CS 352 – Prof. Mark **Assignment #2** **Due Feb 17, 2004**

Problems #1-6:

P&H 3.2 (5 points)
P&H 3.3 (10 points)
P&H 3.5 (5 points)
P&H 3.16 (15 points)
P&H 3.17 (15 points)
P&H B.2 (5 points)

Problem #7: Variable length instructions (30 points)

Read about the x86 instruction set in section 3.12 of the textbook and answer the following questions:

a) Based on the 6 opcode formats (Fig. 3.34) shown in the chapter, if we wanted to fetch four instructions per cycle, what is the minimum number of bytes that might need to be fetched (you may assume a certain combination of instructions), and what is the maximum?
b) How many different possible combinations of instruction layouts (where the opcodes might appear) are there?

Ungraded: Please think about, and write short paragraphs discussing the following two questions. I'll cover these in class after people have considered them.
1. What are the challenges for quick decoding of any four of these instructions in a given cycle?
2. Suggest how to build the fastest possible decoder for four of these instructions.

Use Pair Programming for Problems #8-10 Below

For the following programming problems, you will complete the problems using the "pair programming" approach. With this approach, you work on the assignment side-by-side with your partner. You may choose your partners, but stick with one partner for all three problems below. One person is the "driver" (i.e. he/she types) while the other person watches and thinks. Switch roles from time to time during the assignment.

The names of both partners must be in a comment at the start of the program. Only one partner should run the "turnin" program for the assignment.

All of your code turned in should be be adequately commented, and should obey the guidelines set forth on the course *homework* page. Points will be deducted if the submissions do not follow the guidelines, so read them carefully. Also, please note that we will not grade illegible assignments.

Problem #8: Assembly language programming #1 (50 points)

The following two programs will require you to use the `XSPIM` simulator to write, debug, test, and run MIPS assembly programs. The XSPIM simulator runs on the departmental Linux machines. Professor Keckler has written an excellent tutorial on XSPIM which I have made available on our class's web page under "handouts".

Write a MIPS assembly program that contains the string "`A longhorn cow says MOO`" in simulated memory, and prints it out to the screen in reverse. You may assume that the previous location in memory (before the start of the string) contains a zero. Turn in the file `moo.asm`:

```
csh> turnin --submit harsha hw2 moo.asm
```

Problem #9: Assembly language programming #2 (60 points)

Write a function `choose(x,y)` where the function you implement is:

$$choose(x,y) = \frac{x!}{y!(x-y)!}$$

Remember that ! is the factorial operator, so 4! = 4 x 3 x 2 x 1 = 24. Also remember that 0! = 1. The program should have `x` and `y` be entered by the user as integers, and then output the result.
Please write the output for the following quantities. If anomalies occur, explain why you get the answers you do for each:

```
a) choose(10,2)
b) choose(100,50)
c) choose(1,0)
```

Turn in the file `factorial.asm`:

csh> turnin --submit harsha hw2 factorial.asm

Problem #10: MIPS Opcode decoding (65 points)

Write a program in Java to open a file of binary numbers, read the 32-bit numbers one-by-one, and print out the MIPS opcode that each 32-bit number represents. You are responsible for all real MIPS instructions, but not pseudo instructions as defined in Appendix A. For example, an instruction with the number 0x00430820 would correspond to the "add" instruction on the back inside cover of the textbook, and should be printed:
ADD
Each output result should be capitalized, no space/tab before and after the instruction name, and separated by a new-line. Turn in a file mips_decode.java that can be run by typing:

csh> javac mips_decode.java
csh> java mips_decode <input file>

Turn in your program using the following command.:

csh> turnin --submit harsha hw2 mips_decode.java