High Performance Computer Architecture (CS60003)

Tutorial 1

Indian Institute of Technology Kharagpur

Exercises

1. A benchmark suite B_1 consists of equal proportion of class-X and class-Y instructions. Machine M_1 and M_2, with identical 1 GHz clock, have equal performance of 100 MIPS (Million Instructions Per Second) on B_1. If we replace half of class-X instructions in B_1 with class-Y instructions to derive another benchmark suite B_2, M_1's running time becomes 80% that of M_2. If we replace half of class-Y instructions with class-X instructions to transform B_1 to B_3, M_2 becomes 1.4 times as fast as M_1. What can you say about the average CPIs of M_1 and M_2 for the two instruction classes?

2. You have a system that contains a special processor for doing floating-point operations. You have determined that 70% of your computation time is utilized by the floating-point processor. When a program uses the floating-point processor, the speedup of the floating-point processor is 30% faster than when it does not use it.

   (a) Calculate the overall speedup by using the floating-point processor.

   (b) In order to improve the speedup consider two options:

   - **Option 1:** Modify the compiler so that 80% of the computation time is utilized by floating-point processor. Cost of this option is \$50K.
   - **Option 2:** Modify the floating-point processor. The speedup of the floating-point processor is 100% faster than when it does not use it. Assume in this case that 60% of the computation time is utilized by the floating-point processor. Cost of this option is \$60K.

   Which option would you recommend? Justify your answer quantitatively.

3. The code below shows an example that we would like to rename by the hardware.

```
Loop:    LD     F2,0(Rx)
         MULD   F5,F0,F2
         DIVD   F8,F0,F2
         LD     F4,0(Ry)
         ADDD   F6,F0,F4
         ADDD   F10,F8,F2
         SD     F4,0(Ry)
```

   Unroll the loop twice and perform register renaming such that all the false dependencies are resolved and all the true dependencies are maintained properly. Assume that the hardware has a pool of temporary registers (call them T registers, and assume there are 64 of them, T0 through T63).

   Your answer should start in the following way.

```
LD     T9,0(Rx)
MULD  T10,F0,T9
```

$$\vdots$$

4. Consider the following C code which multiplies a scalar to a vector:

```
for (int i = 0; i < 100; i++)
    x[i] = x[i] * s;
```

   The assembly code for the above C program snippet is given as follows:

```
Loop:  LD      F0   0(R1)
       MULTD   F4   F0   F2
       SD      F4   0(R1)
       SUBI    R1   R1   #8
       BNEZ    R1   Loop
```

If we predict that the branches are always taken, Tomasulo's Algorithm will allow multiple executions of this loop to proceed at once. Unroll the loop twice and update the following Instruction status table for the unrolled instructions. Consider the following assumptions while answering the question.

- The `MULTD` instruction takes 4 clock cycles. The 1st `LD` instruction takes 8 clock cycles (*L1 cache miss*) and the 2nd `LD` instruction takes 1 clock cycle (*cache hit*). The `SD` instruction takes 1 clock cycle.
- Since, we are only considering floating point operations, the integer ALU operation is ignored and it is assumed that the branch is predicted as taken. However, the issue is always done in-order and there can be only one issue at a particular clock cycle. The instructions `DADDIU` and `BNEZ` consume 1 cycle each.
- The initial content of register `R1` is 80 and value of the scalar s is 10 (stored in `F2`). The contents in address locations #80 and #72 are 20 and 30 respectively.
- There are total 3 execution units - one for load and store, one for multiplication, and one for addition. Also assume that each execution unit has 4 reservation stations.
- Same cycle `ISSUE -> DISPATCH` and `CAPTURE -> DISPATCH` are not allowed.

Instruction	**Issue**	**Execute**	**Write**
LD			
MULTD			
SD			
LD			
MULTD			
SD			