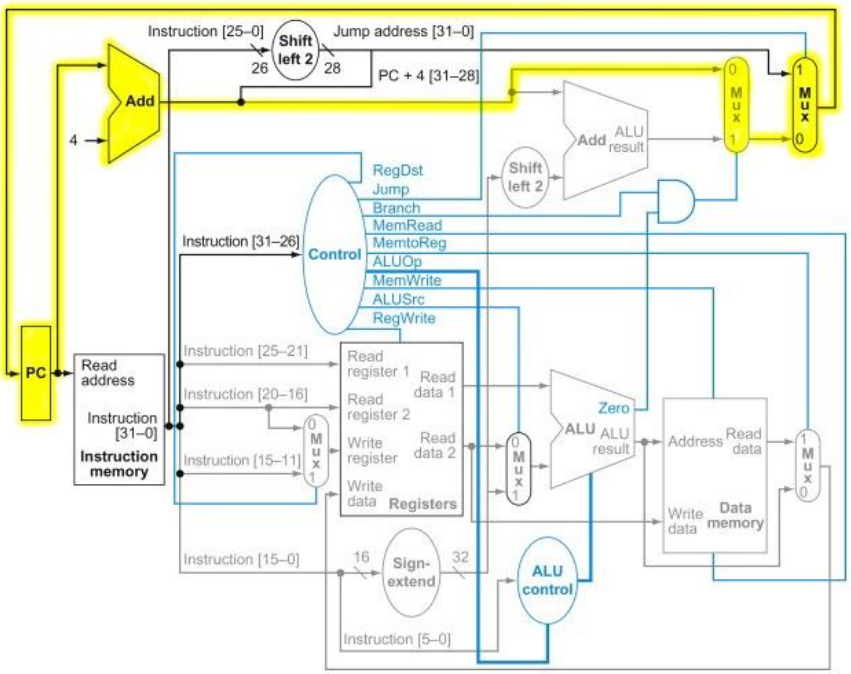
CS161
Assignment-2 Solution

1.
   a. This instruction uses 1) instruction memory, 2) read ports in Registers, 3) the ALU to add Rd and Rs together, 4) data memory, and 5) the write port in Registers.
   b. Unlike regular R-type instructions, this new instruction gets Rd as its second source operand (in regular R-type instructions, Rd is the destination register). So we need to add a MUX for the second source operand (either it is Rt (bits 20-16) or Rd (bits 15-11)).
   c. We need to add a signal in order to control the new MUX, we call it SrcReg.
2.
   a. I-Mem takes longer than the Add unit, so the clock cycle time is equal to the latency of the I-Mem: 200ps
   b. The critical path for this instruction is through the instruction memory, sign-extend and shift-left-2 to get the offset, and the adder. *We do not include the PCSrc Mux because the processor only supports this one type of instruction, and will perform an unconditional PC-relative branch every cycle.*
      We have: 200ps + 15ps + 10ps + 70ps = 295ps
   c. 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 I-Mem, 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: 200ps + 90ps + 20ps + 90ps = 400ps.
3.
   a. The data memory is used by LW and SW instructions, so the answer is 25% + 10% = 35%
   b. The sign-extend circuit is actually computing a result in every cycle, but its output is ignored for ADD and NOT instructions. The input of the sign-extend circuit is needed for ADDI (to support the immediate ALU operand), BEQ (to provide the PC0Related offset) and LW and SW (to provide the offset used in addressing memory) so the answer is 20% + 25% + 25% + 10% = 80%
4.
   a. Output of sign-extend: `0000 0000 0000 0000 0000 0000 0001 0100`
      Output of shift-left-2: `0001 1000 1000 0000 0000 0101 0000`
   b. ALUOp[1-0]: `00`
      Instruction[5-0]: `0001 0100 (20)`
   c. New PC: PC + 4
      Path: See below image. PC to Add (PC + 4) to branch MUX to jump MUX to PC



d. WrReg MUX: `2 or 0 (RegDst: X)`
   ALU MUX: `0001 0100 (20)`
   Mem/ALU MUX: `X`
   Branch MUX: `PC + 4`
   Jump MUX: `PC + 4`

e. ALU: `-3, 0001 0100 (20)`
   Add (PC + 4): `PC, 4`
   Add (Branch): `PC + 4, 0101 0000 (80)`

f. Read Register 1: `00011 (3)`
   Read Register 2: `00010 (2)`
   Write Register: `X/?`
   Write Data: `X/?`
   RegWrite: `0`

5. Similar to R-type instructions, addi instruction needs 4 cycles to be executed (instruction fetch, instruction decode/register fetch, execution, write-back). First two steps are identical to the ones that are shown in the figure (in question), here is next two steps:

   Execution:

   AluSrcA=1 // Selects rs register to ALU
   AluSrcB=10 // Selects sign-extended offset field to ALU
   AluOP=00 // Tells ALU to perform Add operation.

   Write-back:

   MemtoReg=0 // ALUOut to Register's Write Data port
   RegWrite // Tells register to write
   RegDst=0 // Write to rt register