

CSSE3001/7000 2006 – Tutorial 3

1. Explain the reasons why the register windows used by the SPARC architecture are a good idea. Explain why they are, in practise, not as useful as they appear.

2. What are some differences between embedded processors and DSPs? What are some similarities?

5.2 [10] <§5.4> Describe the effect that a single stuck-at 0 fault (i.e., regardless of what it should be, the signal is always 0) would have for the signals shown below, in the single-cycle datapath in Figure 5.17 on page 307 (given at the end of this tutorial). Which instructions, if any, will not work correctly? Explain why.

Consider each of the following faults separately:

- a. RegWrite = 0
- b. ALUop0 = 0
- c. ALUop1 = 0
- d. Branch = 0
- e. MemRead = 0
- f. MemWrite = 0

5.3 [5] <§5.4> This exercise is similar to Exercise 5.2, but this time consider stuck-at-1 faults (the signal is always 1).

5.8 [15] <§5.4> We wish to add the instruction `jr` (jump register) to the single-cycle datapath described in chapter 5. Add any necessary datapaths and control signals to the single-cycle datapath of Figure 5.17 on page 307 and show the necessary additions to Figure 5.18 on page 308. It is recommended you print out copies of the diagram for easy modification.

5.9 [10] <§5.4> This question is similar to Exercise 5.8 except that we wish to add the instruction `sll` (shift left logical).

5.15 [5] <§5.4> Describe the effect that a single stuck-at-0 fault (i.e., regardless of what it should be, the signal is always 0) would have on the multiplexors in the single-cycle datapath in Figure 5.17 on page 307. Which instructions, if any, would still work? Consider each of the following faults separately: RegDst = 0, ALUSrc = 0, MemtoReg = 0, Zero = 0.

5.29 [5] <§5.5> This exercise is similar to Exercise 5.2, but this time consider the effect that the stuck-at-0 faults would have on the multiple-cycle datapath in Figure 5.27. Consider each of the following faults:

- a. RegWrite = 0
- b. MemRead = 0
- c. MemWrite = 0
- d. IRWrite = 0
- e. PCWrite = 0
- f. PCWriteCond = 0.

5.30 [5] <§5.5> This exercise is similar to Exercise 5.29, but this time consider stuck-at-1 faults (the signal is always 1).

5.32 [15] <§5.5> We wish to add the instruction lui (load upper immediate) to the multicycle datapath. Use the same structure of the multicycle datapath of Figure 5.28 on page 323 (also given at the end of this tutorial) and show the necessary modifications to the finite state machine of Figure 5.38 on page 339. You may find it helpful to examine the execution steps shown on pages 325 through 329 and consider the steps that will need to be performed to execute the new instruction. How many cycles are required to implement this instruction?

5.33 [15] <§5.5> You are asked to modify the implementation of lui in Exercise 5.32 in order to cut the execution time by 1 cycle. Add any necessary datapaths and control signals to the multicycle datapath of Figure 5.28 on page 323. You have to maintain the assumption that you don't know what the instruction is before the end of stat 1 (end of second cycle). Please explicitly state how many cycles it takes to execute the new instruction on your modified datapath and finite state machine.

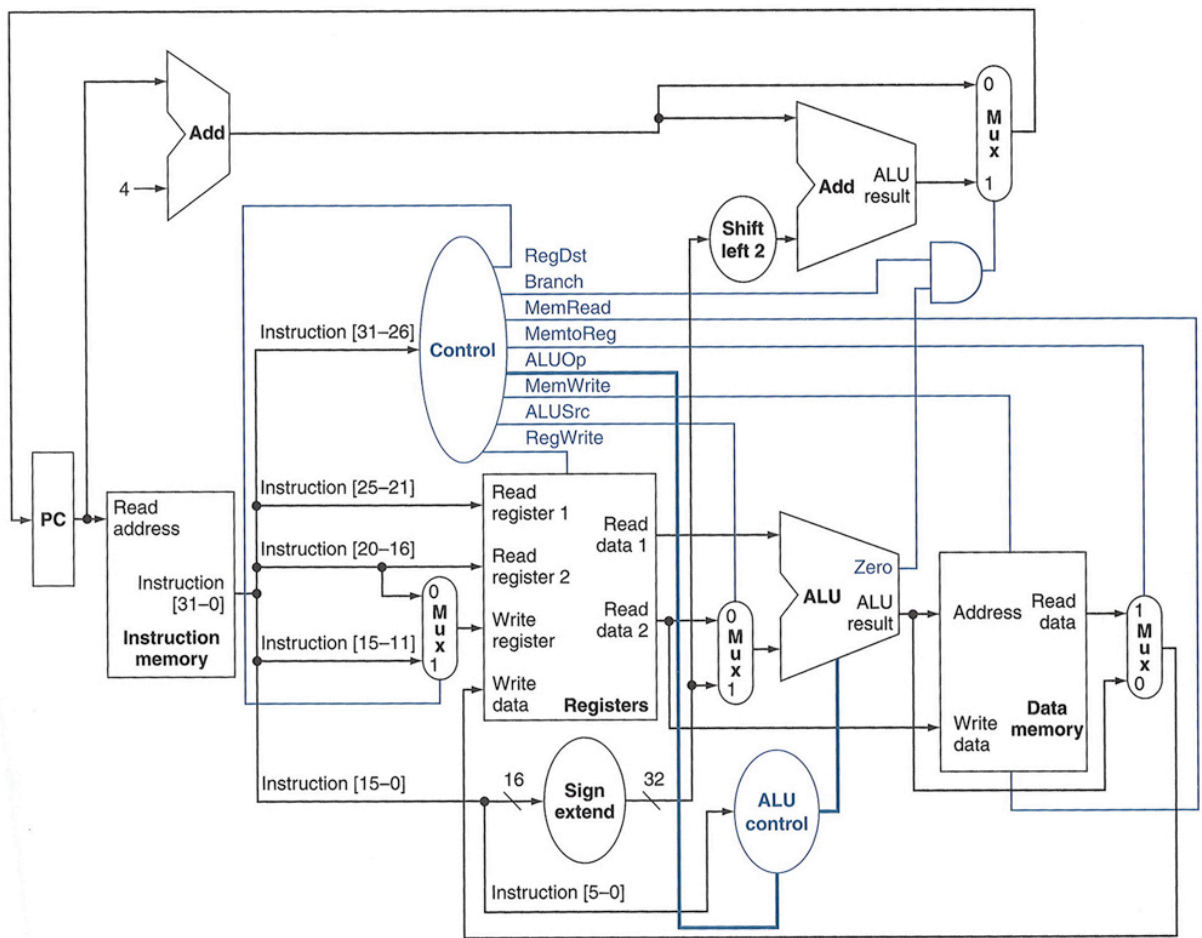


FIGURE 5.17 The simple datapath with the control unit. The input to the control unit is the 6-bit opcode field from the instruction. The outputs of the control unit consist of three 1-bit signals that are used to control multiplexors (RegDst, ALUSrc, and MemtoReg), three signals for controlling reads and writes in the register file and data memory (RegWrite, MemRead, and MemWrite), a 1-bit signal used in determining whether to possibly branch (Branch), and a 2-bit control signal for the ALU (ALUOp). An AND gate is used to combine the branch control signal and the Zero output from the ALU; the AND gate output controls the selection of the next PC. Notice that PCSrc is now a derived signal, rather than one coming directly from the control unit. Thus we drop the signal name in subsequent figures.