Computer Systems Engineering Department
Second Semester, 2012/2013
Computer Architecture - ENCS437
Project #2
Due Tuesday, May 5, 2013
Groups: At most two students can form a group.
Problem: Simulation of a Pipelined Processor

In this project, you will design and simulate a simple 16-bit RISC processor with seven 16-bit general-purpose registers: R1 through R7. R0 is hardwired to zero and cannot be written, so we are left with seven registers. There is also one special-purpose 16-bit register, which is the program counter (PC). All instructions are 16 bits. There are three instruction formats, R-type, I-type, and J-type as shown below:

Instruction Format

R-type format

4-bit opcode (Op), 3-bit register numbers (Rs, Rt, and Rd), and 3-bit function field (funct)

Op^4	Rs^3	Rt^3	Rd^3	funct^3

I-type format

4-bit opcode (Op), 3-bit register number (Rs and Rt), and 6-bit signed immediate constant

Op^4	Rs^3	Rt^3	Immediate^6

J-type format

4-bit opcode (Op) and 12-bit immediate constant

Op^4	Immediate^{12}

For **R-type** instructions, Rs and Rt specify the two source register numbers, and Rd specifies the destination register number. The function field can specify at most eight functions for a given opcode. We will reserve opcode 0 for R-type instructions. It is also possible to reserve more opcodes, if more R-type instructions exist.

For **I-type** instructions, Rs specifies a source register number, and Rt can be a second source or a destination register number. The immediate constant is only 6 bits because of the fixed-size nature of the instruction. The size of the immediate constant is suitable for our uses. The 6-bit immediate constant is signed (and sign-extended) for all I-type instructions.

For **J-type**, a 12-bit immediate constant is used for J (jump), JAL (jump-and-link), and LUI (load upper immediate) instructions.

Instruction Encoding

Seven R-type instructions, eight I-type instructions, and three J-type instructions are defined. These instructions, their meaning, and their encoding are shown below:

Instr	**Meaning**	**Encoding**				
AND	Reg(Rd) = Reg(Rs) & Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 000
ADD	Reg(Rd) = Reg(Rs) + Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 001
SUB	Reg(Rd) = Reg(Rs) – Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 010
SLT	Reg(Rd) = Reg(Rs) < Reg(Rt)	Op = 0000	Rs	Rt	Rd	f = 011
CAS	Reg(Rd) = Max[Reg(Rs) , Reg(Rt)]	Op = 0000	Rs	Rt	Rd	f = 100
CASL	Reg(Rd) =Mem {Max[Reg(Rs) , Reg(Rt)]}	Op = 0000	Rs	Rt	Rd	f = 101
JR	PC = Reg(Rs)	Op = 0000	Rs	000	000	f = 111
ANDI	Reg(Rt) = Reg(Rs) & Immediate^{6}	Op = 1000	Rs	Rt	Immediate^{6}	
ADDI	Reg(Rt) = Reg(Rs) + Immediate^{6}	Op = 0100	Rs	Rt	Immediate^{6}	
SLTI	Reg(Rt) = Reg(Rs) < Immediate^{6}	Op = 0101	Rs	Rt	Immediate^{6}	
LW	Reg(Rt) = Mem(Reg(Rs) + Imm^{6})	Op = 0110	Rs	Rt	Immediate^{6}	
LLB	Reg(Rt[0:7]) = Mem(Reg(Rs) + Imm^{6}) Reg(Rt[8:15]) = Zero	Op = 1110	Rs	Rt	Immediate^{6}	
SW	Mem(Reg(Rs) + Imm^{6}) = Reg(Rt)	Op = 0111	Rs	Rt	Immediate^{6}	
BEQ	Branch if (Reg(Rs) == Reg(Rt))	Op = 1010	Rs	Rt	Immediate^{6}	
BNE	Branch if (Reg(Rs) != Reg(Rt))	Op = 1011	Rs	Rt	Immediate^{6}	
J	PC = PC + Immediate^{12}	Op = 1100	Immediate^{12}			
JAL	R7 = PC + 2, PC = PC + Immediate^{12}	Op = 1101	Immediate^{12}			
LUI	R1 = Immediate^{12} << 4	Op = 1111	Immediate^{12}			

Opcode 0 is used for R-type instructions. The Load Upper Immediate (LUI) is of the J-type to have a 12-bit immediate constant loaded into the upper 12 bits of register R1. The LUI can be combined with ORI (or ADDI) to load any 16-bit constant into a register. Although the instruction set is reduced, it is still rich enough to write useful programs. We can have procedure calls and returns using the JAL and JR instructions.

Memory

Your processor will have separate instruction and data memories with 2^{16} words each. Each word is 16 bits or 2 bytes. **Instruction Memory is Word Addressable and Data Memory is Byte Addressable.**

Register File

Implement a Register file containing Seven 16-bit registers R1 to R7 with two read ports and one write port. R0 is hardwired to zero.

Arithmetic and Logical Unit (ALU)

Implement a 16-bit ALU to perform the specified operations.

Addressing Modes

PC-relative addressing mode is used for branch and jump instructions.
For branching (BEQ, BNE), the branch target address is computed as follows:
PC = PC + sign-extend(Imm6), by adding the contents of PC to sign-extended 6-bit Immediate.

For jumps (J and JAL): PC = PC + sign-extend(Imm12).

For LW and SW base-displacement addressing mode is used. The base address in register Rs is added to the sign-extended 6-bit immediate to compute the memory address.

For the exceptions: you must handle only one type of exception: unknown Op-code.

Testing

To test the implementation: write a simple programs to test all the instructions that you have implemented and then convert them into machine instructions by hand and load it into the instruction memory starting at address 0.

Additional Notes:

- GUI is required for testing.
- You must show the state of the registers, data memory, buffers and signals at each cycle
- You must show stall cycle if occurred
- **Grading Policy:** The grade will be assigned according to the following:
  - Single-cycle implementation (8/15 points)
  - Pipelined implementation same as **figure 4.66 page 387** (15/15 points)
  - Tomasuolo + reorder buffer + branch history table with two bit (20/15 points)
    assume the following:
    - each instruction type have one functional unit with two slots in the reservation station
    - add and sub takes 3 cycle, load and store takes 4 cycle, jumps and branch 1 cycle, and all other instructions takes 2 cycles
    - reorder buffer has 10 slots