Homework Number 2

This homework is based on the chapter 2 and chapter 4 and focuses primarily on performance and computer arithmetic issues.

We recommend solving the following problems from the textbook

Textbook Problems

- 2.18 through 2.24: This is a series of problems that walk you through design decisions and trade-offs that a processor (and compiler) designer has to go through based on expected and needed performance.
- 4.9: simple problem on the completeness of the MIPS instruction set
- 4.10: understanding positive and negative numbers and the MIPS instruction set.
- 4.14: bits, representations and meaning thereof in various contexts.
- 4.18: double precision integer arithmetic. Note, this problem does not specify whether the result should cause an overflow exception or not. I recommend trying to do this problem with overflow detection as well as without.

Additionally try this

1. Suppose I am trying to compress a video file. On an old department PC with a Pentium processor, it takes me about 10 secs. At my brand new home PC with the latest in Itanium, it takes only 4 secs. Explain
   a) The Itanium was ____ times _____ (faster/slower) than the Pentium
   b) The Itanium was ____ percent _____ (faster/slower) then the Pentium

2. The DRAM growth rule is that the capacity of DRAM increases by a factor of 4 every three years. Figure 1.20 in the text shows that the performance of workstations has gone from 9 "specmarks" (the unit of performance for the SPEC benchmark) to 1140 in 10 years. Which is the faster growth rate? Note: there are (at least) two ways to compute this - you could figure out how much growth there would be in 30 years (assuming the growth rates remain the same), or you could determine an average yearly growth rate for each measure. The best exercise would be to try it both ways.

3. Our old friend the FIR filter:
   A finite impulse response (FIR) filter in signal processing, with N taps, is usually represented with the following piece of code:

```
#define N 26
int fir(const int *w, const int *d)
{
    int sum=0;
    int i; // fine I'll actually declare this ☻
    for(i=0;i<N;i++) sum += w[i]*d[i];
    return sum;
}
```

You developed the MIPS assembly code for this in the last Homework.

CSE141-Summer 2003
Assume a machine implementation of 1 clock per MIPS instruction. Assume that the clock speed is 100 MHz.

a) Calculate the performance of this filter (number of taps/second). {Hint: find execution time per loop-step}

b) If the number of taps is given to be 3 (viz., N=3) can you optimize and increase the performance?

c) Comment on the overflow properties of the “sum”.