# CS4342 Advanced Computer Architecture

Take Home Lab 1

**Due – June 18 before 11:55 PM**

## Learning Outcomes

In this lab we will simulate the instruction execution on a 5-stage MIPS pipeline. We will also modify the order of instruction execution to gain better performance. At the end of the lab you will be able to:

- understand the operations of a 5-stage pipeline, what causes common pipeline hazards, and how to overcome them
- develop better Assembly programs that runs in lower number of CPU cycles while increasing the CPU utilization

## Getting Started

For this lab, we will use WinMIPS64 which is an instruction set simulator developed by Mike Scott based on the MIPS architecture described in our textbook.

**Step 1:** Download the latest version of WinMIPS64 from http://indigo.ie/~mscott/ .

**Step 2:** Read the documentation (Using WinMIPS64 Simulator) available on the same web site (also available on Moodle).

## Task 1

**Step 1:** Copy the following Assembly code to the text editor and save it as **sum1.s**. Comment the code to explain its functionality. [2 marks]

```
        .data
A:      .word 10
B:      .word 8
C:      .word 0

        .text
main:
        ld r4,A(r0)
        ld r5,B(r0)
        dadd r3,r4,r5
        sd r3,C(r0)
        halt
```

**Step 2**: Load the program into WinMIPS64 and execute it one instruction at a time. While you are executing the above program note the animation on **Cycles** window. While the execution is going on, note the numbers on the **Statistics** window. Continue to run the program until halt line is executed.

**Step 3**: Repeat **Step 2** with the following options and document your findings (use options in the **Configure** menu): [4 marks]

| | Total no of Cycles | No of Instructions | CPI | Type of Stalls |
|---|---|---|---|---|
| **Without pipeline register forwarding** | | | | |
| **With pipeline register forwarding** | | | | |
| **With delay slot enabled** | | | | |
| **With pipeline register forwarding and delay slot enabled** | | | | |

For rest of the lab enable only the pipeline register forwarding option (**Configure → Enable Forwarding**) unless otherwise stated.

**Step 4**: Consider the following extension to **sum1.s (**save it as **sum2.s)**. Run the program and find the total number of cycles and CPI. Can you improve this program so that it runs faster (i.e., lower the number of cycles). If so, write and test the modified program. [2 marks]

```
      .data
A:    .word 10
B:    .word 8
C:    .word 0

      .text
main:
      ld r4,A(r0)
      ld r5,B(r0)
      dadd r3,r4,r5
      daddi r6,r6,2
      sd r3,C(r0)
      halt
```

How would you improve the program if we replace the "**`daddi r6,r6,2`**" instruction with "**`daddi r4,r4,2`**". Save this program as **sum3.s**. [2 marks]

## Task 2

**Step 1:** Assume we have a double *x* and a positive integer *n*. We want to raise *x* to the *n*-th power (i.e., $x^n$). Here is one (pretty dumb) way of doing it (see the document entitled "Using WinMIPS64 Simulator" for the instruction set).

```
      .data
n:    .word 8
x:    .double 0.5

      .text
      ld          r1,n(r0)
      l.d         f0,x(r0)
      daddi       r2,r0,1     ; r2 = 1
      mtc1        r2,f11      ; f11 = 1
      cvt.d.l     f2,f11      ; f2 = 1
loop: mul.d       f2,f2,f0    ; f2 = f2*f0
      daddi       r1,r1,-1    ; decrement r1 by 1
      bnez        r1,loop     ; if r1 != 0 continue

; result in F2
      halt
```

Save the program as **power1.s**.

**Step 2:** Run this program and find the total number of cycles, total number of instructions, and CPI. While executing the program note the animation on **Pipeline** window. What type of stalls are encountered. [3 marks]

**Step 3:** **Enable Delay Slot** option. Find the total number of cycles and CPI again. What type of stalls are encountered. [2 marks]

**Step 4:** Remove **Enable Delay Slot** option. Next implement the following simple algorithm in Assembly to calculate $w = x^n$. Save it as **power2.s**. Try and minimize the number of clock cycles and save the optimized program as **power3.s**. How much improvement do you gain in terms of speed up? [5 marks]

```
w=1;
forever
{
	if (n%2!=0)
		w*=x;
	n/=2;
	if (n==0)
		break;
	x*=x;
}
```

**What to Submit**

- Submit following files as a single .zip file
  - sum1.s
  - sum2.s
  - sum3.s
  - power1.s
  - power2.s
  - power3.s
- Submit answers to questions in Steps 2 and 3
- Name the .zip file as lab1_<*index no*>.zip.