# ECSE-487 — Assignment 2
A RTL-Level Implementation of a Pipelined RISC CPU Core

## 1 Important Information

This assignment is due **February 9, 2008 at 2:30 pm**. You can work in a group of a maximum of three students. Submission is done both by WebCT Vista and in the assignment box in Trottier. The submission is not complete without both hardcopy and electronic copy (in PDF format) and the time of the latest will be the submission time. Read the entire handout carefully before starting and start early!

## 2 Introduction

In this assignment you will implement a simplified version of the MIPS-64 architecture. The instruction set reference manual is posted on WebCT. The course textbook for ECSE 425 [1] is a good reference for implementing your design. The textbook [2] is also a good reference for implementing similar (but 32-bit) MIPS processors.

## 3 Requirements

Implement a 5-stage pipelined-version of the MIPS-64 processor to support the subset of the instruction set shown below. Just implement the datapath and control unit to handle the execution of these instructions, you do not need to implement the other parts of the processor, for example, exception handling.

```
NOP
LD, SD
DADD, DADDI, DADDU, DADDUI
DSUB, DSUBI, DSUBU, DSUBUI
AND, ANDI
OR, ORI, XOR, XORI
DSLL, DSRL
```

Your control unit should catch data hazards and react accordingly. It is your choice how to handle data hazards. Develop a testbench to exercise and test your model. Your testing strategy should include corner cases and show that all required operations were properly implemented. Show cases with and without data hazards.

Synthesize your design without the testbench and report on the results including the hardware resources used and clock frequency of your circuit. How long and where is the critical path of the circuit?

### 3.1 Important Considerations

You might find it useful to use VHDL *records* (similar to C *structs*) to group signals and make your code easier to maintain and debug. Note that some synthesis tools do not like it when a single record is used to connect to input and output ports of a module. For that reason, you should use one record to connect inputs and another to connect outputs (i.e. when doing port maps). The following is an example of VHDL records:

```
type reg_t is std_logic_vector(31 downto 0);

type CPU_Reg_t is record
  PC : reg_t;
  IR : reg_t;
  MAR: reg_t;
end record;

signal CPU_Regs : CPU_Reg_t;

...

p1: process(clk)
begin
  if rising_edge(clk) then
     CPU_Regs.PC <= CPU_Regs.PC + 1;
  end if;
end process;
```

## 4 Tools

Two MIPS-64 simulators are:

- EduMIPS64 (`http://www.edumips.org`)
- WinMIPS64 (`http://www.computing.dcu.ie/~mike/winmips64.html`).

EduMIPS64 handles data hazards in the ID stage, while WinMIPS64 may handle data hazards in later pipeline stages. In both simulators you can select whether or not data hazards are handled with forwarding. WinMIPS64 includes an assembler which is useful for generating the machine code to run on your processor.

### 4.1 CPU Tracing

Think about adding messages during the simulation to report what the CPU is executing. You can also propagate signals in your execution pipeline only to trace information. If you don't use the result at any stage of the pipeline, the synthesis tool will likely remove the signals and the corresponding flip-flips. However, those signals will still exist in the simulation and they can help you develop your code.

### 4.2 Lab Report

The maximum length of this report is 8-pages excluding the title page, diagrams and appendices. The hardcopy should not include the VHDL code, but include it with your electronic version. You should describe your implementation and the design choices you made. Use diagrams to illustrate. Include simulation traces to support your description and discuss the results. Include the synthesis results.

## References

[1] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. The Morgan Kaufmann Series in Computer Architecture and Design, San Francisco, California, fourth edition.

[2] David A. Patterson and John L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. 1998.