Computer Architecture - CS2323. Autumn 2023

Lab-5 (Pipeline Stall Detector/Simulator)

In this assignment, we will identify the instructions in a given program where a pipeline might stall. The input will be an assembly program and the output will be a modified assembly program with added nops (no-operation) to make it stall-free.
Also, we would output the total number of cycles that the given assembly program will take. Consider the standard 5-stage pipeline discussed in the class.

Example-1: If the input is:
add t0, t1, t2
sub t4, t0, t3

The output should be (due to a dependence between the two instructions):
add t0, t1, t2
nop
nop
sub t4, t0, t3
Total: 8 cycles

Example-2: If the input is:
ld t0, 0(t2)
sub t4, t0, t3

The output should be (due to a dependence between the two instructions):
ld t0, 0(t2)
nop
nop
nop
sub t4, t0, t3
Total: 9 cycles

However, if data forwarding and hazard detection was implemented, then the outputs for the above examples would be:
add t0, t1, t2
sub t4, t0, t3
Total: 6 cycles

and
ld t0, 0(t2)
nop
sub t4, t0, t3
Total: 7 cycles

Problem Statement: Implement a C/C++ program to analyze a given assembly program and indicate the stalls as shown in above examples. Please note that the register names in the input program can be specified as x0, x1, … x31 or using their calling convention based aliases like a0, t0, s0, … or a mix of these. The output should retain the same register name as in the input program.
The output program should indicate the nops (i) assuming no data forwarding and no hazard detection is implemented, (ii) assuming that data forwarding without hazard detection is implemented.

You may make the following assumptions about the input program (if it helps in your implementation):

1. The program is syntactically correct (there is no syntax error)
2. There is only 1 space in between the instruction and the first operand. Also, only 1 space between “,” and second operand and so on.
3. The program starts on the first character in each line
4. There is only one instruction in each line
5. There are no blank lines in the input file
6. No branch/jump instructions are used
7. There are no labels used in the program (to ease the parsing of the assembly code)

Submission instructions:

The assignment is to be done individually. Submit your code and instructions to compile/execute your code should be put in a separate README file. Prepare a short report on your coding approach and what all you did for testing your code to be correct. Submit a zip file containing the following:

1. Source files of code
2. README: A text file containing instructions to compile/execute your code
3. Testcases, if you tried out specific input files
4. report.pdf - a short report on your coding approach and what all you did for testing your code to be correct. If a proper report is not submitted, you will receive 10% lesser marks.
5. Submission deadline: 07 November 2023, 11:59 PM