DCC006: Computer Organization I **(Due: 21/10/19 ◷ 23:59)**

Submission Assignment #2

Professor: Omar Paranaiba Vilela Neto *Teacher Assistant: Angelica Moreira*

Course Policy: Read all the instructions below carefully before you start working on the assignment, and before you make a submission.

- Please typeset your submissions in LaTeXand include your name and ID with submission.
- Late assignments can be submitted on moodle but there will be a penalty for each day you delay, and the penalty function is in NP ☹.
- You shall do this assignment alone ☺.
- All sources of material must be cited ✓.
- The goal of this project is to help you pin the concepts taught by professor Omar about the principles and techniques used in implementing a processor. To complete this assignment with success you should read chapter 4 of the textbook [2] and the support material [1], available at moodle. For this assigment you may want to use the Ripes Simulator (https://github.com/mortbopet/Ripes). Ripes is a graphical 5-stage processor pipeline simulator and assembly code editor that allows you to follow along what happens at each step of execution of your code. This simulator is developed by Morten Borup Petersen and has the same ISA as RISC-V, although it has some modifications. You can check the url: https://github.com/mortbopet/Ripes/wiki/RISC-V-Assembly-Programmer%27s-Manual-(Adapted-for-Ripes) to see the modifications of the ISA syntax used by the simulator. Be aware that the class material and the textbook use the 64-bit wide registers but in this simulator we're using 32-bit wide registers. Using the Ripes simulator is really simple, you will see it is easy peasy lemon squeezy!!!!

Problem 1: 5-Stage Pipeline (1.5 points)

Gabriela has just translated a code in C++ to RISC-V assembly, and she wants to optimize her code performing loop unrolling. But, before she does that, she wants to find out how many cycles each loop iteration takes. Thus your task is to help her to find out how many cycles each loop iteration takes. Use nops to denote stalls.

Hint: Be aware that this is a 5-stage pipeline with bypassing, and maybe Ripes can help you answer it ☺.

Listing 1: code 1

```
int main(int argc,char** argv)
{
  int N = atoi(argv[0]);
  int array[N];
  for(int i=0; i< N; i++)
   array[i] = atoi(argv[i+1]);

  int total = 0;
  for(int i=0; i<N; i++)
    total += array[i];

  cout<<total<<endl;

  return 0;
}
```

Listing 2: code 2

```
.data
  # User input
   array: .word 1 2 3 4 5 6 7 8 9 10
   array_size: .word 1

.text
main:
    la x8, array
    la x5, array_size
    lw x6, 0(x5)  # N
    addi x9, x8, 0  # x = &array[0]
    addi x10, x0, 0 # total = 0
    addi x11, x0, 0 # i = 0
    loop:
        lw x12, 0(x9)   # x12 = array[i]
        add x10, x10,x12# sum += array[i]
        addi x9, x9, 4  # &array[i++]
        addi x11, x11, 1# i++
        blt x11, x6, loop

     addi a1, x10, 0 # a1 will stores the result
     addi a0, x0, 1 # Making ecall be the output
     ecall # Printing the result
     jal  zero, end
end:
```

Problem 2: Study Questions (1.5 points)

The following study questions were originally elaborated by professor Yonghong Yan, you can find them at https://passlab.github.io/CSE564/exercises.html.

1. What are the five stages of a typical RISC CPU pipeline?

What are pipeline hazards, and how do they happen? Give examples of methods to resolve hazards.

Problem 3: Exercise 4.22 reproduced from the textbook (2.0 points)

4.22 [5] <§4.5> Consider the fragment of RISC-V assembly below:

Listing 3: code 3

```
sd x29, 12(x16)
ld x29, 8(x16)
sub x17, x15, x14
beqz x17, label
add x15, x11, x14
sub x15, x30, x14
```

Suppose we modify the pipeline so that it has only one memory (that handles both instructions and data). In this case, there will be a structural hazard every time a program needs to fetch an instruction during the same cycle in which another instruction accesses data.

4.22.1 [5] <§4.5> Draw a pipeline diagram to show were the code above will stall.

4.22.2 [5] <§4.5> In general, is it possible to reduce the number of stalls/NOPs resulting from this structural hazard by reordering code?

4.22.3 [5] <§4.5> Must this structural hazard be handled in hardware? We have seen that data hazards can be eliminated by adding NOPs to the code. Can you do the same with this structural hazard? If so, explain how. If not, explain why not.

References

[1] Omar P. V. Neto. Aula 6 – o processador – ciclo unico. https://virtual.ufmg.br/20192/pluginfile.php/392438/mod_resource/content/1/Aula06_nova.pdf, 2019.

[2] David A. Patterson and John L. Hennessy. *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2017.