Computer Architecture and Programming Languages
Jacobs University Bremen
Fjolla Dedaj

Course: CO20-320241
November 25, 2019

Homework 10

Problem 10.1

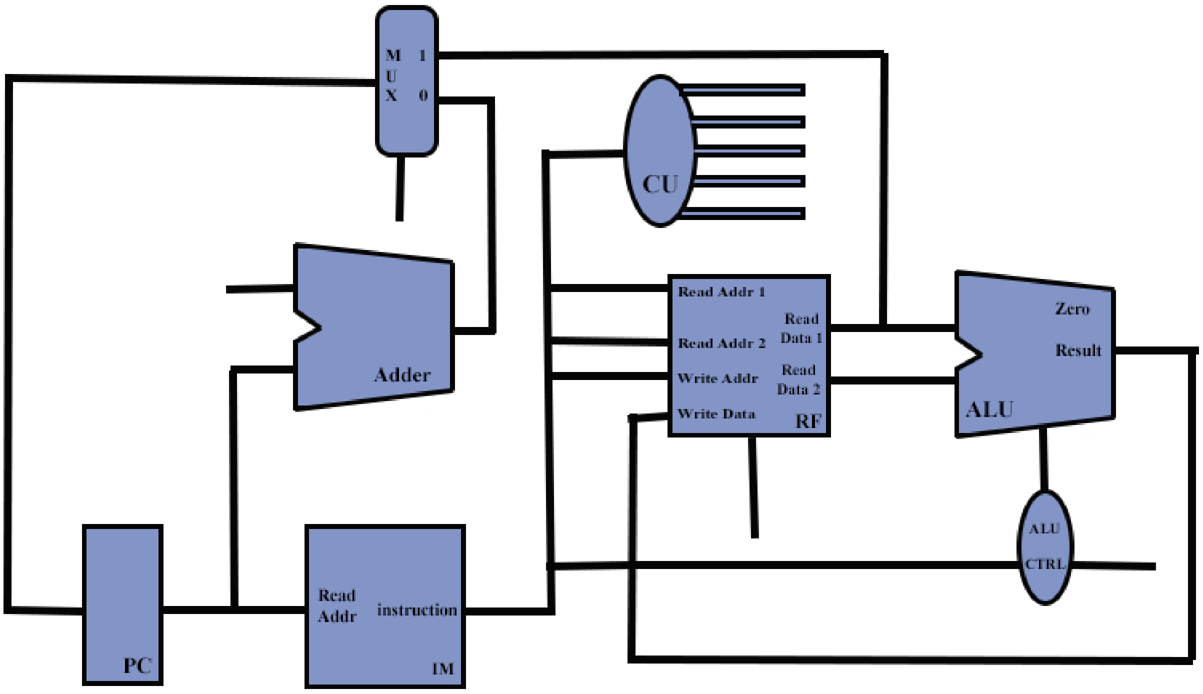
Why are contol lines necessary in a single cycle datapath? How exactly do they interact with the multiplexors?

Solution:
The control lines are necessary in a single-cycle datapath because the multiplexers use them to determine which input lines are needed to be included in the instruction. This depends on the opcode that the instruction has. The control lines from the control unit also activate or deactivate memory and registrar manipulations (read from memory/write to memory, read from registrar/write to registrar).
Control lines interact with multiplexers by deciding which input from the multiplexers will be used as an output.

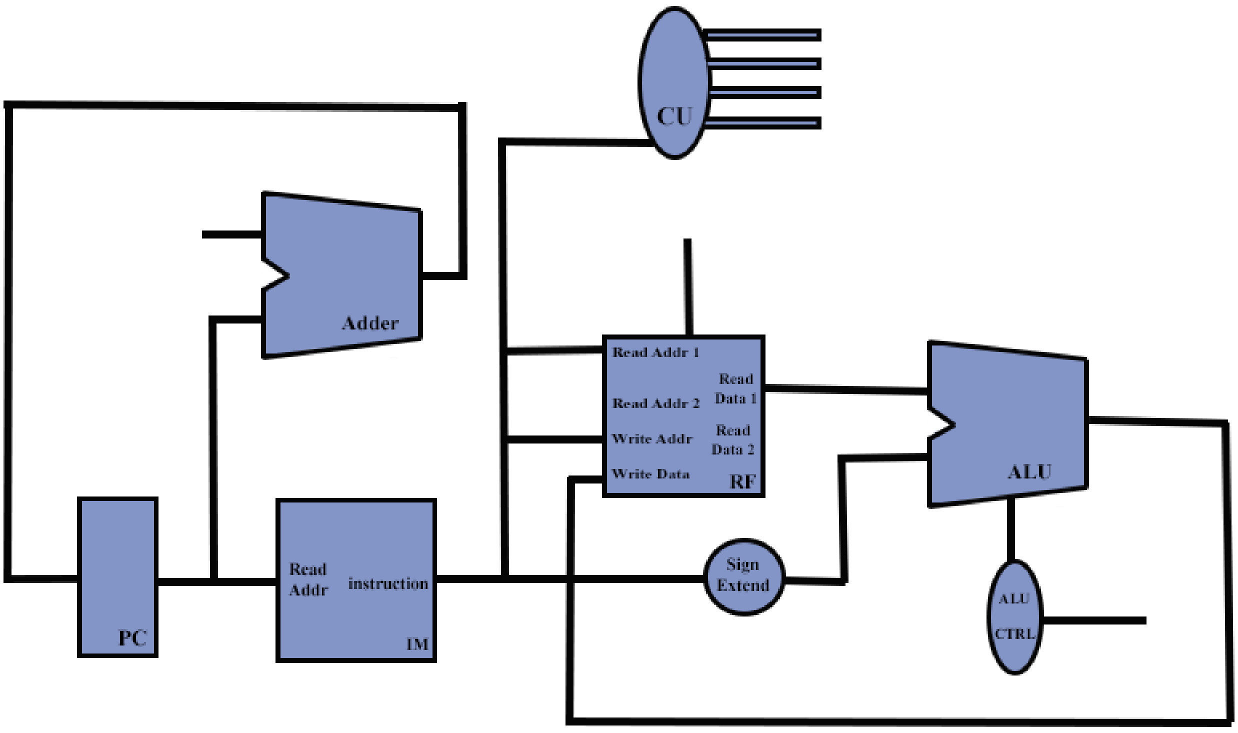
Problem 10.2

Solution:

add:



addi:



When *add* operation is being executed, first the instruction is read from memory with the help of *PC*, therefore the *PC* adder is shown. Since the data is read from two source registers, *RegDst* is activated. *WriteReg* is also signaled, since we are writing to a register. *ALUSrc* is deactivated at this time. *ALUSrc* determines the datapath so that both values from the source registers are put into the ALU where the operation is executed and then written to the destination register.

When *addi* operation is being executed, first the instruction is read from memory with the help of *PC*, therefore the *PC* adder is shown. *RegDst* is deactivated. The binary value flows through the sign-extend component and into the multiplexer where the *ALUSrc* being activated this time, leads the bits into the ALU along with the value from one of the source registers. Then, The operation is executed in the ALU and it is written back in the register.

Problem 10.3

I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-left-2
450ps	110ps	30ps	120ps	250ps	350ps	20ps	0ps

a) What is the clock cycle time, if the only type of instruction we need to support are the ALU instructions *(add, and, etc.)*

Solution:
For instructions like *add, and, etc.* the following logic blocks are activated:
I-Mem, Regs (RegF), Mux2, ALU, Mux3, Regs (RegF).
Thus, the clock cycle time will be: $450 + 250 + 2 \times 30 + 120 + 3 \times 30 + 250 =$ **1220ps**

b) What is the clock cycle time, if we only have to support *sw* instructions?

Solution:
The execution of *sw* would follow the following steps in your diagram:

1. Instruction is read and decoded from the PC in the Instruction Memory subcircuit.
2. The register file is read for $rs and $rt (Registers subcircuit)
3. The value of $rs is added to the sign extended immediate (selected by ALUSrc) (ALU subcircuit).
4. The added value and $rt are passed to the Data Memory subcircuit where the value of $rt is written to memory.

Thus, the following logic blocks are activated: **I-Mem, Regs, ALU and D-Mem**
The clock cycle time will be: $450 + 250 + 120 + 350 = \mathbf{1170ps}$

c) What is the clock cycle time, if we have to support add, beq, lw, and sw instructions?

Solution:
Whenever we have *sw, lw, add, beq* instructions, *lw* is the longest possible time-consuming instruction path, the following logic blocks are activated:
I-Mem, Regs (RegF), ALU, D-Mem, Mux3, Regs (RegF).
The clock cycle time will be: $450 + 250 + 120 + 350 + 3 \times 30 + 250 = \mathbf{1510ps}$