $7																		
and $12, $3, $5																		
sub $2, $4, $4																		
sw $13, 12($2)																		

b. What is the CPI (cycles per instruction) for the case above?

c. Assuming a *fully-bypassed* 5-stage pipeline. (Hint: data hazards are resolved by forwarding).

	Cycle																	
Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
sub $3, $7, $8																		
add $6, $4, $8																		
add $4, $4, $7																		
and $12, $3, $5																		
sub $2, $4, $4																		
sw $13, 12($2)																		

d. What is the CPI (cycles per instruction) for the case above?

15. Do exercise 4.14.1 and 4.14.2 in session 4 of the text book (5th edition by Patterson and Hennessy).