EE 7700-4 Homework 1 Due: 4 September 1998

Problem 0 below has been pre-solved, the solution and other files needed can be found in /data4/koppel/tca/hw01 on the ECE Solaris systems. The techniques used in its solution can be applied to the other problem.

Problem 0: Using Shade, write a program to determine the ten most frequently executed consecutive instruction pairs, omitting any pair that contains a `save` or `restore`. (These allow the processor to save and restore its own registers, a special feature of the SPARC architectures.) The output might be:

```
[sol] % echo /opt/local/bin/X11/netscape  igroup
 Instr 1  Instr 2       Count  Percent
 lduw     subcc      14471267   2.997%
 subcc    be         13783327   2.854%
 sethi    lduw       10366120   2.147%
 lduw     add        10363630   2.146%
 add      add        10269228   2.127%
 lduw     lduw       10252944   2.123%
 subcc    bne        10109655   2.094%
 subcc    bne,a       7845333   1.625%
 add      lduw        7550369   1.564%
 or       jmpl        6790452   1.406%
```

This would indicate that in a run of netscape (Communicator) 2.997% of the instructions executed consisted of a lduw (load unsigned word) followed by subcc (subtract and set condition code).

Problem 1: One technique to improving computer performance is merging two instructions into one so that the combined unit does the same work in half the time. (Though it would be quite naïve to assume that such benefit could really be achieved by combining any two instructions, it does make a good first simulation assignment. Hardware may have to be duplicated, the cost of that hardware might be more effectively "spent" elsewhere. Another major problem is coding the operands in the combined instruction.)

Choose two SPARC instructions to combine, and based on simulations, determine the performance improvement. The pair cannot include two loads, two stores, or a load and a store. The combined instructions have a CPI of one but the two individual instructions, when appearing alone, have a CPI of 1.2. All other instructions have a CPI of one.

Problem 2: Choose a classmate to be your adversary[1]. Have your adversary provide you with his or her analyzer executable (not the source, so you can't find out which instructions were combined). Run a set of programs of your choosing on your adversary's analyzer; your goal is to obtain the highest credible execution time possible on your adversary's system. List the programs used, the input data, and the execution time of each run. Justify the credibility of your programs but include a rebuttal from your adversary. (For example, justification, "ran widely used utility program;" rebuttal, "xeyes? If that's a utility the productivity paradox is solved!")

[1] Keep it friendly.