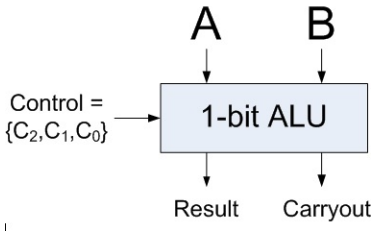
CS320 Spring 2014 Practice Midterm

Problem 1: Implement a 1-bit ALU, with inputs and outputs as shown. The ALU must perform the operations shown in the table according to the specified control signals. Use the minimal number and smallest sized AND, OR, NOT and MUX gates to implement the unit.



C_2	C_1	C_0	Operation
0	0	0	A AND B
0	0	1	A + B (carryout)
0	1	0	A OR B
0	1	1	A XOR B
1	0	0	A NOR B
1	0	1	NOT A
1	1	0	A NAND B
1	1	1	NOT B

Problem 2: Using the 1-bit ALU from the previous problem, we want to build a 4-bit ALU. What additional inputs/outputs are required? Assume these additional inputs and outputs are added to your 1-bit ALU. Draw/Implement the 4-bit ALU out of 1-bit ALUs. Assuming each logic gate (AND, OR, NOT, MUX) takes 1ns to operate, how much time does each operation of the ALU take?

Problem 3: Implement the Single Cycle Datapath ALU Control Unit (Figure 4.13 in textbook) using AND, OR and NOT gates.

Problem 4: Implement the Single Cycle Datapath Control Logic Unit (Figure 4.22 in textbook) using a Decoder and OR gates.

Problem 5: For the following instructions:

sw $4, -100 ($16)

add $1, $2, $3

beq $5, $7, -20

a. What is the binary value of the instruction word? Use the table in the textbook to determine opcodes.

b. What is the binary values at the input of the Register Read 1 & 2 Register File inputs? Are they actually read for this instruction? Is the Output used?

c. Which Register numbers are given to the write register input of the Register File? Is the register written? Why or why not?

Problem 6: For the following instructions:

10101100011000100000000000010100

00000000100000100000100000101010

a. What are the outputs of the sign-extend and the shift left 2 (Figure 4.24) for each instruction word?

b. What are the values of the ALU control unit's inputs for each instruction word?

c. What is the new PC address after each instruction is executed? Where did this value come from, which path through the datapath?

d. Write the instruction in assembly language.

Problem 7: Your friend is proposing that the control signal MemtoReg be eliminated from the single-cycle datapath (Figure 4.17). The multiplexor that has MemtoReg as an input will instead use the control signal ALUSrc or MemtoRead.

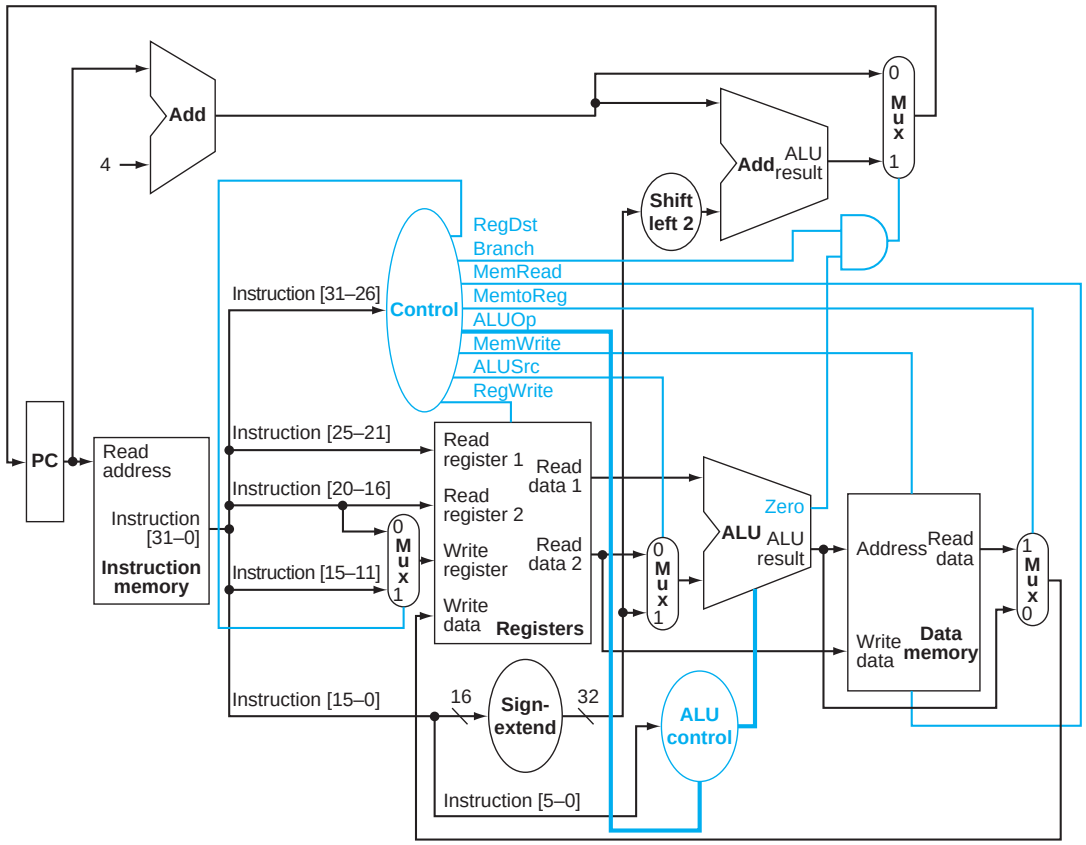


FIGURE4.17 The simple datapath with the control unit. The input to the control unit is the 6-bit opcode field from the instruction. The outputs of the control unit consist of three 1-bit signals that are used to control multiplexors (RegDst, ALUSrc, and MemtoReg), three signals for controlling reads 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.

a. Will this modification work? Can one of the two signals (MemRead and ALUSrc) substitute for the other? Explain.

b. Furthermore, determine whether any of the control signals in the single-cycle implementation can be eliminated and replaced by another existing control signal, or its inverse. Note that such redundancy is there because we have a very small set of instructions at this point, and it will disappear (or be harder to find) when we implement a larger number of instructions.

Problem 8: When estimating the performance of the single cycle datapath (Figure 4.17) we assumed that only the major functional units had any delay (ie, mux, control unit, PC access, sign extension unit, all other combinatorial logic and wires had negligible delay). Assume the following delays, with a different delay of ALU and adders for simple addition.

- Instruction/Data Memory Read and Write: 200ps
- Register Files Access: 50ps
- ALU: 70ps
- adder for PC + 4: X ps
- adder for branch address computation: Y ps

a. What would the cycle time be if X = 30 and Y = 30?

b. What would the cycle time be if X = 50 and Y = 50?

c. What would the cycle time be if X = 10 and Y = 80?

Problem 9: Consider the following timing values for the single cycle datapath (Figure 4.24).

I-Mem	Adder	Mux	ALU	Reg File	D-Mem	Sign Ext	Shift Left 2	ALU Cntl
200ps	70ps	20ps	90ps	90ps	250ps	15ps	10ps	30ps

a. What is the critical path timing for the ADD, LW and BEQ instructions based on the above timings? Assume the Control Unit has negligible timing.

b. Assume you are designing the control unit and must know the timing requirements for the unit. How much time can the control unit take to generate the MemWrite Signal (critical path from input:opcode to output of signal: MemWrite) without impacting the overall datapath timing?

c. In addition to above, now assume that the Control Unit requires the following amount of time to generate the control signals What is the clock cycle time of the processor?

RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	Jump
500ps	100ps	450ps	500ps	200ps	500ps	450ps	200ps	500ps

Problem 10: In class we covered the MIPS single-cycle implementation which handled only a subset of the MIPS instructions: R-type (add, sub, and, or, slt), memory references (lw, sw), conditional branch (beq) and jump (j). In this problem we will add to the implementation functionality for additional MIPS instructions. Use the datapath in (Figure 4.24) and control table below to specify your changes, add a row for each instruction and any additional columns (new control signals) to the control table as necessary. Be sure to mark don't cares in the control table whenever possible.

Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUp0	Jump
R	1	0	0	1	0	0	0	1	0	0
lw	0	1	1	1	1	0	0	0	0	0
sw	X	1	X	0	0	1	0	0	0	0
beq	X	0	X	0	0	0	1	0	1	0
j	X	X	X	0	X	0	X	X	X	1

Problem Guidelines:

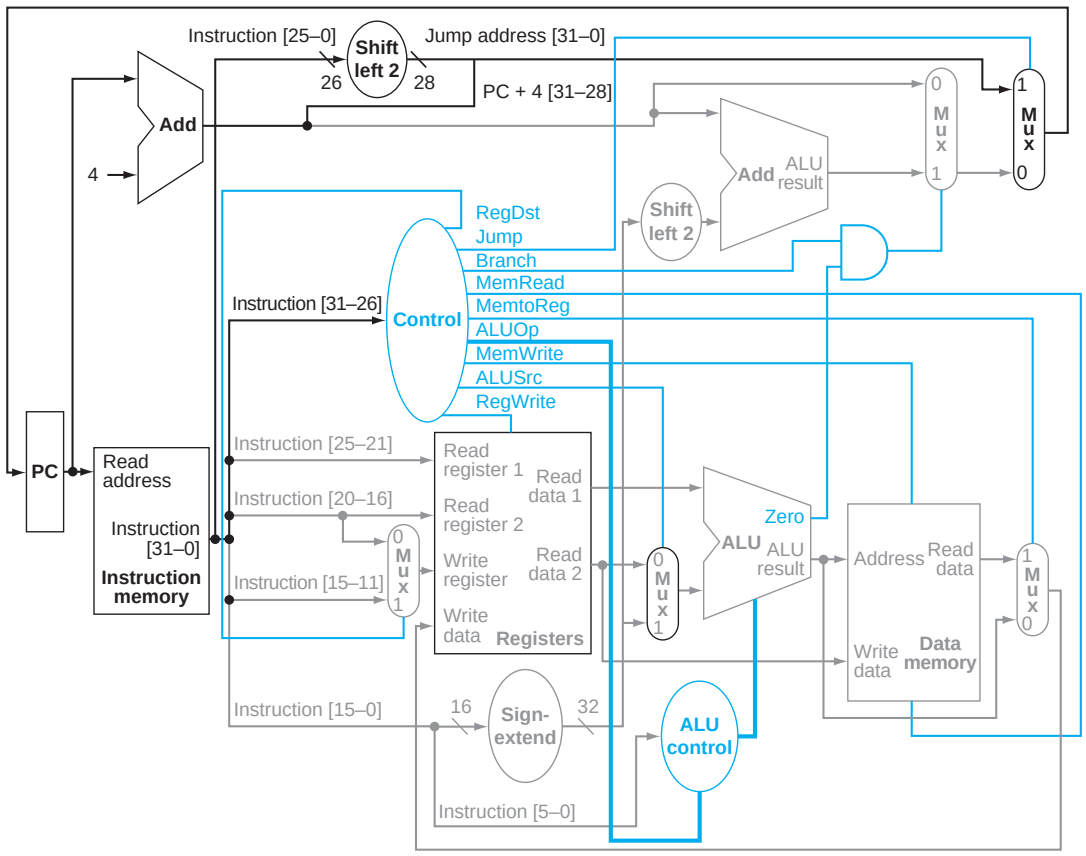


FIGURE 4.24 The simple control and datapath are extended to handle the jump instruction. An additional multiplexor (at the upper right) is used to choose between the jump target and either the branch target or the sequential instruction following this one. This multiplexor is controlled by the jump control 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. Copyright © 2009 Elsevier, Inc. All rights reserved.

- When adding new instructions, don't break the operation of the standard ones.
- Avoid adding ALUs, adders, Reg Files, or memories to the datapath
- You can add MUXes, logic gates, etc. but try to do minimally. (these cost in terms of area, cycle time, etc)

a. Modify the datapath and the control table to implement 'jal' instruction.
Reg[31] ← PC+4 # $ra register stores the return address
PC ← PC+4[31:28], (Instruction[25:0] << 2)
PC is the top 4 bits of the PC+4 value concatenated with the lowest 26 bits of the jump address shifted to the left by 2 bits

b. Modify the datapath and the control table to implement new 'lwr' instruction.
lwr Rd, Rs, Rt # Reg[Rt] = Mem[Reg[Rd] + Reg[Rs]]

c. Modify the datapath and the control table to implement new 'seq' instruction.
seq Rd, Rs, Rt # Reg[Rd] = 1 if Reg[Rs] == Reg[Rt], otherwise Reg[Rd] = 0

d. Calculate the delay in the modified datapath when performing the new instructions. Assume the delays given in the previous problem. Disregard the Control Unit delays.