Homework 5

Computer Architecture
CSCI 361, Fall 2014

Due: November 24, 2014 in class

Instructions: Complete the problems enumerated below. No collaboration is permitted on this or any assignment. Correct answers without work shown will not receive full credit. Answers must typed and neatly formatted. You will submit a printed hardcopy of your work at the beginning class on the specified due date. Include your name and email address on each page of your submission. Late assignments will not be accepted.

Problem 1 **[15 pts]**

Consider the following instruction:

```
AND Rd,Rs,Rt          # Reg[Rd] = Reg[Rs] AND Reg[Rt]
```

(a) What are the values of control signals generated by the control in Figure 4.2 for the following 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?

Problem 2 **[15 pts]**

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

I-Mem	**Add**	**Mux**	**ALU**	**Regs**	**D-Mem**	**Sign-Extend**	**Sign-Left-2**
200ps	70ps	20ps	90ps	90ps	250ps	15ps	10ps

(a) If the only thing we need to do in a processor is fetch consecutive instructions (Figure 4.6), what would the cycle time be?

(b) Consider a datapath similar to the one in Figure 4.11, 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) Consider a datapath similar to the one in Figure 4.11, but for a processor that only has one type of instruction: *conditional* PC-relative branches. What would the cycle time be for this datapath?

Problem 3 **[20 pts]**

In this problem, you will examine in detail how an instruction is executed in a single-cycle datapath. This problem refers to a clock cycle in which the processor fetches the following instruction word:

1010 1100 0110 0010 0000 0000 0001 0100

Assume that data memory is all zeros and that the processors registers have the following values at the beginning of the cycle in which the above instruction word is fetched:

r0	**r1**	**r2**	**r3**	**r4**	**r5**	**r6**	**r8**	**r12**	**r31**
0	-1	2	-3	-4	10	6	8	2	-16

(a) What are the outputs of the sign-extend and the jump "Shift left 2" unit (near the top of Figure 4.24) for this instruction word?

(b) What are the values of the ALU control unit's inputs for this instruction?

(c) What is the new PC address after this instruction is executed? Describe the path (list the resources) through which this value is determined.

(d) For the ALU and the two add units, what are their data input values?

Problem 4 **[15 pts]**

In this problem, you will determine how pipelining affects the clock cycle time of the processor. This problem assumes that individual stages of the datapath have the following latencies:

IF	**ID**	**EX**	**MEM**	**WB**
250ps	350ps	150ps	300ps	200ps

(a) What is the clock cycle time in a pipelined and non-pipelined (single-cycle) processor?

(b) What is the total latency of an LW instruction in a pipelined and non-pipelined (single-cycle) processor?

(c) If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

Problem 5 **[15 pts]**

In this problem, assume the following sequence of instructions:

```
I1:  or r1, r2, r3
I2:  or r2, r1, r4
I3:  or r1, r1, r2
```

(a) Indicate all dependencies and their types:

- read after write (RAW)
- write after read (WAR)
- write after write (WAW)

(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.

Problem 6 **[10 pts]**

Consider the following loop.

```
loop:  lw  r1, 0(r1)
       and r1, r1, r2
       lw  r1, 0(r1)
       lw  r1, 0(r1)
       beq r1, r0, loop
```

Assume that perfect branch prediction is used (no stalls due to control hazards), that there are no delay slots, and that the pipeline has full forwarding support. Also assume that many iterations of this loop are executed before the loop exits.

Show a pipeline execution diagram for the third iteration of this loop, from the cycle in which we fetch the first instruction of that iteration up to (but not including) the cycle in which we can fetch the first instruction of the next iteration. Show all instructions that are in the pipeline during these cycles (not just those from the third iteration).

Problem 7 [10 pts]

This exercise is intended to help you understand the relationship between delay slots, control hazards, and branch execution in a pipelined processor. In this exercise, assume that the following MIPS code is executed on a pipelined processor with a 5-stage pipeline, full forwarding, and a predict-taken branch predictor:

```
        lw r2, 0(r1)
label1: beq r2, r0, label2      # not taken once, then taken
        lw r3, 0(r2)
        beq r3, r0, label1      # taken
        add r1, r3, r1
label2: sw r1, 0(r2)
```

Draw the pipeline execution diagram for this code, assuming there are no delay slots and that branches execute in the EX stage.

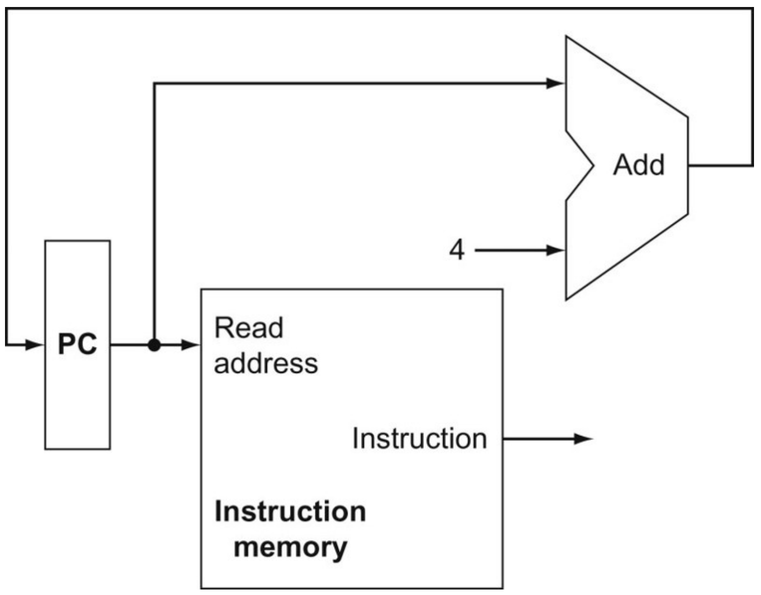

FIGURE 4.6: A portion of the datapath used for fetching instructions and incrementing the program counter. The fetched instruction is used by other parts of the datapath.

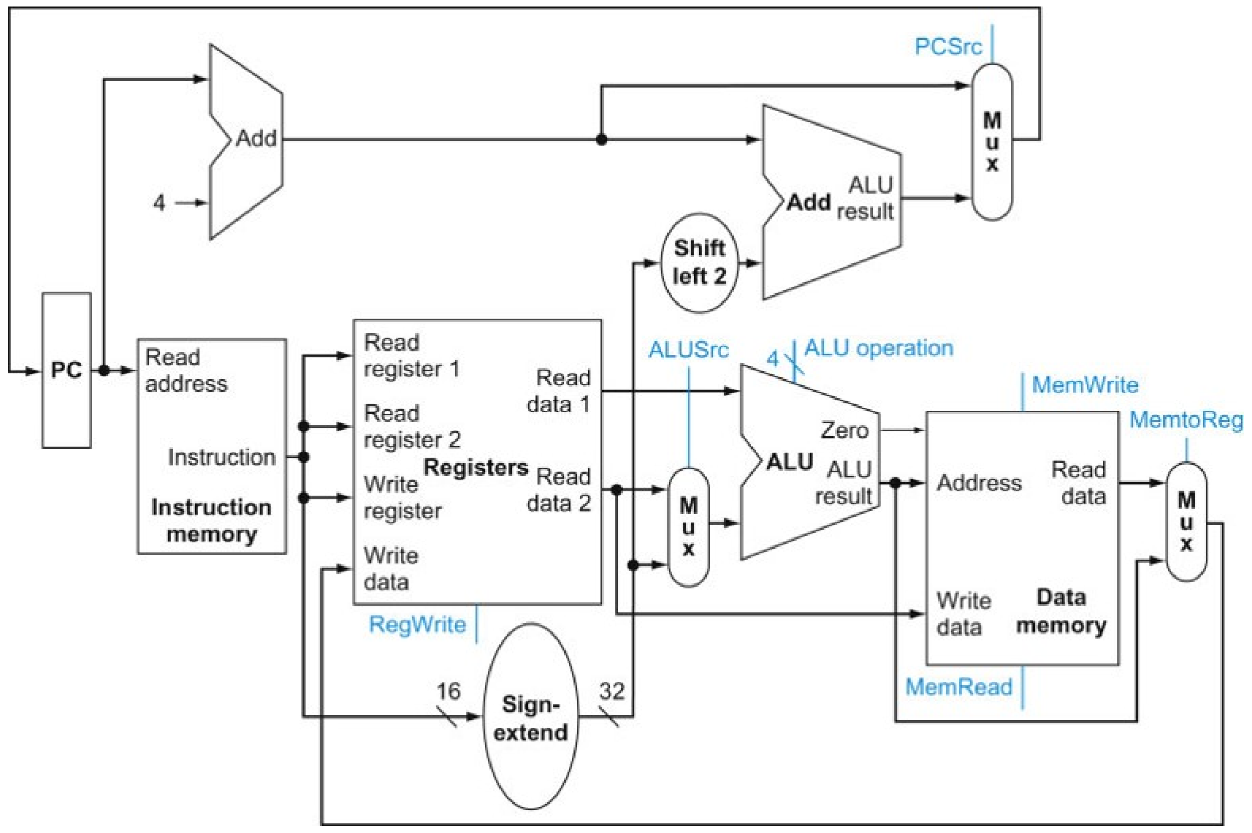

FIGURE 4.2: The basic implementation of the MIPS subset, including the necessary multiplexors and control lines. The top multiplexor ("Mux") controls what value replaces the PC (PC + 4 or the branch destination address); the multiplexor is controlled by the gate that "ANDs" together the Zero output of the ALU and a control signal that indicates that the instruction is a branch. The middle multiplexor, whose output returns to the register file, is used to steer the output of the ALU (in the case of an arithmetic-logical instruction) or the output of the data memory (in the case of a load) for writing into the register file. Finally, the bottommost multiplexor is used to determine whether the second ALU input is from the registers (for an arithmetic-logical instruction or a branch) or from the offset field of the instruction (for a load or store). The added control lines are straightforward and determine the operation performed at the ALU, whether the data memory should read or write, and whether the registers should perform a write operation. The control lines are shown in color to make them easier to see.

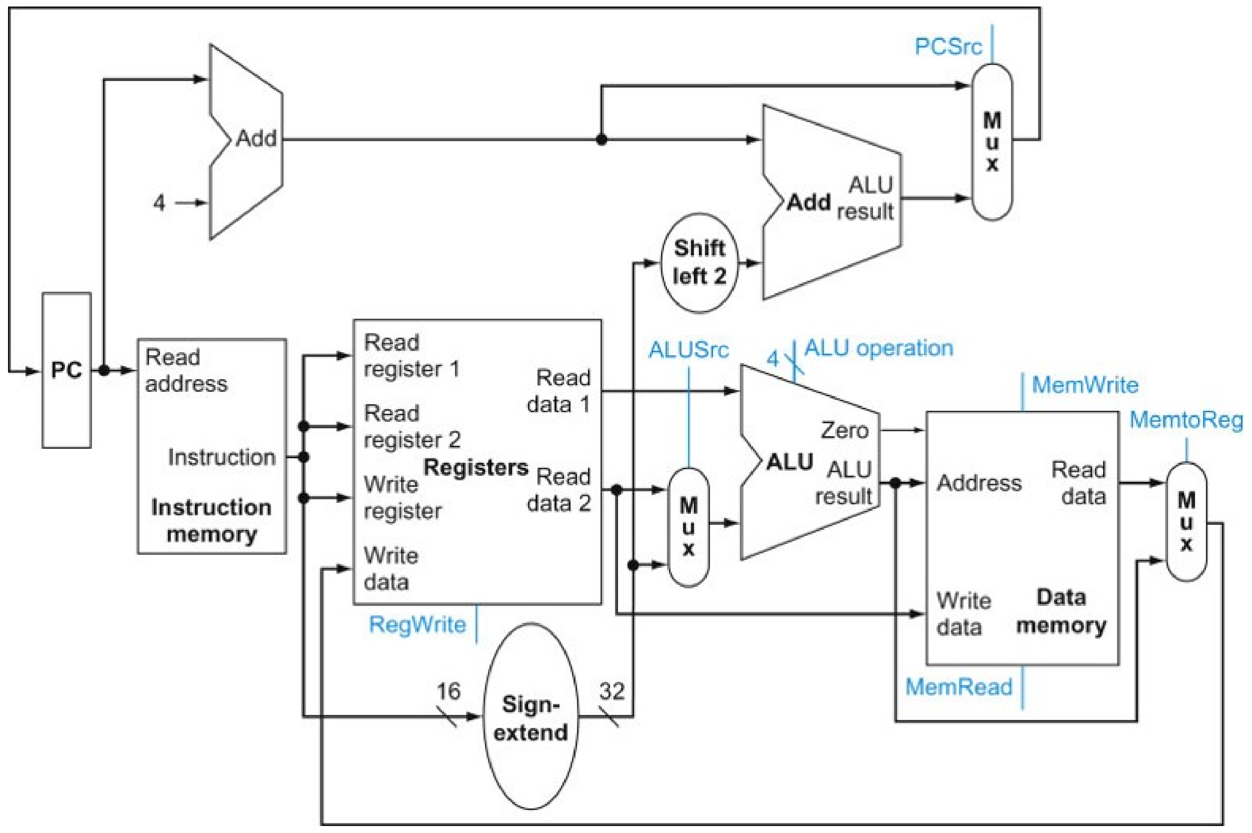

FIGURE 4.11: The simple datapath for the core MIPS architecture combines the elements required by different instruction classes. The components come from Figures 4.6, 4.9, and 4.10. This datapath can execute the basic instructions (load-store word, ALU operations, and branches) in a single clock cycle. Just one additional multiplexor is needed to integrate branches. The support for jumps will be added later.

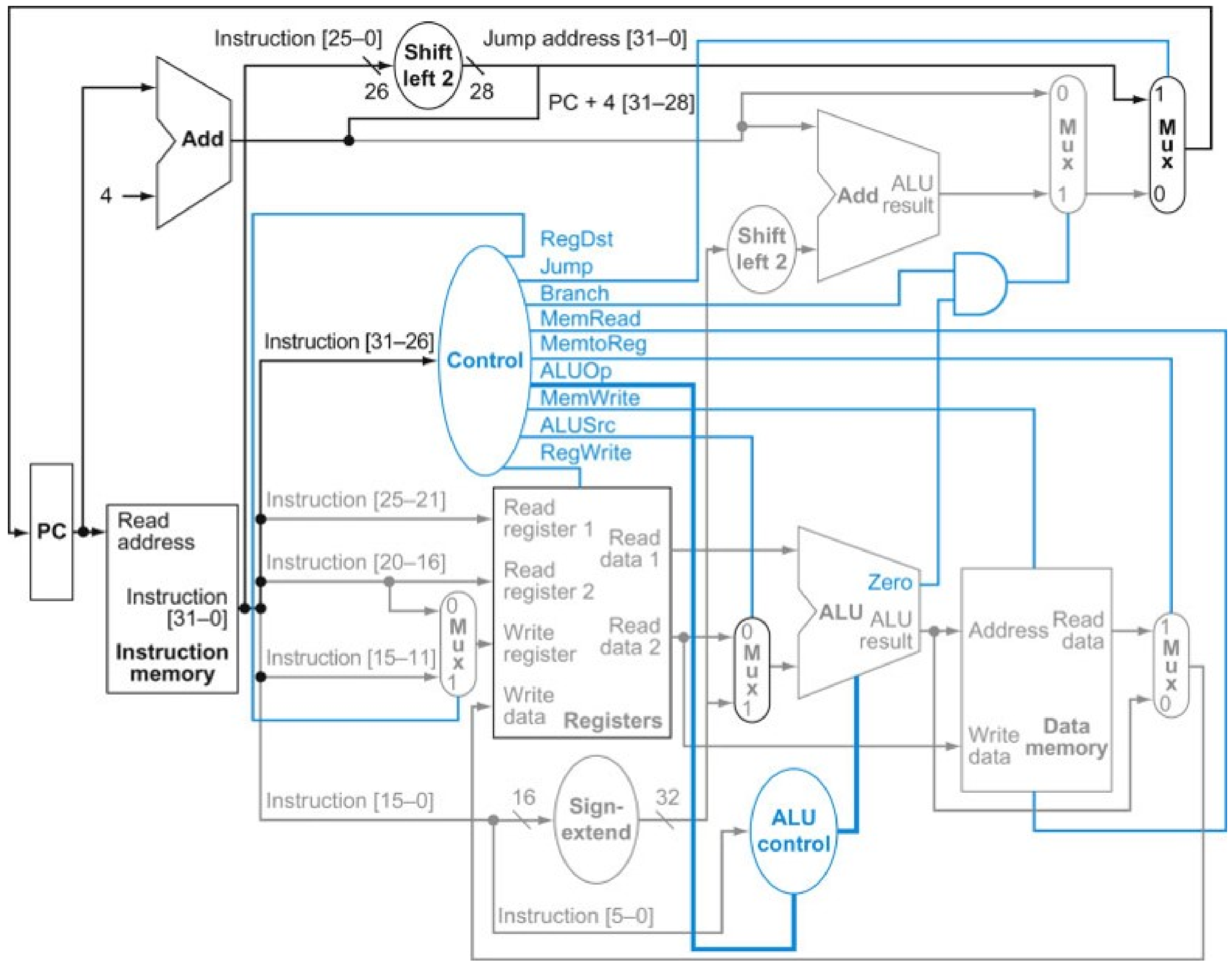

FIGURE 4.24: The simple control and datapath are extended to handle the jump instruction. An additional multiplexor (at the upper right) is used to choose between the jump target and either the branch target or the sequential instruction following this one. This multiplexor is controlled by the jump control signal. The jump target address is obtained by shifting the lower 26 bits of the jump instruction left 2 bits, effectively adding 00 as the low-order bits, and then concatenating the upper 4 bits of PC + 4 as the high-order bits, thus yielding a 32-bit address.