CSCI 2500 — Computer Organization
Homework 5 (document version 1.0)
Pipelining in MIPS

Overview

- This homework is due by 11:59:59 PM on Tuesday, December 4, 2018.
- This homework is to be completed **individually**. Do not share your code with anyone else.
- You **must** use C for this homework assignment, and your code **must** successfully execute on Submitty to obtain full credit.

Homework Specifications

For this individual homework assignment, you will use C to implement a simulation of MIPS pipelining. As we've covered in lecture, there are five stages to the pipeline, i.e., `IF`, `ID`, `EX`, `MEM`, and `WB`.

For your simulation, you are required to support the `add`, `sub`, `and`, `or`, `lw`, and `sw` instructions; note that some of these are pseudo-instructions, which is fine. More specifically, you must simulate (and output) how a given sequence of instructions would be pipelined in a five-stage MIPS implementation.

Do **not** implement forwarding in your simulation.

You can assume that each given instruction will be syntactically correct. You can also assume that there is a single space character between the instruction and its parameters. Further, each parameter is delimited by a comma or parentheses. Below are a few example instructions that you must support:

```
add $t0,$s2,$s3
add $t1,$t3,73
or $s0,$s0,$t3
lw $a0,12($sp)
sw $t6,32($a1)
```

Required Command-Line Argument

Your program must accept one command-line argument as input. This argument (i.e., `argv[1]`) specifies the input file containing MIPS code to simulate. You may assume that no more than five instructions are given in the input file. And note that each instruction will end with a newline character (i.e., `'\n'`).

Required Output

For your output, you must show *each cycle* of program execution. Each cycle will correspond to a column of output. Initially, each column is empty, indicated by a period (i.e., `'.'`). Use TAB characters (i.e., `'\t'`) to delimit each column. And assume that you will have no more than nine cycles to simulate.

Recall that a *data hazard* describes a situation in which the next instruction cannot be executed in the next cycle until a previous instruction is complete. Your code should be able to detect when it is necessary to insert one or more "bubbles" (see Section 4.7 of the textbook and corresponding lecture notes for more details).

More specifically, you will need to properly handle data hazards by adding `nop` instructions as necessary. Show these cases by indicating an asterisk (i.e., `'*'`) in the appropriate columns and adding the required number of `nop` instructions. To ensure proper formatting, add an extra TAB character after the `nop`.

On the next few pages, we present a few example runs of your program that you should use to better understand how your program should work, how you can test your code, and what output formatting to use for Submitty.

The first example (i.e., `ex01.s`) includes no data hazards.

```
START OF SIMULATION

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $s1,$s0,$s0 IF      .       .       .       .       .       .       .       .
add $t2,$s0,$s5 .       .       .       .       .       .       .       .       .
add $t4,$s3,70  .       .       .       .       .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $s1,$s0,$s0 IF      ID      .       .       .       .       .       .       .
add $t2,$s0,$s5 .       IF      .       .       .       .       .       .       .
add $t4,$s3,70  .       .       .       .       .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $s1,$s0,$s0 IF      ID      EX      .       .       .       .       .       .
add $t2,$s0,$s5 .       IF      ID      .       .       .       .       .       .
add $t4,$s3,70  .       .       IF      .       .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $s1,$s0,$s0 IF      ID      EX      MEM     .       .       .       .       .
add $t2,$s0,$s5 .       IF      ID      EX      .       .       .       .       .
add $t4,$s3,70  .       .       IF      ID      .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $s1,$s0,$s0 IF      ID      EX      MEM     WB      .       .       .       .
add $t2,$s0,$s5 .       IF      ID      EX      MEM     .       .       .       .
add $t4,$s3,70  .       .       IF      ID      EX      .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $s1,$s0,$s0 IF      ID      EX      MEM     WB      .       .       .       .
add $t2,$s0,$s5 .       IF      ID      EX      MEM     WB      .       .       .
add $t4,$s3,70  .       .       IF      ID      EX      MEM     .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $s1,$s0,$s0 IF      ID      EX      MEM     WB      .       .       .       .
add $t2,$s0,$s5 .       IF      ID      EX      MEM     WB      .       .       .
add $t4,$s3,70  .       .       IF      ID      EX      MEM     WB      .       .

END OF SIMULATION
```

The second example (i.e., `ex02.s`) includes a dependency on register `$t1`.

```
START OF SIMULATION

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $t1,$s0,$s0 IF      .       .       .       .       .       .       .       .
add $t2,$s0,42  .       .       .       .       .       .       .       .       .
add $t4,$t1,70  .       .       .       .       .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $t1,$s0,$s0 IF      ID      .       .       .       .       .       .       .
add $t2,$s0,42  .       IF      .       .       .       .       .       .       .
add $t4,$t1,70  .       .       .       .       .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $t1,$s0,$s0 IF      ID      EX      .       .       .       .       .       .
add $t2,$s0,42  .       IF      ID      .       .       .       .       .       .
add $t4,$t1,70  .       .       IF      .       .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $t1,$s0,$s0 IF      ID      EX      MEM     .       .       .       .       .
add $t2,$s0,42  .       IF      ID      EX      .       .       .       .       .
add $t4,$t1,70  .       .       IF      ID      .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $t1,$s0,$s0 IF      ID      EX      MEM     WB      .       .       .       .
add $t2,$s0,42  .       IF      ID      EX      MEM     .       .       .       .
nop             .       .       IF      ID      *       .       .       .       .
add $t4,$t1,70  .       .       IF      ID      ID      .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $t1,$s0,$s0 IF      ID      EX      MEM     WB      .       .       .       .
add $t2,$s0,42  .       IF      ID      EX      MEM     WB      .       .       .
nop             .       .       IF      ID      *       *       .       .       .
add $t4,$t1,70  .       .       IF      ID      ID      EX      .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $t1,$s0,$s0 IF      ID      EX      MEM     WB      .       .       .       .
add $t2,$s0,42  .       IF      ID      EX      MEM     WB      .       .       .
nop             .       .       IF      ID      *       *       *       .       .
add $t4,$t1,70  .       .       IF      ID      ID      EX      MEM     .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
add $t1,$s0,$s0 IF      ID      EX      MEM     WB      .       .       .       .
add $t2,$s0,42  .       IF      ID      EX      MEM     WB      .       .       .
nop             .       .       IF      ID      *       *       *       .       .
add $t4,$t1,70  .       .       IF      ID      ID      EX      MEM     WB      .

END OF SIMULATION
```

The third example (i.e., `ex03.s`) includes two dependencies on register `$t2`.

```
START OF SIMULATION

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
lw $t2,20($a0)  IF      .       .       .       .       .       .       .       .
and $t4,$t2,$t5 .       .       .       .       .       .       .       .       .
or $t8,$t2,$t6  .       .       .       .       .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
lw $t2,20($a0)  IF      ID      .       .       .       .       .       .       .
and $t4,$t2,$t5 .       IF      .       .       .       .       .       .       .
or $t8,$t2,$t6  .       .       .       .       .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
lw $t2,20($a0)  IF      ID      EX      .       .       .       .       .       .
and $t4,$t2,$t5 .       IF      ID      .       .       .       .       .       .
or $t8,$t2,$t6  .       .       IF      .       .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
lw $t2,20($a0)  IF      ID      EX      MEM     .       .       .       .       .
nop             .       IF      ID      *       .       .       .       .       .
nop             .       IF      ID      *       .       .       .       .       .
and $t4,$t2,$t5 .       IF      ID      ID      .       .       .       .       .
or $t8,$t2,$t6  .       .       IF      IF      .       .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
lw $t2,20($a0)  IF      ID      EX      MEM     WB      .       .       .       .
nop             .       IF      ID      *       *       .       .       .       .
nop             .       IF      ID      *       *       .       .       .       .
and $t4,$t2,$t5 .       IF      ID      ID      ID      .       .       .       .
or $t8,$t2,$t6  .       .       IF      IF      IF      .       .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
lw $t2,20($a0)  IF      ID      EX      MEM     WB      .       .       .       .
nop             .       IF      ID      *       *       *       .       .       .
nop             .       IF      ID      *       *       *       .       .       .
and $t4,$t2,$t5 .       IF      ID      ID      ID      EX      .       .       .
or $t8,$t2,$t6  .       .       IF      IF      IF      ID      .       .       .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
lw $t2,20($a0)  IF      ID      EX      MEM     WB      .       .       .       .
nop             .       IF      ID      *       *       *       .       .       .
nop             .       IF      ID      *       *       *       .       .       .
and $t4,$t2,$t5 .       IF      ID      ID      ID      EX      MEM     .       .
or $t8,$t2,$t6  .       .       IF      IF      IF      ID      EX      .       .
```

```
CPU Cycles ===> 1       2       3       4       5       6       7       8       9
lw $t2,20($a0)  IF      ID      EX      MEM     WB      .       .       .       .
nop             .       IF      ID      *       *       *       .       .       .
nop             .       IF      ID      *       *       *       .       .       .
and $t4,$t2,$t5 .       IF      ID      ID      ID      EX      MEM     WB      .
or $t8,$t2,$t6  .       .       IF      IF      IF      ID      EX      MEM     .

CPU Cycles ===> 1       2       3       4       5       6       7       8       9
lw $t2,20($a0)  IF      ID      EX      MEM     WB      .       .       .       .
nop             .       IF      ID      *       *       *       .       .       .
nop             .       IF      ID      *       *       *       .       .       .
and $t4,$t2,$t5 .       IF      ID      ID      ID      EX      MEM     WB      .
or $t8,$t2,$t6  .       .       IF      IF      IF      ID      EX      MEM     WB

END OF SIMULATION
```

Assumptions

Given the complexity of this assignment, you can make the following assumptions:

- Assume all input files are valid.
- Assume the length of `argv[1]` is at most 128 characters, but do not assume that `argv[1]` is present.

Error Checking

Given the complexity of this assignment, you can assume that the input file is valid. Be sure to verify that you have the correct number of arguments by checking `argc`; display an error message if argument(s) are missing.

In general, if an error occurs, use either `perror()` or `fprintf( stderr, "..." )`, depending on whether the global `errno` is set.

And be sure to return either `EXIT_SUCCESS` or `EXIT_FAILURE` upon program termination.

Submission Instructions

Before you submit your code, be sure that you have clearly commented your code (this should not be an after-thought). Further, your code should have a clear and logical organization.

To submit your assignment (and also perform final testing of your code), please use Submitty. Note that the test cases for this assignment will be available on Submitty a minimum of three days before the due date and will include hidden test cases.

Also as a reminder, your code **must** successfully execute on Submitty to obtain credit for this assignment.