Computer Architecture and Programming Languages
Jacobs University Bremen
Romelda Blaceri

Course: CO20-320241
25 November, 2019

Homework 10

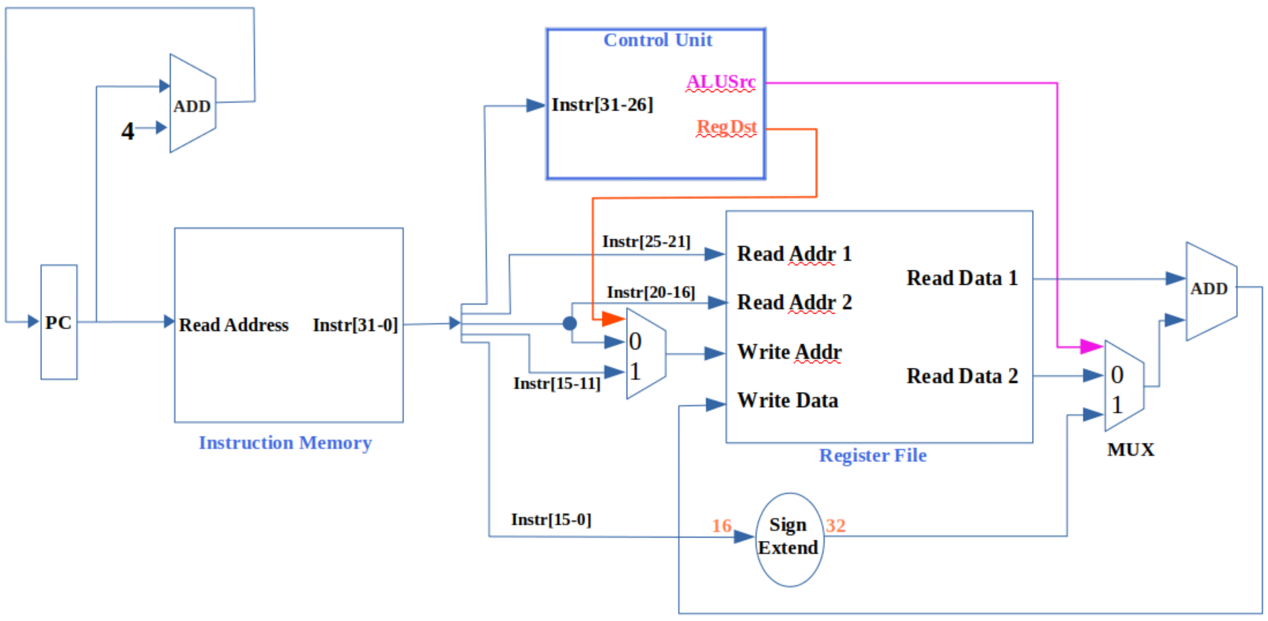
Problem 10.1

Solution:
Control lines are necessary in a single cycle datapath because they control all the paths that need to be followed for each given instruction, specifically, they control whether the result derived from a process will be stored into a register, whether we will read from the memory or write in it, and in this case go to the register where the data will be written, specify the inputs for arithmetic and logic instructions, etc. They're also neccesary for multiplexors, as they determine their output. So, if the control line has the value zero, for a two-input multiplexor, the first input signal will get selected and if the value is one, the second input signal will be selected to be the output.

Problem 10.2

Solution:
First of all, since both istructions are basically addition, we don't need to use a full ALU but just a 32-bit parallel adder. Since there is no branching, we increase PC by 4 when moving to the next location. The destination registers for `add` and `addi` are `rd` and `rt` respectively, so a two-input multiplexor can be used to choose between the two. Anyway, the first input to the main adder will always be `rs`, then, depending on the instruction, the second one can be either register `rt` (for `add`), or the constant value (for `addi`), where we use a multiplexor again. For both these instructions we only use registers, so we neither read nor write in the memory (no control lines for memory).

The datapath is drawn below:



Explanation: The program counter gets updated during the negative going transition of the clock cycle. Then, there's generated the instruction, where we get the `op` code expressed by the first 6 bits, which will give the information about the instruction that will be used (`add` - 000000 or `addi` - 001000). Accordingly, we will then have: for `add`, both RegDst and ALUSrc are 0 and for `addi` they're both 1. In the end, when the sum is calculated and written in the destination register, we reach the end of the clock cycle and the program counter is updated.

Problem 10.3

Solution:

a) The 4 possible paths an ALU instruction can take are:
1. I-Mem → RegF → Mux2 → ALU → Mux3 → WBack to RegF
2. I-Mem → RegF → ALU → Mux3 → WBack to RegF
3. Add → Mux4 → PCWrite
4. I-Mem → Mux1 → RegF

The first 2 paths illustrate how different tasks are executed in order to make the operation possible, the third one updates the PC (that's basically just adding 4 to the PC, so the time for PCWrite will be same as time for operation `add`), and the last one specifies the destination register.
We check the final values for each path in the table below:

Path	**I-Mem** 450ps	**Add** 110ps	**Mux** 30ps	**ALU** 120ps	**Regs** 250ps	**D-Mem** 350ps	**Sign-Extend** 20ps	**Shift- left-2** 0ps	**Total time**
1.	450	-	$2 \cdot 30$	120	$2 \cdot 250$	-	-	-	1130
2.	450	-	30	120	$2 \cdot 250$	-	-	-	1100
3.	-	$2 \cdot 110$	30	-	-	-	-	-	250
4.	450	-	30	-	250	-	-	-	730

As we can see from the table, the longest path (in the concept of time) is the first one (I-Mem → RegF → Mux2 → ALU → Mux3 → WBack to RegF). This path defines the final clock cycle time, which is 1130 ps.

b) The 4 possible paths a `sw` instruction can take are:
1. I-Mem → Sign-Extend → Mux2 → ALU → D-Mem
2. I-Mem → RegF → ALU → D-Mem
3. I-Mem → RegF → D-Mem
4. Add → Mux4 → PCWrite

Same explanation as for part a, except that the third task, except from the destination register, specifies also the memory that will be occupied.
We check the final values for each path in the table below:

Path	**I-Mem** 450ps	**Add** 110ps	**Mux** 30ps	**ALU** 120ps	**Regs** 250ps	**D-Mem** 350ps	**Sign-Extend** 20ps	**Shift- left-2** 0ps	**Total time**
1.	450	-	30	120	-	350	20	-	970
2.	450	-	-	120	250	350	-	-	1170
3.	450	-	-	-	250	350	-	-	1050
4.	-	$2 \cdot 110$	30	-	-	-	-	-	250

The longest path is the second one (I-Mem → RegF → ALU → D-Mem). This path defines the final clock cycle time, which is 1170 ps.

c) The 4 possible paths a `lw` instruction can take are:
1. I-Mem → Sign-Extend →Mux2→ ALU → D-Mem →Mux3→ WBack to RegF
2. I-Mem → RegF → ALU → D-Mem → Mux3 → WBack to RegF
3. Add → Mux4 → PCWrite
4. I-Mem → Mux1 → RegF

Path	**I-Mem** 450ps	**Add** 110ps	**Mux** 30ps	**ALU** 120ps	**Regs** 250ps	**D-Mem** 350ps	**Sign-Extend** 20ps	**Shift- left-2** 0ps	**Total time**
1.	450	-	$2 \cdot 30$	120	250	350	20	-	1250
2.	450	-	30	120	$2 \cdot 250$	350	-	-	1450
3.	-	$2 \cdot 110$	30	-	-	-	-	-	250
4.	450	-	30	-	250	-	-	-	730

The longest path is the second one (I-Mem → RegF → ALU → D-Mem → Mux3 → WBack to RegF), therefore the clock cycle time = 1450 ps.

Meanwhile, the 4 possible paths a `beq` instruction can take are:
1. I-Mem $\rightarrow$ Sign-Extend $\rightarrow$ Shift-left-2 $\rightarrow$ Add $\rightarrow$ Mux4 $\rightarrow$ PCWrite
2. I-Mem $\rightarrow$ RegF $\rightarrow$ Mux2 $\rightarrow$ ALU $\rightarrow$ Mux4 $\rightarrow$ PCWrite
3. I-Mem $\rightarrow$ RegF $\rightarrow$ ALU $\rightarrow$ Mux4 $\rightarrow$ PCWrite
4. Add $\rightarrow$ Mux4 $\rightarrow$ PCWrite

Path	**I-Mem** 450ps	**Add** 110ps	**Mux** 30ps	**ALU** 120ps	**Regs** 250ps	**D-Mem** 350ps	**Sign-Extend** 20ps	**Shift- left-2** 0ps	**Total time**
1.	450	$2 \cdot 110$	30	-	-	-	20	0	720
2.	450	110	$2 \cdot 30$	120	250	-	-	-	990
3.	450	110	30	120	250	-	-	-	960
4.	-	$2 \cdot 110$	30	-	-	-	-	-	250

The longest path is the second one (clock cycle time = 990 ps).

Now, we consider the longest time for each instruction:

Instruction	**Time**
`add`	1130 ps
`sw`	1170 ps
`lw`	1450 ps
`beq`	990 ps

The longest time possible determines the final clock cycle, which in our case is 1450 ps.
(instruction `lw`, path: I-Mem $\rightarrow$ RegF $\rightarrow$ ALU $\rightarrow$ D-Mem $\rightarrow$ Mux3 $\rightarrow$ WBack to RegF)

For the paths above, considering the latencies that were the same as the ones mentioned in the given table, it is clear why their time is counted. The only latencies that seem different are: RegF and WBack to RegF, which will be counted as Regs, since they're basically registers, and PCWrite, which is considered as addition, for the reasons already mentioned in part a. In the tables, the cells that are filled with ' - ' are the ones that are not included in the path, while the others are the latencies of the path, multiplied with the number of times they're mentioned, and added in the end to derive the total time taken.