ECE 474/774: Computer Architecture

Project 1 (Due: October 27, 2017)

The goal of this project is to write a simulator for Tomasulo's algorithm. You can implement the simulator using any one of the following programming languages C/C++/Java/Python. If you want to use another programming language, kindly discuss with the course instructor before proceeding. The project is do be done in groups of two.

Processor Configuration

The processor only supports 4 types of instructions: integer add, integer sub, integer multiply, and integer divide. The processor has two units: integer add unit and integer multiply unit. The add unit can perform addition and subtraction operations. Add and subtract operations each take two cycles to execute. The add unit has three reservation stations (RS1-RS3). The multiply unit can perform multiply and divide operations. Multiply operations require 10 cycles to execute. Divide operations require 40 cycles to execute. The multiply unit has two reservation stations (RS4-RS5). The processor has 8 registers (R0-R7). The processor does not allow same-cycle issue and dispatch. The processor does not allow same-cycle capture and dispatch. If a reservation station is freed in a certain cycle, it cannot be allocated again in the same cycle. If both the add unit and the multiply unit try to broadcast in the same cycle, the multiply unit will have precedence. If two or more instructions are ready to be dispatched in the same cycle in the add unit, the instruction in RS1 will have the highest priority, followed by RS2, followed by RS3. Similarly, if two instructions are ready to be dispatched in the same cycle in the multiply unit, the instruction in RS4 will have priority over RS5. The processor has an instruction queue that can hold up to a maximum of 10 instructions.

Simulator

To implement the simulator, you will need the following data structures. Instructions can be stored in the queue as instruction records. Each instruction record will have four fields: (1) opcode; (2) destination operand; (3) source operand 1; and (4) source operand 2. A 10-entry array of instruction records can be used as the instruction queue. Each cycle, the instruction queue array should be updated as a queue. An 8-entry array of integers can be used as the register file. An 8-entry array of integers can also be used for the RAT. The RAT array will store the tag. For example, RS1 can be recorded as the value 1. A special value can be stored to identify the situation where a register is currently not tagged with a reservation station. Reservation stations can also be represented as records with several fields.

The input to the simulator will be a text file that has the following format:

```
2
7
0 2 4 6
2 4 3 5
3
4
6
21
3
4
9
0
```

The first line indicates the number of instructions. The second line indicates the number of cycles of simulation. The third line onward are the encoded instructions. For example 0 2 4 6 is the instruction R2=R4+R6. The first number indicates the instruction opcode (0: add; 1: sub; 2: multiply; 3: divide). The second number indicates the destination register. The third and the fourth numbers indicate the source registers. After the list of encoded instructions, the input file should have the initial values to the RF. The output of the simulator should display the state of the reservation stations, register file, RAT and instruction queue after simulating the processor for n cycles, where n is obtained from the second line of the input file. Sample display is given below:

```
         Busy      Op     Vj     Vk       Qj       Qk    Disp
RS1        1       Add            3       RS2              1
RS2        0
RS3        0
RS4        0
RS5        0

      RF                  RAT
0:    4                   RS4
1:    29
2:                        RS2
3:
4:    3
5:
6:
7:

Instruction Queue
Add, R2, R4, R7
Sub R6, R1, R1
```

Submission Instructions

Email your project as a zipped folder to Yifu Gong (yifu.gong@ndsu.edu, cc: sudarshan.srinivasan@ndsu.edu) with subject exactly "ECE474 Project 1." The submission folder should include your code and a README file that includes information about the project members and instructions for compiling and executing your simulator on one of the machines in the ECE 121 cluster. *Project is due end of the day October 27, 2017.*