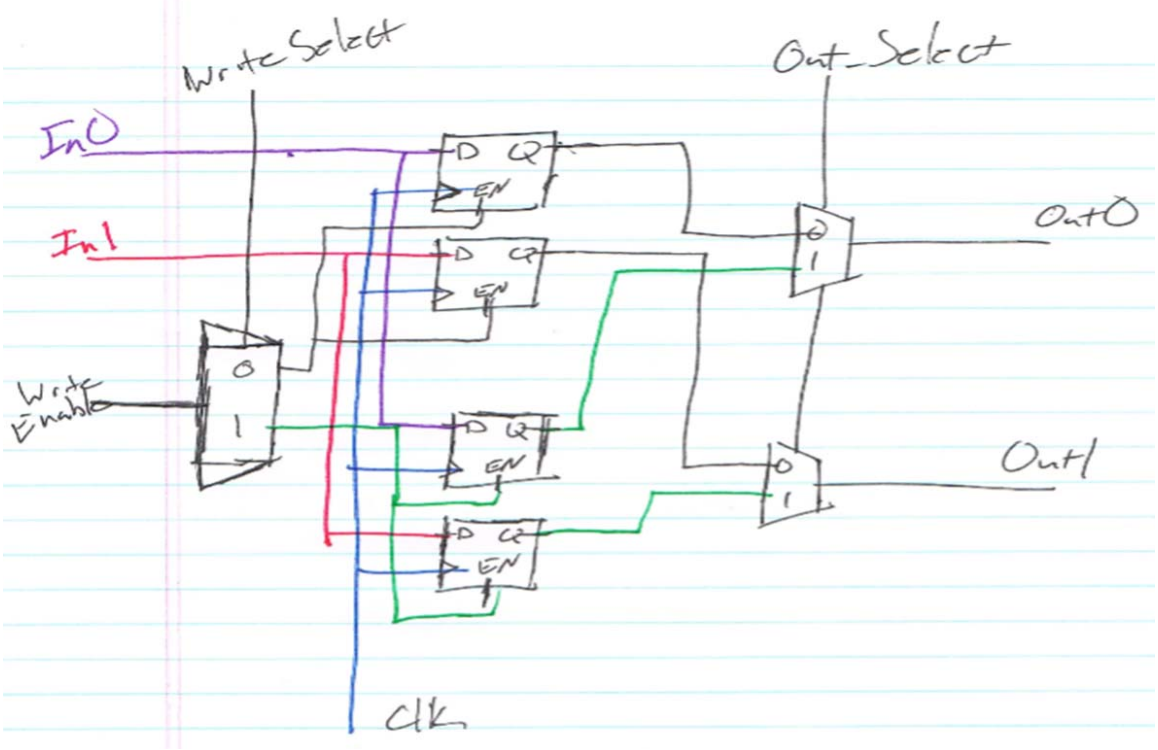
Homework 4 – Solutions

1. [Datapath]



2. [Datapath]

	T0	T1	T2	T3	T4	T5	T6
IN_R0	3	5	7	9	11	13	15
IN_R1	0	2	4	6	8	10	12

OUT is shifted one cycle:

OUT_R0	X	3	5	7	9	11	13	15
OUT_R1	X	0	2	4	6	8	10	12

K1	0	0	1	0	0	1	1
K2	1	0	0	0	1	0	1

ENABLE = K1 OR K2

ENABLE (LOAD)	1	0	1	0	1	1	1

IN_R2 = Select(OUT_R0,OUT_R1) where Select=K1 → effective on ***current*** cycle

IN_R2	X	3	2	7	9	8	10

OUT_R2 is shifted one cycle and selected by enable of ***previous*** cycle

OUT_R2	X	X	X	2	2	9	8	10

3. [Instruction Encoding/Decoding]
   a) 100011 01100 00111 0000000000110000
      0x8D870030
   b) Not enough information – the address of 'procedure' is needed to encode the instruction.
   c) 00100001111 10101 1111110000011000
      0x21F5FC18

4. [Instruction Encoding/Decoding]
   beq $10,$11,done
   Done is 3 instructions behind the *beq*, so the offset is -4.
   (Since the two least significant bits are dropped, variations are also acceptable here.)
   000100 01010 01011 1111111111111100
   0x114BFFFC

5. [Instruction Encoding/Decoding]
   a) j 0x10000A
   b) LW $12, 276($10)

6. [Converting MIPS AL to Bare MIPS]
   SLT $3, $5, $4
   BNE $3, $0, 16

7. [Converting MIPS AL to Bare MIPS]
   LUI $1, 0xEFEF
   ORI $1, 0xEFEF
   OR $8, $9, $1

8. [Converting MIPS AL to Bare MIPS]
   LUI $1, 0xEFEF
   ORI $1, $1, 0xEFEF
   ADD $11, $10, $1
   LW $9, $11

9. [Single Cycle CPU]

4.1

4.1.1 The values of the signals are as follows:

RegWrite	MemRead	ALUMux	MemWrite	ALUop	RegMux	Branch
0	0	1 (Imm)	1	ADD	X	0

ALUMux is the control signal that controls the Mux at the ALU input, 0 (Reg) selects the output of the register file, and 1 (Imm) selects the immediate from the instruction word as the second input to the ALU.

RegMux is the control signal that controls the Mux at the Data input to the register file, 0 (ALU) selects the output of the ALU, and 1 (Mem) selects the output of memory.

A value of X is a "don't care" (does not matter if signal is 0 or 1)

4.1.2 All except branch Add unit and write port of the Registers

4.1.3 Outputs that are not used: Branch Add, write port of Registers

No outputs: None (all units produce outputs)

10. [Single Cycle CPU]

4.2

4.2.1 This instruction uses instruction memory, both register read ports, the ALU to add Rd and Rs together, data memory, and write port in Registers.

4.2.2 None. This instruction can be implemented using existing blocks.

4.2.3 None. This instruction can be implemented without adding new control signals. It only requires changes in the Control logic.

11. [Single Cycle CPU]

4.3.1 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: I-Mem (read instruction), Regs (takes longer than Control), Mux (select ALU input), ALU, Data Memory, and Mux (select value from memory to be written into Registers). The latency of this path is 400 ps + 200 ps + 30 ps + 120 ps + 350 ps + 30 ps = 1130 ps. 1430 ps (1130 ps + 300 ps, ALU is on the critical path).

12. [Single Cycle CPU]

4.4

4.4.1 I-Mem takes longer than the Add unit, so the clock cycle time is equal to the latency of the I-Mem:

200 ps

4.4.2 The critical path for this instruction is through the instruction memory, Sign-extend and Shift-left-2 to get the offset, Add unit to compute the new PC, and Mux to select that value instead of PC+4. Note that the path through the other Add unit is shorter, because the latency of I-Mem is longer that the latency of the Add unit. We have:

200 ps + 15 ps + 10 ps + 70 ps + 20 ps = 315 ps

4.4.3 Conditional branches have the same long-latency path that computes the branch address as unconditional branches do. Additionally, they have a long-latency path that goes through Registers, Mux, and ALU to compute the PCSrc condition. The critical path is the longer of the two, and the path through PCSrc is longer for these latencies:

200 ps + 90 ps + 20 ps + 90 ps + 20 ps = 420 ps

4.4.4 PC-relative branches.

4.4.5 PC-relative unconditional branch instructions. We saw in part c that this is not on the critical path of conditional branches, and it is only needed for PC-relative branches. Note that MIPS does not have actual unconditional branches (bne zero,zero,Label plays that role so there is no need for unconditional branch opcodes) so for MIPS the answer to this question is actually "None".

4.4.6 Of the two instructions (BNE and ADD), BNE has a longer critical path so it determines the clock cycle time. Note that every path for ADD is shorter than or equal to the corresponding path for BNE, so changes in unit latency will not affect this. As a result, we focus on how the unit's latency affects the critical path of BNE.

This unit is not on the critical path, so the only way for this unit to become critical is to increase its latency until the path for address computation through sign extend, shift left, and branch add becomes longer than the path for PCSrc through registers, Mux, and ALU. The latency of Regs, Mux, and ALU is 200 ps and the latency of Sign-extend, Shift-left-2, and Add is 95 ps, so the latency of Shift-left-2 must be increased by 105 ps or more for it to affect clock cycle time.