Project #3
EGCP 4210

Develop a simulation of a pipelined processor which executes a subset of the MIPS instruction set. You will work in teams of 2.

Specific instructions / discussion:

a. Your simulation should be done at about the granularity of the pipelined processor from Figure A.18 of the text. In addition to the 4 pipeline registers shown in the diagram, you can consider the PC to be a pipeline register. During each clock cycle of the simulation, your task will be to calculate the next value of each pipeline register. Additionally, your simulation must track the values in memory.
b. You do not have to concern yourself with a memory hierarchy; rather, you can just assume 100% hit rate in cache and that all memory operations can be done in 1 clock cycle.
c. I recommend you develop the project in stages. For example, get the simulation working without any branches or without any dependences requiring forwarding or interlocks. Then slowly add in detail to your simulation, after you have the basics working. You can insert NOPs to hide dependences till you get that part working.
d. One of the first items you will need to address is to understand the control signals which need to be present, and when/how they will be generated. You can then specify the required fields of the pipeline registers.
e. You will need to model the proper handling of branches, according to the branch prediction scheme you envision for your hardware. This may involve nullifying instructions or halting the pipeline.
f. You will need to model forwarding, and interlocking where forwarding isn't effective.
g. Handling of exceptions is not required.
h. Your simulator should provide a high-level menu-based interface (I don't mean GUI), which allows for such things as clearing memory, setting memory locations, viewing memory locations, and running the simulation. You should consider providing not just a "go," but also a single-step mode that displays state after each clock cycle. You might also consider allowing the program/data for a particular run to be loaded from file. I have provided a rough outline which you are welcome to use.
i. Your simulation must handle the HALT instruction, allowing the simulation to stop when the HALT reaches writeback.
j. One of the primary outputs of the simulation will be a count of clock cycles required.
k. You may want to consider the use of a log file, since the data generated by the simulation could be sizable. You might consider a verbose/quiet mode also to manage the amount of data.

Project #3
EGCP 4210

l. All data to run a program must start in memory. You cannot assume any values will be in register at the beginning of the program.
m. Your simulation should be developed in either C++ or Java. Code that I provide will be in Java.
n. To limit the scope of the simulation, you only have to deal with one data size, which should be the width of the datapath. The assembler I use assumes 32-bit words; you don't have to worry about 8, 16, and 64 bit sizes.
o. Your simulation only needs to handle the MIPS subset provided in the handout "MIPS32 Subset for EGCP 4210 Projects."

Code framework provided – I will provide you a Java framework for the project, to give you an outline of the project, and provide some of the mundane support functionality. This code has had limited testing, and I do not guarantee its functionality.

a. I provide you with an assembler, which will translate your program into an object file, of a format I created.
b. I provide you with a class called MemoryModel, which knows how to parse he object file and initialize the program and any program data into memory. It provides an interface allowing you to read instructions and to load and store data.
c. I provide you with an outline on a simple GUI-based system that allows the user to load, single step, run, use breakpoints, and view memory. Your main task till be to create the functionality to simulate what occurs each clock cycle in your hardware.

Required for turn-in:

a. Instructions for the use of your simulator.
b. A discussion of the simulation; this will likely require 2-3 pages. In particular, discuss why certain things are modeled the way they are in your simulation: e.g., why do you handle branches the way your do, or how you model interlocking. Clearly state assumptions; e.g., when you assume branches resolve, and thus what branch penalty you expect. Specify details such as your branch prediction technique; e.g., predict-not-taken.
c. Your report should include a diagram of the hardware you are simulating (ala Fig A.18).
d. An report on the results from running your two programs from Project #2, including correctness, clock cycles required, number of stall cycles, and any problems and/or lessons learned from the simulation.