Computer Organization and Structure

Homework #4
Due: 2012/12/11

1. The following piece of code has pipeline hazard(s) in it. Please try to reorder the instructions and insert the minimum number of `NOP` to make it hazard-free. (Note: assume all the necessary forwarding logics exist.)

```
haz:  move $5, $0
      lw $10, 1000($20)
      addiu $20, $20, -4
      addu $5, $5, $10
      bne $20, $0, haz
```

2. Assuming that the breakdown of dynamic instructions into various instruction categories is as follows:

	R-Type	**`beq`**	**`jmp`**	**`lw`**	**`sw`**
a.	50%	15%	10%	15%	10%
b.	30%	10%	5%	35%	20%

Also, assume the following branch predictor accurancies:

	Always-taken	**Always not-taken**	**2-bit**
a.	40%	60%	80%
b.	60%	40%	95%

   a. Stall cycles due to mispredicted branches increase the CPI. What is the extra CPI due to mispredicted branches with the always-taken predictor? Assume that branch outcomes are determined in the EX stage, that there are no data hazards, and that no delay slots are used.
   b. Repeat the above problem for the “always not-taken” predictor.
   c. Repeat the above problem for the “2-bit” predictor.
   d. With the 2-bit predictor, what speed-up would be achieved if we could convert half of the branch instructions in a way that replaces a branch instruction with an ALU instruction? Assume that correctly and incorrectly predicted instructions have the same chance of being replaced.
   e. With the 2-bit predictor, what speed-up would be achieved if we could convert half of the branch instructions in a way that replaced each branch instruction with two ALU instructions? Assume that correctly and incorrectly predicted instructions have the same chance of being replaced.
   f. Some branch instructions are much more predictable than others. If we know that 80% of all executed branch instructions are easy-to-predict loop-back branches that are always predicted correctly, what is the accuracy of the 2-bit predictor on the remaining 20% of the branch instructions?

3. Different instructions utilize different hardware blocks in the basic single-cycle implementation. The next three problems refer to the following instruction:

	Instruction	Interpretation
a.	`add Rd, Rs, Rt`	`Reg[Rd]=Reg[Rs]+Reg[Rt]`
b.	`lw Rt, Offs(Rs)`	`Reg[Rt]=Mem[Reg[Rs]+Offs]`

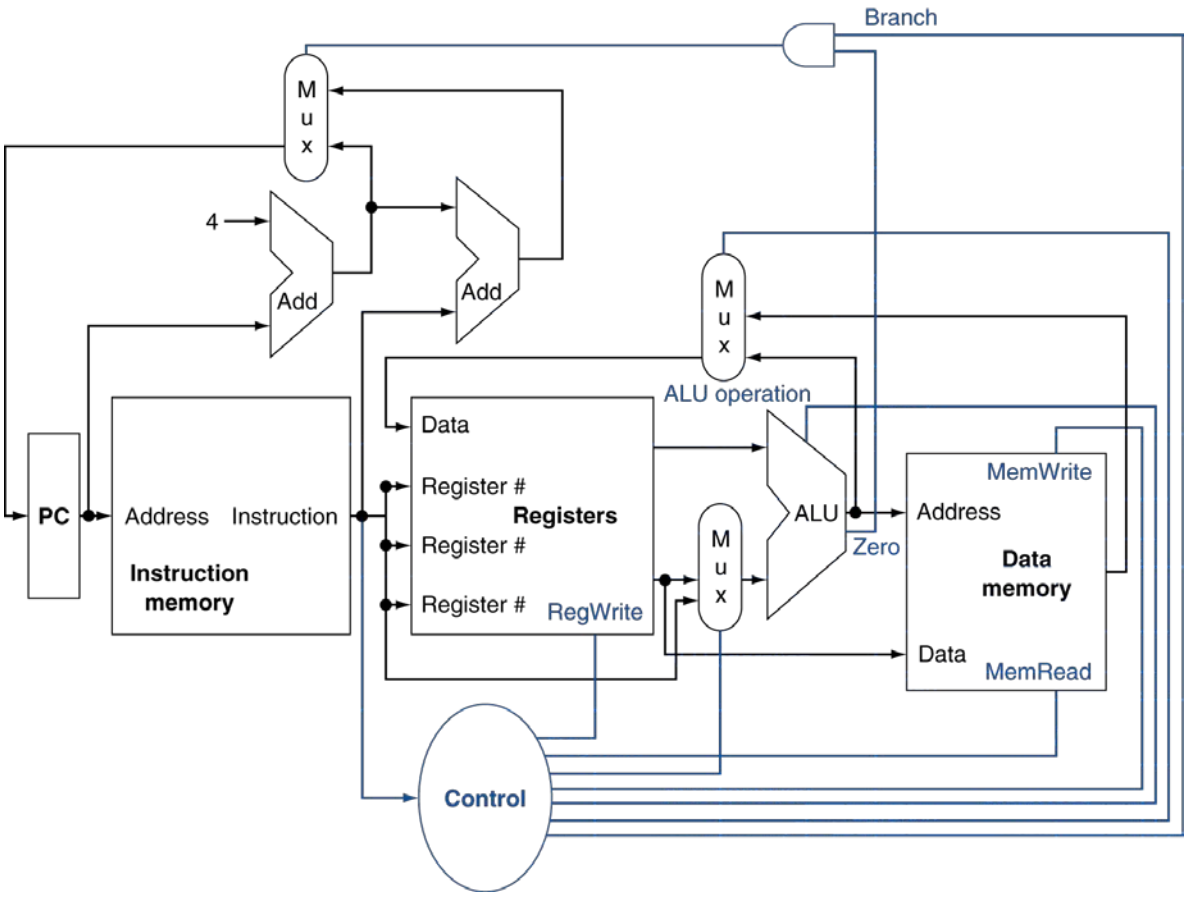

Figure 1: The basic implementation of the MIPS subset, including the necessary multiplexors and control lines.

a. What are the values of control signals generated by the control in Figure 1 for this instruction?
b. Which resources (blocks) perform a useful function for this instruction?
c. Which resources (blocks) produce outputs, but their outputs are not used for this instruction? Which resources produce no outputs for this instruction?

Different execution units and blocks of digital logic have different latencies (time needed to do their work). In Figure 1 there are seven kinds of major blocks. Latencies of blocks along the critical (longest-latency) path for an instruction determine the minimum latency of that instruction. For the following three problems, assume the following resource latencies:

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Control
a.	400ps	100ps	30ps	120ps	200ps	350ps	100ps
b.	500ps	150ps	100ps	180ps	220ps	1000ps	65ps

d. What is the critical path for a MIPS `AND` instruction?
e. What is the critical path for a MIPS load (`LD`) instruction?
f. What is the critical path for a MIPS `BEQ` instruction?

The basic single-cycle MIPS implementation in Figure 1 can only implement some instructions. New instructions can be added to an existing ISA, but the decision whether or not to do that depends, among other things, on the cost and complexity such an addition introduces into the processor datapath and control. The next three problems refer to this new instruction:

	Instruction	Interpretation
a.	add3 Rd, Rs, Rt,Rx	Reg[Rd]=Reg[Rs]+Reg[Rt]+Reg[Rx]
b.	sll Rt, Rd, Shift	Reg[Rd]= Reg[Rt] << Shift (shift left by Shift bits)

g. Which existing blocks (if any) can be used for this instruction?
h. Which new functional blocks (if any) do we need for this instruction?
i. What new signals do we need (if any) from the control unit to support this instruction?

4. What is the average CPI for each of the following 2 schemes taking to execute the code sequence below? (Note: For the pipeline scheme, there are five stages: IF, ID, EX, MEM, and WB. We assume the reads and writes of register file can occur in the same clock cycle, and the stall circuits are available.)

```
add $t3, $s1, $s2
sub $t1, $s1, $s2
lw $t2, 100($t3)
sub $s1, $t1, $t2
```

a. single cycle scheme
b. pipelined scheme with data forwarding hardware (one from EX/MEM to ALU input, and the other from MEM/WB to ALU input) available