CSE-362 Homework 2
September 2006

- Read: Heuring & Jordan (H & J), Chapter 2.0, 3.1, 3.2, 4.1.
- Review: Appendix C (Logic Design Review)
- VHDL reading (if needed): Yalamancheli, Sections 3.4, 4.1-4.6.

DUE: Wednesday, Sept. 27, 2006

1.0 Queuing Problem
Say that measurements have shown that, for a given computer, instructions have an average CPI (Clocks Per Instruction) of 1.5 cycles/instruction. Say that the number of cycles for instructions has been measured to be exponentially distributed. The clock frequency is given as 2.5GHz.

1.1 A memory fetch unit provides instructions to the processor for execution at an average rate \( \lambda \). Say that the time between instruction arrivals (inter-arrival time) is modeled as being exponentially distributed (i.e., Poisson arrival process). What should the average arrival rate, \( \lambda \), be set to if we would like the processor to be utilized 75% of the time.

1.2 At 75% processor utilization what is the average delay that an instruction encounters after it has been fetched from memory?

1.3 Say that we model the system as an M/M/1 queuing system (with infinite FIFO queue). The queue in the system will hold instructions waiting to be executed. In a real system this queue is finite in length. Say that to ensure a low probability of the queue getting filled, its size is set to 4 times the average queue length. How large should the instruction queue be?

1.4 It has been suggested that system performance can be improved by replacing the single processor above with two processors each having a clock rate of 2.0GHz. Parallelism is exploited by grouping the instructions into pairs that can be executed independently. That is, the pair of instructions are be sent to the two processors for execution simultaneously. However, this pairing only can be done 50% of the time. The remaining 50% of instructions utilize only a single processor. Compare the performance of this system with the single processor system above. For simplicity, neglect the effects of variance in the instruction execution times. Show and explain your work.

2.0 Assembly Language Programming: In this problem you are to write a simple SRC assembly language program and use the downloaded assembler and simulator tools to assemble and execute the program.

2.1 Download the assembler from the web site that is given in the text or follow the instructions provided in class (and on the class website).
2.2 Using a simple text editor (not word), write an SRC program to sort through an array of 128 positive 16-bit integers, ordering the integers from smallest to longest. Have the program being at location 1000D and the array to be sorted at 4000D. You can use any sorting algorithm you please.

One simple sorting algorithm, for example, considers the array as being divided into integer pairs. Beginning at the start of the array (4000D), the first pair of integers is compared and then, if necessary, the two integers are swapped so that the smaller one is in the lower pair address. The next pair of integers in the array is then considered and the same operation is performed. This is continued on successive pairs until the entire array has been scanned. The process is now repeated, however, the pairs now start at location 4002D (the odd numbered integer pairs). After doing this, the process is repeated starting again at 4000D. After continuing this for N iterations the array is sorted. Here is an example:

```
Array  6  3   2  12   25  7   4  5
[6  3] [2  12] [25  7] [4  5]
Compare and Swap [3  6] [2  12] [7  25] [4  5]
  3 [6  2][12  7][25  4] 5
  3 [2  6][7  12][4  25] 5
[3  2] [6  7] [12  4] [25  5]
[2  3] [6  7] [4  12][5  25]
  2 [3  6] [7  4][12  5] 25
  2 [3  6] [4  7][5  12] 25
[2  3] [6  4] [7  5] [12  25]
[2  3] [4  6] [5  7] [12  25]
  2 [3  4] [6  5][7  12] 25
Sorted Array     2  3    4   5   6  7  12  25
```

2.3 Load the program into the system, set a breakpoint at the beginning of the program and then 1-step the program. Follow the various register contents to ensure that it is working properly. Run the program to completion and determine whether the final sort is correct. Use an array of random integers.

2.4 If the above algorithm is used, determine how many instructions the program executes during one scan and swap of the array, and also for the entire program. If a different algorithm is used, perform the equivalent analysis. If the CPI for the machine on which the program will execute equals 1.2 & its clock frequency is 2.5GHz, determine the time it takes to execute the program.

2.5 Determine the number of memory reads and memory writes associated with the program. For the memory reads, also break
up the number into its two components, instruction reads and data reads.

Hand in:
- A copy of your program (with comments).
- An annotated screen printout showing the screen when the program is completed.
- Answers to the questions above including a discussion of problems that were encountered in the process.
- Email, as an attachment, the program to: Eric Tyson (ejt1@cec.wustl.edu) and Saurabh Gayen (sg3@wustl.edu).

3.0 Multiplexers VHDL Implementation and Testing
Consider a 4-to-1 and 16-to-1 multiplexers designed in HW-1.

3.1 Complete 2.5 and 2.6 of HW-1. Write a VHDL (structural) program for the 4-to-1 multiplexer using 2-input gate components (you can use VHDL AND, OR, etc. primitives). Write a VHDL (structural) program for the 16-to-1 multiplexer using the 4-to-1 multiplexer of 2.4 as a component.

3.2 Develop a set of tests that ensures that the two multiplexers operate properly. Describe the tests and indicate how they prove correct operation. Run the tests using ModelSim and hand in your commented VHDL and annotated samples of your output.

3.3 Say you are to design a small embedded processor that has 16, 16-bit general purpose registers (GPR). In the processor’s STORE instruction, the contents of a selected GPR is first moved into a special register; the Data Register (DR). Draw a block diagram indicating how one could implement this move portion of the instruction using a group of 16-to-1 MUXes.

4.0 Heuring & Jordan: Problem 2.21

5.0 Addressing Modes
Write SRC instructions to load a value into a register using the selected addressing modes found in Table 2.8: a) Register Indirect; b) Indirect; c) Relative; d) Auto-increment. Use multiple SRC instructions for a mode only when necessary.

6.0 CPI and MIPS
A program has the following instruction mix: 25% load/store (execution time 2ns each); 55% ALU instructions (execution time 1.5ns each); and 20% branch instructions (execution time 1.75ns each).

6.1 If the clock period is 350ps, calculate the average CPI for the program.
6.2 Determine the average MIPS rate of the program.