Solutions to Homework 4

4.1

In this problem, refer to Figure 4.17 (or similar).

4.1.1

RegWrite: 0 then 1
ALUSrc: 0
ALUOp: 0000
MemWrite: 0
MemRead: 0
MemtoReg 0

Note that ALUOp here refers to the output of the ALU control. Unfortunately, the text refers to those four bits and the two bits coming out of the main control both as "ALUOp."

4.1.2

Registers, ALU, MUXes

4.1.3 No output (or at least nothing useful): Data Memory
Not used: Sign-extend, Shift left 2, Branching MUX

4.5

In this problem, refer to Figure 4.17 (or similar). Remember that Figure 4.12 has an incorrect table heading.

The instruction is `0x00c6ba23`. To do much of this problem, it's best to think in binary:

0	0	c	6	b	a	2	3
0000	0000	1100	0110	1011	1010	0010	0011

The instruction is `subu $s7, $a2, $a2` (`subu rd, rs, rt`).

4.5.1

ALU control's **input**: 2 bits from Control (ALU_{op}): 10; 6 bits from instruction: 100011

4.5.2

New address: PC + 4

4.5.3

There are four MUXes (1-4), from left to right.

MUX 1 inputs: 00110 (top; 0) and 10111 (bottom; 1); these represent $a2 and $s7, respectively.

MUX 1 output: 10111 ($s7).

MUX 2 inputs: Reg[$a2] (0) and 1111 1111 1111 1111 1011 1010 0010 0011 or 0xffffba23 (1).

MUX 2 output: Reg[$a2].

MUX 3 inputs: PC + 4 (0) and PC + 1111 1111 1111 1110 1110 1000 1001 0000 (1).

MUX 3 output: PC + 4.

MUX 4 inputs: result of read (unknown) (1) and 0 (0). The value at 0 (the bottom) is zero because the instruction is subtracting $a2 from itself.

MUX 4 output: 0, the result of the computation.

4.5.4

Left add inputs: PC (top) and 4 (bottom).

Right add inputs: PC+4 (top) and 1111 1111 1111 1110 1110 1000 1000 1100 (bottom).

ALU inputs: Reg[$a2] (top) and Reg[$a2] (bottom).

4.5.5

Read register 1 is $a2; Read register 2 is $a2; Write register is $s7; Write data is 0.

4.7

In this problem, refer to Figure 4.17, but it may be easier to think in terms of logical units within each instruction sub-area, as shown below:

4.7.1

IF		ID		EXE		MEM		WB	
Reg setup:	20	Reg setup:	20	ALU:	200	D-Mem:	250	MUX:	25
Reg read:	30	Reg file:	150	MUX:	25			Reg setup:	20
I-Mem:	250	MUX:	25					Reg file:	150
Subtotals:	300		195		225		250		195

Total: 1165ps (915 if not including D-Mem)

4.7.2

Similar to 4.7.1.

4.7.3

Similar to 4.7.1.

4.7.6

The clock period would be the maximum of all the above. Since in 4.7.1 we showed the total datapath (including D-Mem which is not needed for that instruction), 1165ps would be the clock period.

4.16.1 Non-pipelined: Add up all the stages

$250 + 350 + 150 + 300 + 200 = 1250$ps

Pipelined: Choose the longest stage = 350ps (we are assuming no splitting of stages; that comes in 4.15.3)

4.16.2 Non-pipelined: 1250ps

Pipelined: $350 \times 5 = 1750$ps (for lw, you still have to go through all of the stages)

4.16.3 Split the largest one: 350ps

The new clock cycle will be the largest remaining: 300ps

This is the very simple solution. What they're doing here is just adding a 6^{th} stage, but by doing this reducing the clock time per stage. If you want to double up the WB stage with the ID stage, this won't quite work, because each half of the ID stage takes 175ps which both won't finish within the new 300ps clock cycle time. So if you still wanted five stages and you want the benefit of having WB done in the first half of a cycle and ID done in the second half, the clock time would actually have to be 400ps so that WB can be done in half of that time (since it takes 200ps) and reading registers could happen in the second half (175ps).

Additional problems:

1

The circuit diagram should have 4 `OR` gates and 2 `AND` gates; the `AND` gates should be in parallel with one of the `OR` gates.

2

140ps.