Pipeline Processor Simulation

In the extra-credit project you are to simulate a pipeline processor using (early) hazard detection and stalling, i.e., no data forwarding. You need to use the existing project code as the basis and make modifications based on the hazard detection algorithms discussed in the text. The extra-credit project is worth 40 points that may be added to your project's grade (100 points), i.e., the extra credits are worth an extra 40% of the project grade.

The attached spreadsheet describes the execution of a sample program of MIPS instructions using a pipeline processor. Your pipeline processor simulator should take the MIPS machine code of such program and simulate the execution according to the timeline described in the spreadsheet. The extra credit consists of a design document and an implementation (a C program); they are worth 20 points each.

The following are some guidelines and requirements:

(1) Your program will use the existing code spimcore.c, spimcore.h, and project.c as the basis, modify them to simulate the pipeline processor.

(2) You need to add/modify (at least) the following data structures and code to the main program file "spimcore.c":

- ➢ stall-count: a global variable that counts the remaining clock cycles before the next instruction can resume execution
- ➢ pipeline registers: per the textbook's discussions you need to add 4 pipeline registers (IF/ID, ID/EX, EX/MEM, and MEM/WB), each hold necessary control signals to facilitate the pipeline execution; output their contents along with registers $0 - $31 when the user enters the "r" command during simulation
- ➢ Halt condition: the simulation will "terminate" only after all instructions in the pipeline are complete and if one of the existing halt condition exists:
  - ➢ an illegal instruction is encountered
  - ➢ Jumping to an address that is not word-aligned (not a multiple of 4).
  - ➢ The address of a lw or sw is not word-aligned, or a lh is not halfword aligned
  - ➢ Accessing data or jumping to an address that is beyond the end of memory.).
- The function Step() needs to be revised so that each "step" (i.e., when the user enters command "s") corresponds to advancing one clock cycle. Thus, each clock cycle requires executing all components of the processor (if they are not stalled), by calling the functions implemented in project.c. Also, each cycle requires saving results to pipeline registers.
- Hazard detection: detect both data and control hazards as early as possible; use stall (no data forwarding) to resolve hazardous situations.

(3) You also need to change some parts of project.c; for example, the function PC_update() needs to check the halt condition, whether the next instruction is stalled, and, if so, decrement stall-count.

(4) There will be a separate link on WebCourses for submitting the extra-credit project. Every team/person may submit the design document (PDF or Word file) and earn extra credits. In the design describe the modifications needed for the pipeline simulation, including pseudo-code description of the program code or data structure.

(5) You should have a working program for the original project before attempting to implement the code for the extra credit. More specifically, if your project.c code is not at least 85% working, then your extra-credit project code will not be graded. (However, you could still submit the design and earn extra credit).

(6) The extra-credit project is due 9 pm on Friday, 8/3/2012, to WebCourses.