

# CSC3420 Computer System Architectures

## Project Phase 3

**Deadline: 23:59 4<sup>th</sup> April, 2009**

### 1. Introduction

This course project is intended to let you have a more in-depth understanding of CPU architecture, by writing an assembler of a simplified assembly language, a single-cycle simulator and a multi-cycle pipelined simulator of a simplified CPU. The project is divided into 3 related phases. In this phase, you have to write a **multi-cycle pipelined simulator**.

### 2. Project Settings

In order to encourage real understanding, each of you is required to finish a slightly different task. The variations and the hashcodes are exactly equal to the settings of phase 1:

- 4 sets of opcodes (set 1 to 4)
- 4 kinds of conditional branches – 0: beq, 1: bne, 2: blt, 3: bge
- Either 16 registers (hash code 0) or 32 registers (hash code 1)

To simplify the design, “**jal**” and “**jr**” instructions need **NOT** be implemented in this phase. No bonus marks will be given even if you implement it.

### 3. Architecture

The architecture (instruction set) you are simulating is exactly the same with phase 1. Figure 1 in this specification shows the datapath you need to simulate. **It should be noted that the datapath in Figure 1 is not a complete datapath. You are required to understand it by yourself in this phase.**

### 4. Simulator

You are required to write a pipelined simulator in standard C (and CANNOT use extra libraries) which simulates the MIPS CPU **cycle-by-cycle**.

## 4.1 Input

Your simulator should be able to accept two arguments, the first one is your student ID and the second one is the filename of your binary assembly program. The main function is provided for you in the provided resources, and you are highly NOT suggested to edit any provided code segments in the skeleton program.

## 4.2 Tasks

You are required to complete the first four methods in the provided `phase3.c` (task 1 to task 4) and to implement methods for every component in the datapath (task 5). **It is forbidden to declare any extra global variables in any files, and it is NOT allowed to modify any methods except the following methods in `phase3.c`**

### 4.2.1 Read Binary File

In `read_machine_code_file` method, you have to read the instructions from the binary file and store them into the memory correctly. Please note that the instructions should be stored at the memory address starting from `0x0000` and the input binary file is in *big-endian*.

### 4.2.2 Connect Component

In `connect_components` method, you have to connect the components according to the datapath properly by constructing linkage from every input to its corresponding value.

### 4.2.3 Initialization

In `initialize_register_states` method, you have to initialize the output states of some components to 0.

### 4.2.4 Simulation Sequence

In `simulate_cycle` method, you have to put down the correct working sequence of the components in one single cycle by calling the component methods sequentially. It should be noted that the simulation sequence is highly critical and sensitive in this phase. **Please choose the sequence wisely.**

### 4.2.5 Component Methods

In this task, you have to implement all the component methods. Please note that usually each component should have ONE component method, but sometime it can have more than one. Such as the Registers Unit, it has more than one different working task in one single cycle which can be reading, writing... etc., therefore it can have more than one component method responding for these tasks. Inside the component method, you have to simulate what it does according to the control signals.

## 5. Marking Scheme

The distribution of marks of phase 3 is as follows:

Part	Marks
Simulator	
Connect the components properly	10%
Correct simulation sequence	20%
Complete the component methods	20%
Test cases	50%

Similar to phase 2, the simulation environment of phase 3 is Linux, Please make sure that your submitted program can be compiled on the server *linux2*, using *gcc*, so please make sure your simulator can be compiled and run correctly under this setting. A sample simulator has been provided to test the behaviors of your simulator.

## 6. Submission Guidelines

You should tar and gzip the following files as *tmchan.tar.gz* and send the file to [csc3420@cse.cuhk.edu.hk](mailto:csc3420@cse.cuhk.edu.hk) with uuencode and mailx command

- For the simulator
  - All *.c* and *.h* files of your simulator
  - A makefile, which when *make* is executed, should compile your simulator properly
- there will be **ZERO tolerance for plagiarism**

## 7. Remarks

- ◇ **Signed comparisons and arithmetics** should be implemented in the ALU
- ◇ Limited by the 16 bit immediate field, all immediates should be between  $-2^{15}$  and  $2^{15}-1$ .
- ◇ The behavior of the sample simulator can be tabulated as follows:

Condition	Number of additional cycles
Jump	1
Branch (if taken)	2
Forwarding	0
Hazard	1
Normal	0

**Figure 1**

