Computer Organization and Structure

Homework #4
Due: 2010/12/21

1. 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]`

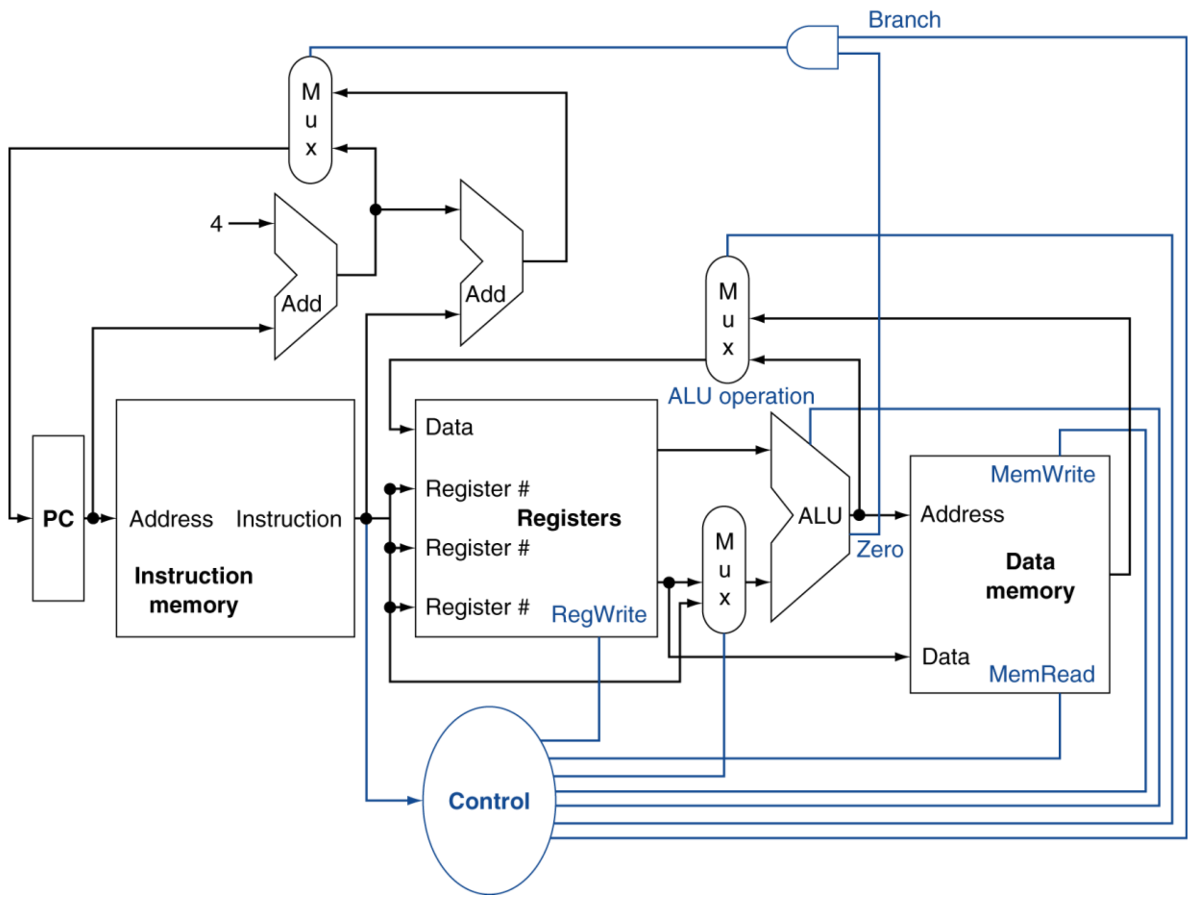

Figure 1: The basic implementation of the MIPS subset, including the necessary multiplexors and control lines.

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?

2. Assuming that logic blocks needed to implement a processor's datapath have the following latencies:

	I-Mem	**Add**	**Mux**	**ALU**	**Regs**	**D-Mem**	**Sign-extend**	**Shift-left-2**
a.	400ps	100ps	30ps	120ps	200ps	350ps	20ps	2ps
b.	500ps	150ps	100ps	180ps	220ps	1000ps	90ps	20ps

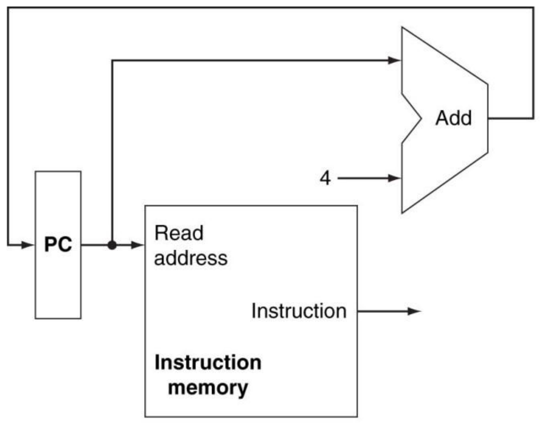

Figure 2: A portion of the datapath used for fetching instructions and incrementing the program counter. The fetched instruction is used by other parts of the datapath.

a. If the only thing we need to do in a processor is fetch consecutive instructions (Figure 2), what would the cycle time be?
b. Consider a datapath similar to the one in Figure 3, but for a processor that only has one type of instruction: unconditional PC-relative branch. What would the cycle time be for this datapath?
c. Repeat the above problem, but this time we need to support only *conditional* PC-relative branches.

The remaining three problems refer to the following logic block (resource) in the datapath:

	Resource
a.	Add 4 (to the PC)
b.	Data Memory

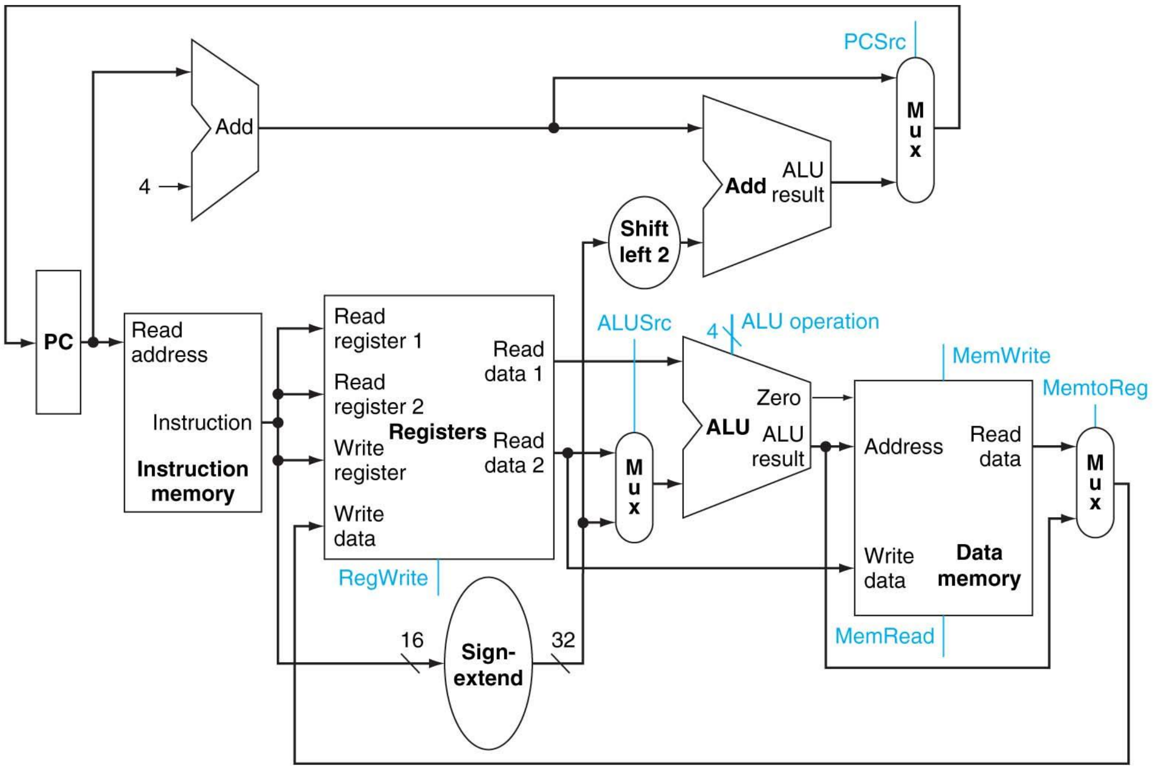

Figure 3: The simple datapath for the MIPS architecture combines the elements required by different instruction classes.This datapath can execute the basic instructions (load-store word, ALU operations, and branches) in a single clock cycle.

d. Which kinds of instructions require this resource?
e. For which kinds of instructions (if any) is this resource on the critical path?
f. Assuming that we only support `beq` and `add` instructions, discuss how changes in the given latency of this resource affect the cycle time of the processor. Assume that the latencies of other resources do not change.

3. The following problems refer to the following sequence of instructions:

	Instruction sequence

a.	`lw $1, 40($6)`
	`add $6, $2, $2`
	`sw $6, 50($1)`

b.	`lw $5, -16($5)`
	`sw $5, -16($5)`
	`add $5, $5, $5`

a. Indicate dependences and their type.
b. Assume there is no forwarding in this pipelined processor. Indicate hazards and add `nop` instructions to eliminate them.
c. Assume there is full forwarding. Indicate hazards and add `nop` instructions to eliminate them.

The remaining problems assume the following clock cycle times:

	Without forwarding	**With full forwarding**	**With ALU-ALU forwarding only**
a.	300ps	400ps	360ps
b.	200ps	250ps	220ps

d. What is the total execution time of this instruction sequence without forwarding and with full forwarding? What is the speed-up achieved by adding full forwarding to a pipeline that had no forwarding?
e. Add `nop` instructions to this code to eliminate hazards if there is ALU-ALU forwarding only (no forwarding from the MEM to the EX stage)?
f. What is the total execution time of this instruction sequence with only ALU-ALU forwarding? What is the speed-up over a no-forwarding pipeline?

4. 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?