CS 3410 Spring 2015 Homework 1

Due: Monday, Feb. 23rd, 2015, 11:59 pm

Name: _____ NetID: _____

Part 1: Logic, Gates, Numbers, Arithmetic

1.

Consider the following truth table:

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

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

b. Fill in the Karnaugh map.

ab / cd	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 the groupings with different colors on the Karnaugh Map.

d. Draw the circuit that has the output of the sum-of-products form. Use a minimal number of elements.

2.
Answer the following questions:

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

Decimal	Hexadecimal	Octal	Binary
3410			
	FF7F		
		017037	
			1111 1011 1011 0100

b. What are the largest and smallest values that can be represented by a number in 64-bit 2′s complement? How about n-bit 2′s complement when n > 1? How do these values change when the number is unsigned? For each, mention how many digits are to the left of the decimal point.

c. What is the meaning of an arithmetic overflow? Give an example of a calculation that results in an arithmetic overflow and an example of a calculation that does not result in an arithmetic overflow. Use signed numbers. Explain why.

3. Do exercise B.11 from Appendix B of the textbook (**5th edition by Patterson and Hennessy**).

Part 2: FSMs, Memory

4.

a. Draw the FSM (a Moore machine with the fewest number of states) that accepts all strings over the alphabet {a,b} such that the number of a's mod 3 equals the number of b's mod 3.

b. Construct a Moore machine with the fewest number of states that recognizes all binary strings that end in "1101".

c. Pick a minimal binary encoding for the states of the machine in part b, and write down a truth table representing the output and next state functions.

d. Write out the next state and output function for the above.

e. Draw the logic circuit for the function above.

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. How can we use tri-state buffers to create scalable memory with D-latches?

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

e. Why do memories need some sort of 'enable' input (the role filled by the clock in some memories)?

Part 3: CPU, CPU performance & pipeline

6. Give two reasons why, at some point, having more stages in a pipelined processor can be disadvantageous.

7. A processor has a clock rate of 500MHz 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	30000	1
Load/Store	65000	2
Float	12000	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?

c. Now assume that we can reduce the number of cycles for Float instructions from 4 to 2. What is the speedup?

8. Do exercise 4.4, parts 1 to 3, from section 4 of the textbook (**5th edition by Patterson and Hennessy**). Show your work.

Part 4: MIPS, MIPS Pipeline

9. Do exercise 2.1 in section 2 of the text book.
(Use only the instructions in Figure 2.44 or 2.45, on pages 162-163)

10. Do exercise 2.3 in section 2 of the text book.

11. In this question you will be translating an algorithm to find the maximum element in an array, from C to MIPS assembly. We assume that the array `nums` is valid, has positive elements, and has `n` elements where `n` > 0. Here is the algorithm in C:

```
int maximum(int* nums, int n) {
     int maxval = 0;
     for (int i = 0; i < n; i++) {
          if (maxval < nums[i]) {
               maxval = nums[i];
          }
     }
     return maxval;
```

A skeleton of the MIPS code is provided for you. Assume that the base address of array `nums` is stored in register `$a0` (it is passed in) and the length `n` is stored in register `$a1`. Below you can see what the other registers correspond to. Do not worry about any hazards. For example, do not include or use delay slots.

```
$t0: array offset temp
$t1: maxval
$t2: i
$t9: branching test temp
$a0: base address of nums
$a1: n


...saving registers and allocating new frame omitted...

     ADDI $t1, $zero, 0        // maxval = 0
     ADDI $t2, $zero, 0        // i = 0

LOOP:
_________________              // check i < n
_________________              // loop test statement

_________________              // nums array offset calculation
_________________              // add base address to offset
_________________              // load nums[i]

_________________              // check maxval < nums[i]
_________________              // otherwise, branch to INNEREXIT

_________________              // set maxval to nums[i]

INNEREXIT:
     ADDI $t2, $t2, 1          // i++
     J LOOP

EXIT:
     ADDI $v0, $t1, 0 // put maxval in return register
     ...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 80 ns
Math and Logical Operations take 110 ns
Memory Ops: 165 ns

For the pipelined processor:

Critical Path for Fetch 115 ns
Critical Path for Decode 100 ns
Critical Path for Execute 105 ns
Critical Path for Memory 115 ns
Critical Path for Write Back 90 ns

Our program contains 20% branches, 45% Math/Logical, and 35% Memory Ops.

Which processor would we choose strictly based on performance? Show your work.

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).

You may use your VM to do the following problem. Otherwise, you will be required to use 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.

Write a C program that inputs a string from the command line and outputs whether or not the string is a palindrome. A **palindrome** is a word, phrase, number, or other sequence of characters which reads the same backward or forward. You can assume that the input does not contain any white spaces. **Submit the C file and the executable on CMS.** Your program **must** compile and run on the course provided VM and /or the CSUG computers.

To compile the C program you should use the following command, where `palindrome.c` is the C source file and `testpalindrome` is the executable file.

```
gcc –Wall –std=c99  palindrome.c  –o testpalindrome
```

You can run the program with the command:

```
./testpalindrome
```

You should now see the fruits of your labor!

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.

Ex:

Input a String: racecar
racecar is a palindrome

Input a String: Racecar
Racecar is not a palindrome

Input a String: a
a is a palindrome

Input a String: whales
whales is not a palindrome

You may assume the input string is less than 100 characters long. You may not use any functions from the string.h library.

A skeleton has been provided. Do NOT change any of the print statements. You can change the variable names.

```
#include <stdio.h>

int main()
{
    char str[100];
    int i, length=0, flag=0, start, end;

    printf("Input a string: ");


    // Read in input from the command line

    // Find the length of the string.
    // Hint: How do you know when a string ends in C?

    // Check if str is a palindrome.


    if(flag==1)
        printf("%s is not a palindrome.\n", str);
    else
        printf("%s is a palindrome.\n", str);

    return 0;
}
```

Part 6 (study questions -- OPTIONAL): 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
```

You are not required to know the information presented in section b. It is FYI only.

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 these other two types of data hazards (as mentioned in part a) to be a problem 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 from section 4 of the textbook (**5th edition by Patterson and Hennessy**).