ASSUMPTION UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
MSCS Program
SC6231 (Advanced Computer Architecture)
Instructor: Asst. Prof. Dr. Anilkumar K.G

TERM PROJECT DESCRIPTION:

Design and simulate an Instruction Set Architecture (ISA) or Machine Language (ML) for a CPU with size either 16-bit or 32-bit. The simulation should include a pipeline realization of the CPU based on the given ISA (the pipelined version should be able to detect pipeline data hazard (**Read after Write**) with a ***load*** operation). The ISA simulator must be presented before the end of the semester. The design requirements of the project are given below:

Design Requirements:

- The simulation should support at least 16-bit instructions.
- The ISA should consist of at least 8 general purpose registers (GPRs) along with control registers such as instruction pointer (IP)/ Program counter (PC), Instruction register (IR), etc.
- The ISA should include op-codes such as MOV (move/copy) or LOAD/STORE, ADD (addition), SUB (subtraction), MUL (multiplication), DIV (division), JMP (jump), CMP (compare) etc., at least.
- The instructions (either with 2-operand or 3-operand) must be based on the addressing modes such as *Reg-Reg*, or *Reg-Mem*, or *Immediate*, or all these together, or any other choice by the designer.
- Apply proper clock cycles for various instruction types and hence the simulator is able to calculate and display Clock cycles Per Instruction (CPI) and CPU time.
- The ISA simulator should display the contents of registers/memory locations in binary form during the processing of its instructions.
- The simulator should be able to simulate a pipelined (5-staged) execution of the user instructions (based on the ISA) and is able to detect and solve the RAW (read after write) data hazard.

A sample 24-bit ISA simulator (in 2-operand form):

A sample 24-bit ISA simulator is described here. Assume that the 24-bit ISA simulator has 16-bit data representation and 16-bit 8 general purpose registers (r0, r1, r2, r3, r4, r5, r6, and r7). The user can select any two register operands from the GPR list as a *multiplication_register* for multiplication and a *remainder_ register* for division. An example instruction set with their decoded and encoded forms are shown below (where **r0** is a multiplication_register and **r7** is a remainder_ register):

PC	Decoded form	Instruction meaning	Encoded form	Clock cycle(s)
[00]	mov r1 3	r1 ← 3	[000010010000000000000011]	1
[03]	add r1 3	r1 ← r1 + 3	[000110010000000000000011]	2
[06]	mov r2 r1	r2 ← r1	[000000100010000000000000]	1
[09]	mul r2 -1	r0: r2 ← r2 * -1	[001110101111111111111111]	4
[12]	mov r3 r2	r3 ← r2	[000000110100000000000000]	1
[15]	div r3 2	r7:r3 ← r3 / 2	[010010110000000000000010]	5
	end 0 0			

Values of registers after the execution of the instruction set

During the execution of the above code sequence, the values of registers would be varied in the following way:

r1 = 3 [0000 0000 0000 0011]
r1 = 6 [0000 0000 0000 0110]
r2 = 6 [0000 0000 0000 0110]
r0: r2 = -6 [1111 1111 1111 1111 1111 1111 1111 1010]
r3 = -3 [1111 1111 1111 1101] r7: 0 [0000 0000 0000 0000]

CPI (Clocks Per Instruction):

2.33 (the CPI of the instructions depends on the number of clock cycles used by each instruction).

Pipelined version:

At least a 5-stage Pipelined version of instruction set simulation is an acceptable one (but is an optional) But, anyone with such an additional feature will be rewarded greatly (no doubt!).

Scoring Criteria:

The following key quality will be considered (total 15 points):

- ISA design & coding 11%
- Presentation 2%
- Report with code 2%

(*If the source code of your simulator is not printed along with the report, then the design &coding section of the project is not valid*).