CSCI 120 Introduction to Computation
Homework 5
Due 04/16/09

Saad Mneimneh
Visiting Professor
Hunter College of CUNY

PART 1: Pipelining...
Consider a machine with the following machine cycle:

Fetch: instruction is fetched
Decode: instruction is decoded
Execute: instruction is executed
Write-back: result is written back to memory

Therefore, a normal (non-pipelined) execution will look like the following:

F D E WB F D E WB F D E WB F D E WB F D E WB F D E WB ...

Since both fetch and write-back access memory, they cannot be performed simultaneously. Show how a pipelined machine cycle will look like using 9 instructions. How much speedup is gained over the non-pipelined execution?

PART 2: Cache memory
The CSCI 120 teacher collected all the tests and went to his office in HN 1090F. Few seconds after being in the office, he received a call on his cell phone from his wife to come home, so he left the building carrying the tests with him. He got into a cab on 68^{th} street and Lexington avenue and went home. At home, and while talking to his wife, he suddenly realized that the tests are no longer with him! He figured that he must have forgotten them in the cab, and this must have happened because he placed them on the seat when he was paying the driver. Unfortunately, while he always makes sure to remember the four digit cab number, this time he didn't. Therefore, all the tests are gone. The next day, when he got into his office, he saw the tests on his desk.

(a) Exaplain how sometimes we can be tricked by our memory and why inconsistencies such as the one described above could occur.

(b) Describe the above scenario from a machine's perspective using memory and cache memory and how this inconsistency could occur because of a failure to invalidate cache. Use your imagination.

PART 3: Computing the absolute value
Assume the machine uses 2's complement representation of numbers. Write a program using the instruction set presented in class to perform the following:

if memory location 103 contains a positive number, keep it the same; otherwise if negative, chance it to a positive.

Example: if memory location 103 contains 00100101 (that's 37) then nothing is to be done, but if it contains 10010010 (that's -110), then memory location 103 should be changed to 01101110 (that's 110).

The idea is the following: Compare 0 to the number (using the CMP instruction). If 0 is greater than the number (i.e. number is negative), jump to a location in memory where you negate the number. The jump instruction must rely on the result of the compare; therefore, it must use the register that holds the result of the comparison.