Computer Architecture and Programming Languages — Course: CO20-320241
Jacobs University Bremen — 18 November, 2019
Romelda Blaceri

Homework 9

Problem 9.1

Solution:

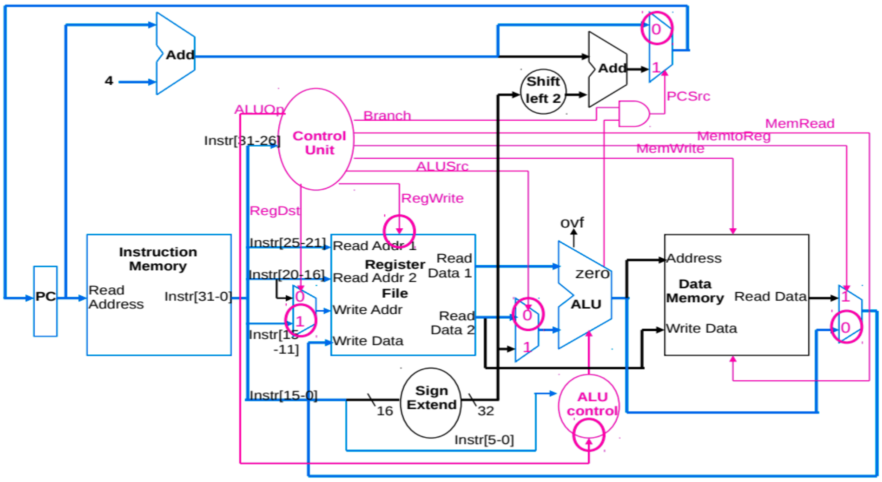
a) An explicit write signal is not needed in a single-cycle datapath, as after every instruction, the PC will always be updated.

b) Different from a single-cycle datapath instruction which takes only one cycle, in a multicycle datapath, each of the instructions can take more then one cycle, so since the PC needs to get updated after every current instruction, a write signal is needed.

Problem 9.2

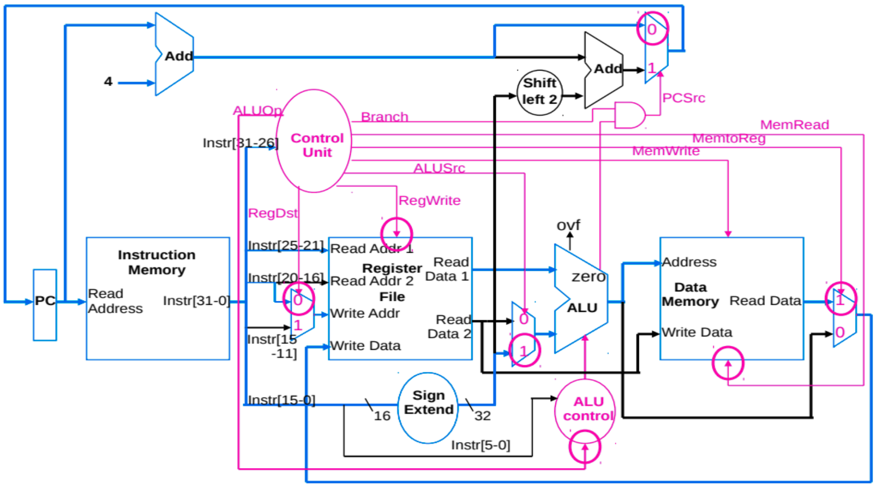
Solution:

a) Considering the MIPS assembler instruction `add $s0,$s1,$s2` → the single cycle datapath will be:



The operation is `add`, therefore we have an R-type instruction, which means that Branch, MemRead, and MemWrite will be zero, and ALUOp will be 10. The remaining part of the values of the control lines are as follows: RegDst is 1 since for `add` there is a destination register (`$s0` in our case), RegWrite is 1 as in the previously mentioned destination register the result will be written, MemtoReg is 0 as the sum will be written by ALU immediately in the destination register.

For instruction `lw $s3, 16($s2)` → the single cycle datapath will be:



The operation is `lw`, therefore we have an I-type instruction, which means that only rs and rt are used, RegDst and Branch are 0, ALUOp is 00, and RegWrite is 1 as the value will be loaded in register rt. With `lw` we are loading a value from the memory, so MemtoReg is 1. Since we're basically reading information from the specified memory cell, MemRead will be 1 and MemWrite will be 0. Constructing the table with values of the control lines for the two instructions, we get:

Instruction	**RegDst**	**ALUSrc**	**MemtoReg**	**RegWrite**	**MemRead**	**MemWrite**	**Branch**	**ALUOp**
`add`	1	0	0	1	0	0	0	10
`lw`	0	1	1	1	1	0	0	0

b) Some cases when the ALU needs to add its inputs are when using `lw` and `add` instructions (it is also mentioned in the lecture slides that these operations cause the ALU action to be addition). Specifically, when using `lw` and `add`, as mentioned in point a, ALUOp is 00 and 10, respectively, but the ALU action is 0010 for both (therefore it's addition).