COMPARC **3rd Term 2014-2015**
Case Project

INTRODUCTION

In this case project, you will implement a simulator for a simplified MIPS64 processor, miniMIPS. The miniMIPS processor offers the following subset of MIPS64 instructions:

(For Group 1/2/3/4/5/6)

1. R-type instructions: DADDU, DMULT, OR, DSLLV, SLT
2. I-type instructions: BNE, LW, LWU, SW, DADDIU, ANDI
3. J-type instruction: J

(For Group 7/8/9/10/11)

1. R-type instructions: DSUBU, DDDIV, AND, DSRLV, SLT
2. I-type instructions: BEQ, LW, LWU, SW, DADDIU, ORI
3. J-type instruction: J

The miniMIPS processor is based on the MIPS64 architecture.

The objective this project is to "execute" the program using pipelining with the following schemes to solve the hazards:

- Structural Hazard: Separate Memory
- Data Hazard: No Forwarding
- Control Hazard: Pipeline freeze (Group 3/6/10); pipeline flush (Group 4/7); predict-not-taken (Group 5/9/11); pipeline #2 (Group 1/2/8)

In this case project, you will create the following modules:

1. Utility program to input the MIPS program. You have the option to use **drop-down menu** method or to **let user input the program**
2. Utility program to input value for registers R1 to R31; HI and LO
3. Utility program to input value for memory (data segment). Note: code segment is from address 0000-1FFF while data segment is from 2000-3FFF. Also, provide a "GOTO Memory" option to go to target memory location
4. Write a simulator program using pipeline. Simulator should support (a) **single-step instruction execution** mode and (b) **full execution** mode
5. Output screen #1: the equivalent opcode of the MIPS program (in HEX)
6. Output screen #2: Error message screen
7. Output screen #3: the "pipeline map"
8. Output screen #4: the internal MIPS64 registers as follows:
   IF Cycle: IF/ID.IR, IF/ID.NPC, PC
   ID Cycle: ID/EX.IR, ID/EX.A, ID/EX.B, ID/EX. IMM
   EX Cycle: EX/MEM.IR, EX/MEM.ALUOUTPUT, EX/MEM.B, EX/MEM.cond
   MEM Cycle: MEM/WB.IR, MEM/WB.ALUOUTPUT, MEM/WB.LMD, MEM[ALUOUTPUT]
   WB Cycle: Registers affected

Note: The affected registers (including HI and LO) and affected memory should contain the actual value.
Note: The program should be in an "Integrated Development Environment (IDE)" interface
Note: The program is assume to be stored at address 0

Milestone #1: March 18, 2015: Program input w/ error checking and Opcode (#1,5,6)
Milestone #2: March 30, 2015: GUI, register and memory input, pipeline map (#2,3,7)
Milestone #3: April 13, 2015: Complete program