HW 1 Solution

Note to Students:
If you have a different solution to a particular problem or if you have other things that you would like the TAs to be aware of during grading, please contact the corresponding TAs below: Abhishek will grade P1, Jian Tang will grade P2, and Jia Guo will grade P3.

Problem 1

1.1) Clock cycle time is determined by the critical path, which for the given latencies happens to be to get the data value for the load instruction:
For example,
lw $t0, –4($sp)

I-type format, MIPS Instruction formats

31-26	25-21	20-16	15-0
opcode	rs: register src	rt: register dest	offset
lw	$sp	$t0	-4

When students give the answer, they should give their assumption. If no assumption listed, the answer b) is prefered. Other answers may also be right if their assumption is reasonable.

The following answer ignores the latency of "ALU control"
I-Mem (read instruction), Regs (takes longer than Control), ALU, Data Memory. The latency of this path is:

a) No writeback latency. This is based on the assumption that the Write-back Latency is hidden when a new instruction is fetched (fetch and write-back work in parallel)
Without improvement 400 + 200 + 160 + 500 = 1260 ps **(5 pts)**
With improvement 400 + 200 + 160 + 120 + 500 =1380 ps **(5 pts)**

The answer b) is prefered.

b) Add Write-back Latency (at the same time need to add the MUX latency that is before the register),
Without improvement 400 + 200 + 160 + 500 + 20 + 200 = 1480 ps **(5 pts)**
With improvement 400 + 200 + 160 + 120 + 500 + 20 + 200 =1600 ps **(5 pts)**

Without improvement:

Operation	Location	Delay	Output
reading the instruction memory	I-Mem	400ps	Instruction[31-0]
reading the base register $sp	Regs	200ps	Read data 1
computing memory address $sp-4	ALU	160ps	ALU result
reading the data memory	D-Mem	500ps	Read data
choosing the read data –4($sp)	Mux	20ps	Mux output, when selecting channel 1
storing data back to $t0	Regs	200ps	Register t0

1.2) (1/0.85) * (1480/1600) = 1.088, so the speedup is 1.088, it is faster than original architecture. **(10 pts)**

1.3) The speedup comes from changes in clock cycle time and changes to the number of clock cycles we need for the program: We need x% fewer instructions, but cycle time is 1600 instead of 1480, so we have a speedup of (1/x)*(1480/1600) **>=** 1.2, x **<=** 0.77 so percentage = 1 - x **>=** 0.229 **(10 pts)**, which means that at least 22.9% instructions need to be eliminated.

Problem 2

2.1) **18 pts**

add **(3pts)**	addi **(3pts)**	bne **(3pts)**	lw **(3pts)**	sw **(3pts)**	nor **(3pts)**
36/282	48/282	42/282	72/282	42/282	42/282
12.766%	17%	14.9%	25.53%	14.9%	14.9%

2.2) The data memory is used by LW and SW instructions, so the answer is:

25.53% + 14.9% = 40.43% **(5 pts)**
100%**(2 pts)**

2.3) The sign-extend circuit is actually computing a result in every cycle. The input of the sign-extend circuit is needed for ADDI (to provide the immediate ALU operand), BNE (to provide the PC-relative offset), and LW and SW (to provide the offset used in addressing memory) **(5 pts)**

17% + 14.9% + 25.53% + 14.9% = 72.33% **(5 pts)**

Problem 3

This instruction is OR r3, r11, r8

3.1)

Sign-extend (3 pts)	**Shift-left-2 (2 pts)**
00000000000000000011100000100101	00000000000000000111000010010100

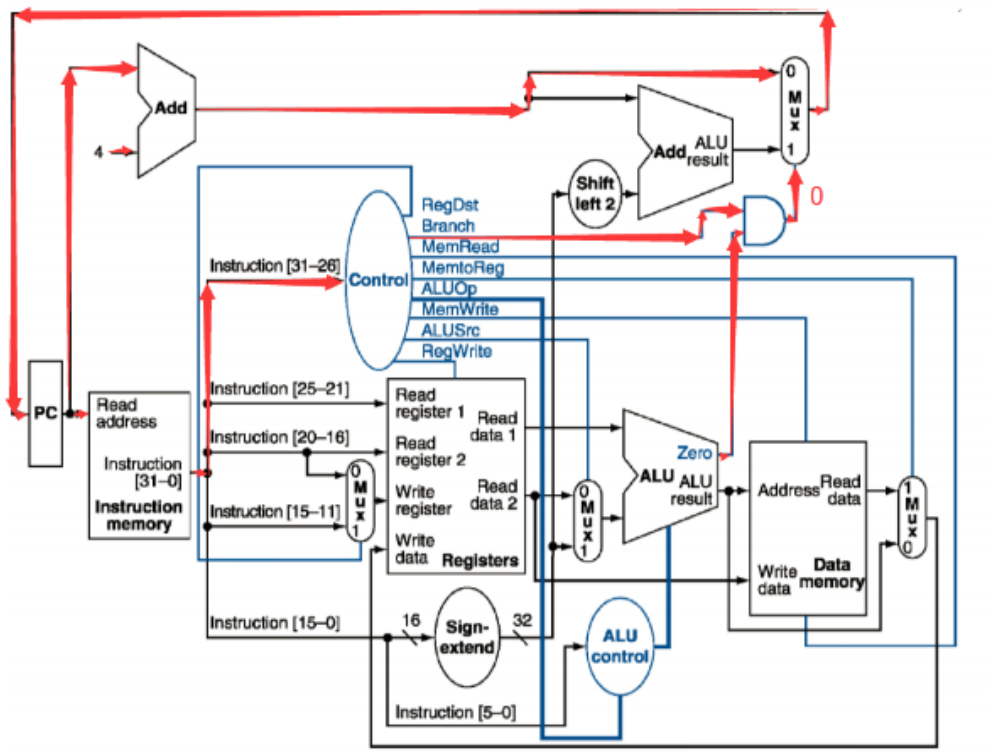
3.2)

ALUOp (3 pts)	**Instruction[5-0] (2 pts)**
10	100101

3.3)

New PC (2 pts)	**Path (3 pts)**

pc+4	PC to Add (PC+4) to branch Mux to PC



3.4)

WrReg Mux (2 pts)	ALU Mux (1 pts)	Mem/ALU Mux (1 pts)	Branch Mux (1 pts)
Instruction[15-11]: 3_{10} or 00011_2	99_{10}	141 OR 99 = 239_{10}	PC+4

3.5)

ALU (2 pts)	Add (PC+4) (1 pts)	Add (Branch)(2 pts)
141 and 99	PC and 4	PC+4 and 00000000000000000011000001001010 0_2

4.6)

Read Register 1 **(2 pts)**	Read Register 2 **(2 pts)**	Write Register **(2 pts)**	Write Data **(2 pts)**	RegWrite **(2 pts)**
11	8	3	243	1