King
CS 331
31 October 2020

Homework Assignment 4

Objectives

The purpose of this assignment is to:

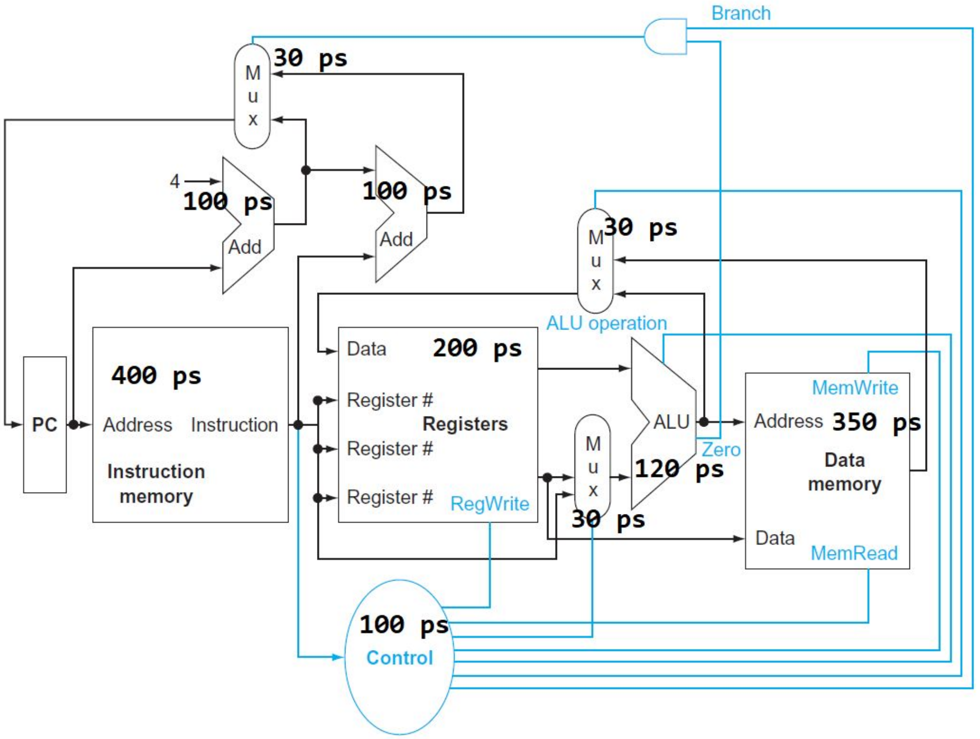
- Understand the key principles for building a single cycle datapath that implements a subset of MIPS instructions,
- Understand how to design the corresponding control to ensure the correct flow of data for different instructions,
- Understand how adding/modifying hardware blocks can affect processor performance.

Guidelines

All question numbers refer to exercises at the end of chapter 4 of the textbook (Computer organization and design: the Hardware/Software interface, 5th edition). Solutions for the following problems are to be done by you and only you.

Questions

Exercise 4.3.1 and 4.3.2



When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are starting with a datapath from Figure 4.2, where I-Mem, Add, Mux, ALU, Regs, D-Mem, and Control blocks have latencies of 400 ps, 100 ps, 30 ps, 120 ps, 200 ps, 350 ps, and 100 ps, respectively, and costs of 1000, 30, 10, 100, 200, 2000, and 500, respectively.

Consider the addition of a multiplier to the ALU. This addition will add 300 ps to the latency of the ALU and will add a cost of 600 to the ALU. The result will be 5% fewer instructions executed since we will no longer need to emulate the MUL instruction.

1. What is the clock cycle time with and without this improvement?
2. What is the speedup achieved by adding this improvement?

Given:
Latencies I-Mem = 400 ps, Add = 100 ps, Mux = 30 ps, ALU = 120 ps, Regs = 200 ps, D-mem = 350 ps, Control = 100 ps.

Costs I-Mem = 1000, Add = 30, Mux = 10, ALU = 100, Regs = 200, D-mem = 2000, Control = 500.

Addition of a multiplier to the ALU, +300 ps ALU latency, +600 ALU cost, results in 5% fewer instructions.

1. Find longest delay/critical path using latencies:
   PC > IM > Add_2 > Mux_1 > PC > ...
   (400 + 100 + 30) ps = 530 ps

   PC > IM > Reg > Mux_3 > ALU > DM > Mux_2 > Regs > ...
   (400 + 200 + 30 + 120 + 350 + 30 + 200) ps = 1330 ps => critical path

   PC > IM > Control > Mux_3 > ALU > DM > Mux_2 > Regs > ...
   (400 + 100 + 30 + 120 + 350 + 30 + 200) ps = 1230 ps

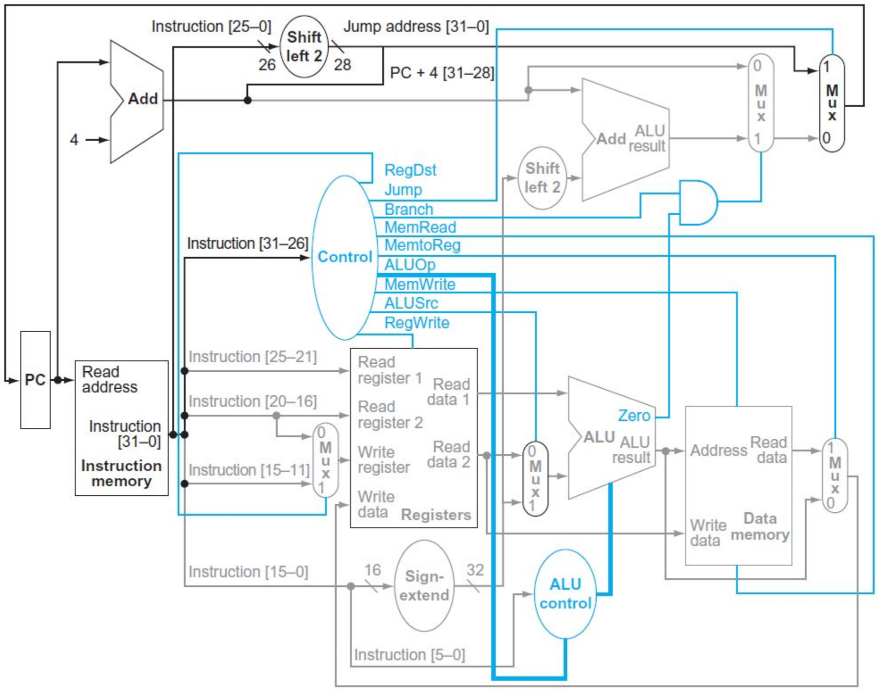
   The clock cycle time without improvement is 1330 ps.

   With addition of a multiplier to the ALU, the clock cycle time is increased by 300 ps in the ALU. The ALU is passed once, so the new clock cycle time is 1630 ps.

2. For speedup, consider a 5% decrease in IC with the new clock time and find the old clock time’s multiplier.
   1630*(0.95) = 1330*(n)
   n = 1.16
   (1/(1.16))*(100%) = 86.2%

   Therefore, the new clock time is 13.8% slower than the old clock time.

Exercise 4.7



In this exercise we examine in detail how an instruction is executed in a single-cycle datapath. Problems in this exercise refer 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 processor’s 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

1. 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?
2. What are the values of the ALU control unit’s inputs for this instruction?
3. What is the new PC address after this instruction is executed? Highlight the path through which this value is determined.
4. For each Mux, show the values of its data output during the execution of this instruction and these register values.
5. For the ALU and the two add units, what are their data input values?

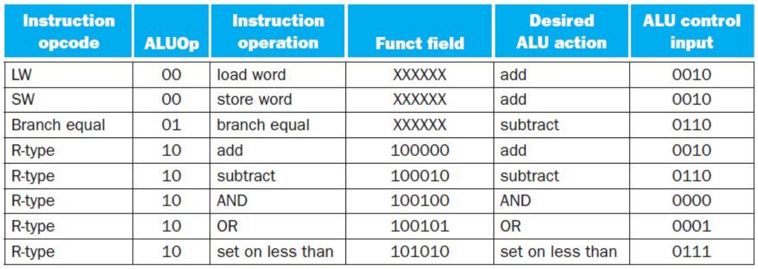
6. What are the values of all inputs for the “Registers” unit?

Given:
Instruction word 1010 1100 0110 0010 0000 0000 0001 0100

101011 00011 00010 0000 0000 0001 0100
opcode = (101011)_2 = sw (I-format)
rs = (00011)_2 = 3
rt = (00010)_2 = 2
imm = (0000000000010100)_2 = 20

1. (0000 0000 0001 0110)_2
   Sign extend => (0000 0000 0000 0000 0000 0000 0001 0100)_2
   Shift left 2 => (0000 0000 0000 0000 0000 0000 0101 0000)_2 = 80

2. Referencing the truth table for the ALU control bits:



Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

   For sw, ALUOp = 00, and its ALU control input is 0010.

3. To move to the next instruction, PC is incremented by 4 through the path PC > Add unit on top left > Mux with its control input received from the Control’s Jump flag > PC.

4. Top left = PC+4, Top right = PC+4
   Lower left = ?, Lower center = ?, Lower right = ?

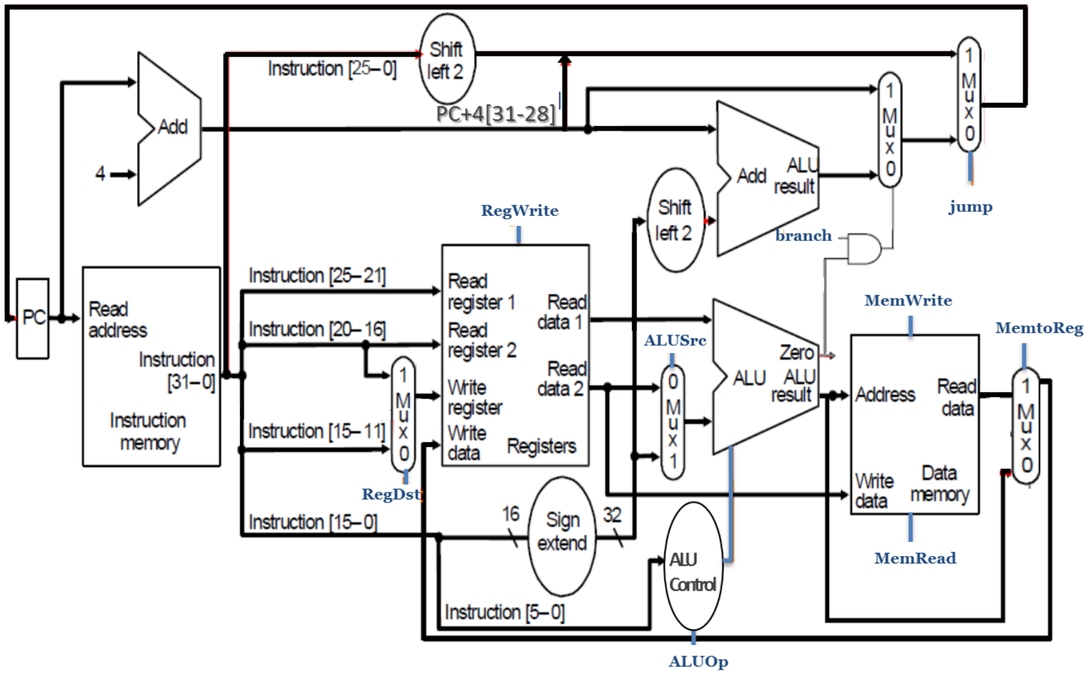
5. ?

Extra Exercise

Consider the basic single-cycle MIPS datapath that implements a subset of the MIPS instruction set (add, subtract, and, or, lw, sw, beq, jump). We would like to modify this datapath to implement the jal instruction:

Instruction: jal address

Interpretation: Reg[31] = PC + 4
                 PC = {PC+4[31:28], address, 2’b0}



Which existing blocks will we use? Do we need to add new functional units or modify existing ones? Do we need extra control signals?

Specify the value of each control signal (RegDst, ALUSrc, ALUOp, MemRead, MemWrite, MemToReg, RegWrite, Branch) as well as the value of any additional signal if needed.

Given:
Instruction: jal address

Interpretation: Reg[31] = PC + 4
PC = {PC+4[31:28], address, 2’b0}

Existing blocks:
The Add unit is used to increment the PC value.
Registers are used to receive two operands, but writing to the register file is not necessary.
?

New blocks:
?

RegDst = x, ALUSrc = x
ALUOp = 1, MemRead = 1
MemWrite = 1, MemToReg = 1
RegWrite = x, Branch = 1
?