CS320 Fall 2014

Homework#4
Quiz in Lecture Thursday October 9, 2014
There are no make-up quizzes!

Problem 1: Consider the following timing values for the single cycle datapath (Figure 4.24).

I-Mem	Adder	Mux	ALU	Reg File	D-Mem	Sign Ext	Shift Left 2	ALU Cntl
250ps	120ps	5ps	70ps	100ps	200ps	10ps	5ps	20ps

a. Highlight each of the following instruction on the single cycle datapath: LW, SW, BEQ, ADD, SLT.

b. What is the critical path timing for each of the instructions based on the above timings? Assume the Control Unit has negligible timing.

c. Assume you are designing the control unit and must know the timing requirements for the unit. How much time can the control unit take to generate the MemWrite Signal (critical path from input:opcode to output of signal: MemWrite) without impacting the overall datapath timing?

To not extend the critical path (aka the overall datapath timing), we need to make sure that the MemWrite signal does not lengthen the sw instruction longer than the lw instruction (current longest instruction, 725ps). The sw instruction takes 620ps to execute (I-mem, Read regFile, ALU, D-mem). The D-mem must know to write the data before its execution (saving of data) therefore 620-200ps = 420ps is when the address data is available at the D-mem. LW is 725ps, therefore the difference between the sw and lw instruction is 105ps. The MemWrite signal must be available no later than 525ps to make sw the same length of execution as lw.

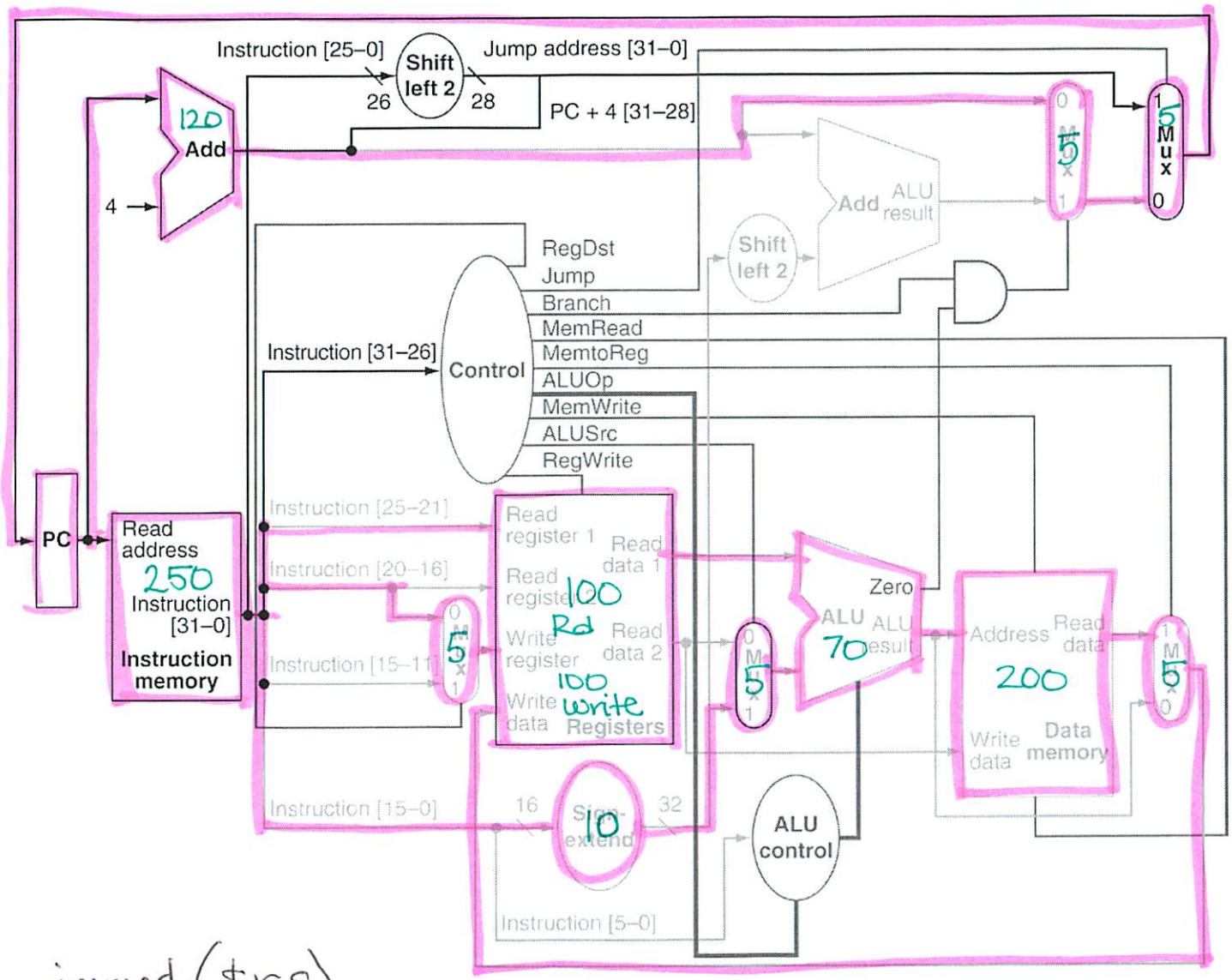
d. In addition to above, now assume that the Control Unit requires the following amount of time to generate the control signals What is the clock cycle time of the processor?

RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	Jump
50ps	10ps	45ps	50ps	20ps	50ps	45ps	20ps	50ps

Clock cycle time is 725ps

PC ← PC+4 130ps

Reg[Rt] ← Mem[Reg[rs]+imm] 725ps



lw $rt, immed($rs)

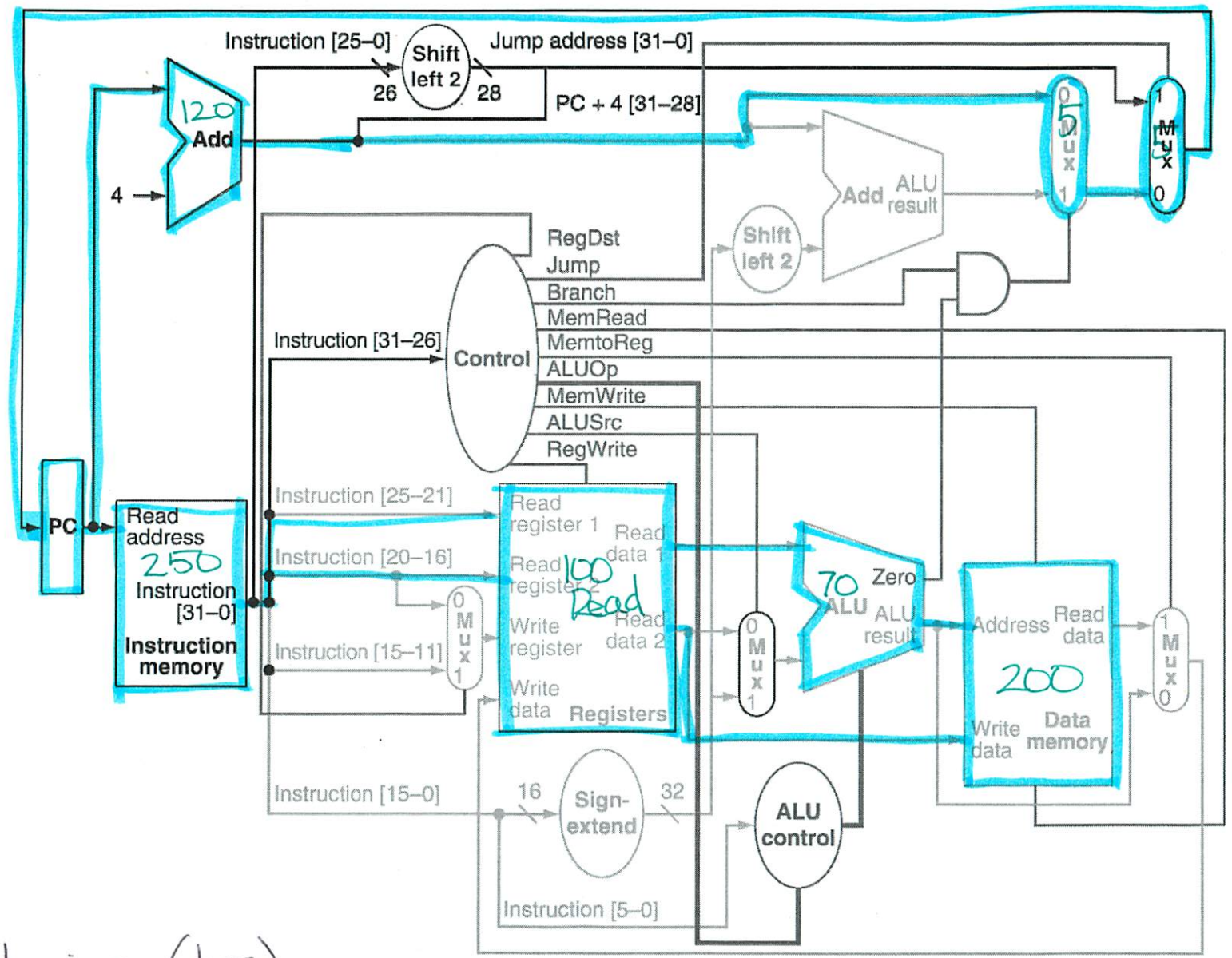
725ps

RegFile, ALU, D-mem, MemtoReg, RegFile Write

Imem, Read

PC ← PC+4 130ps

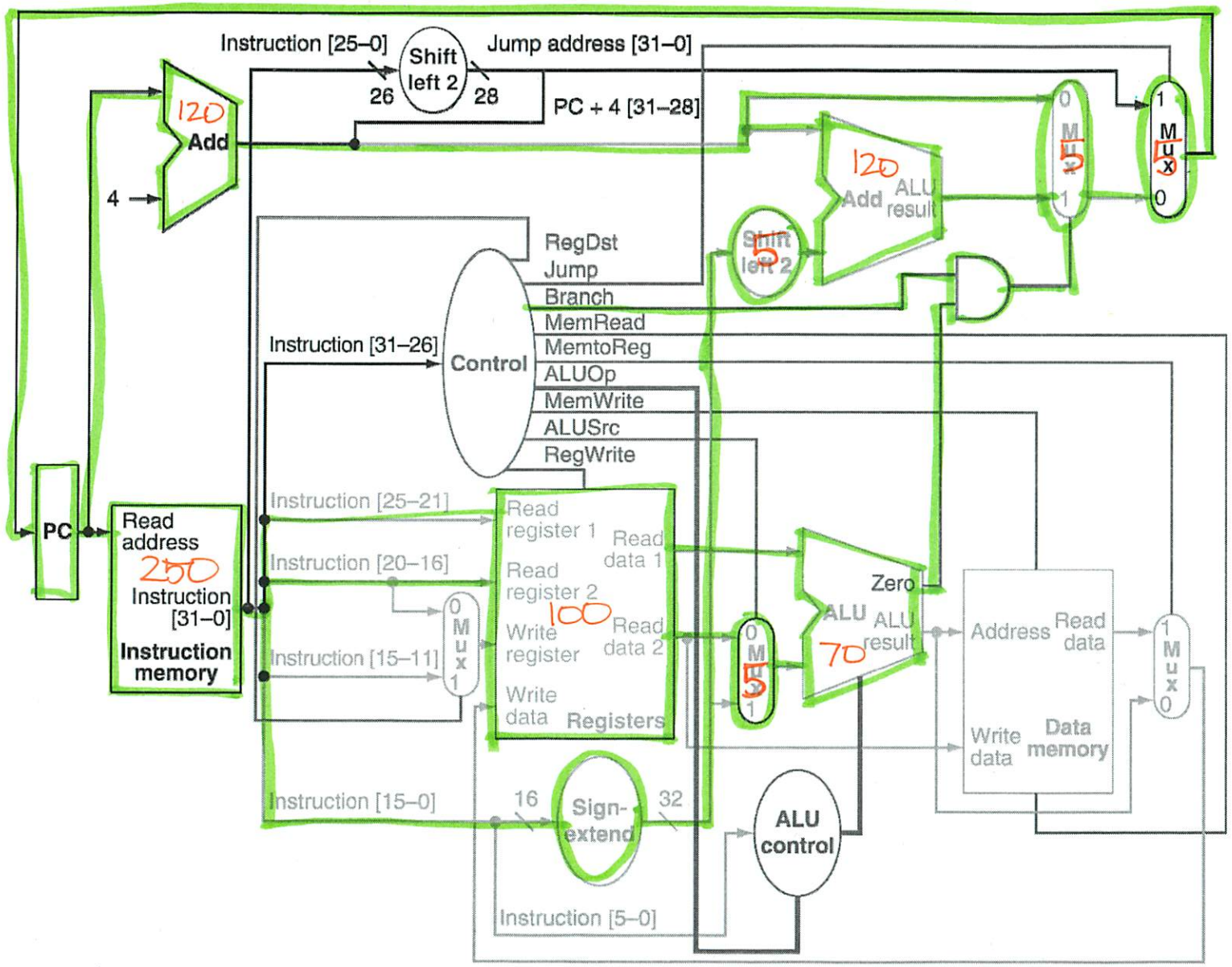
Mem[Reg[Rs]+immed] = Reg[Rt] 620ps



Sw $rt, imm ($rs)

620ps

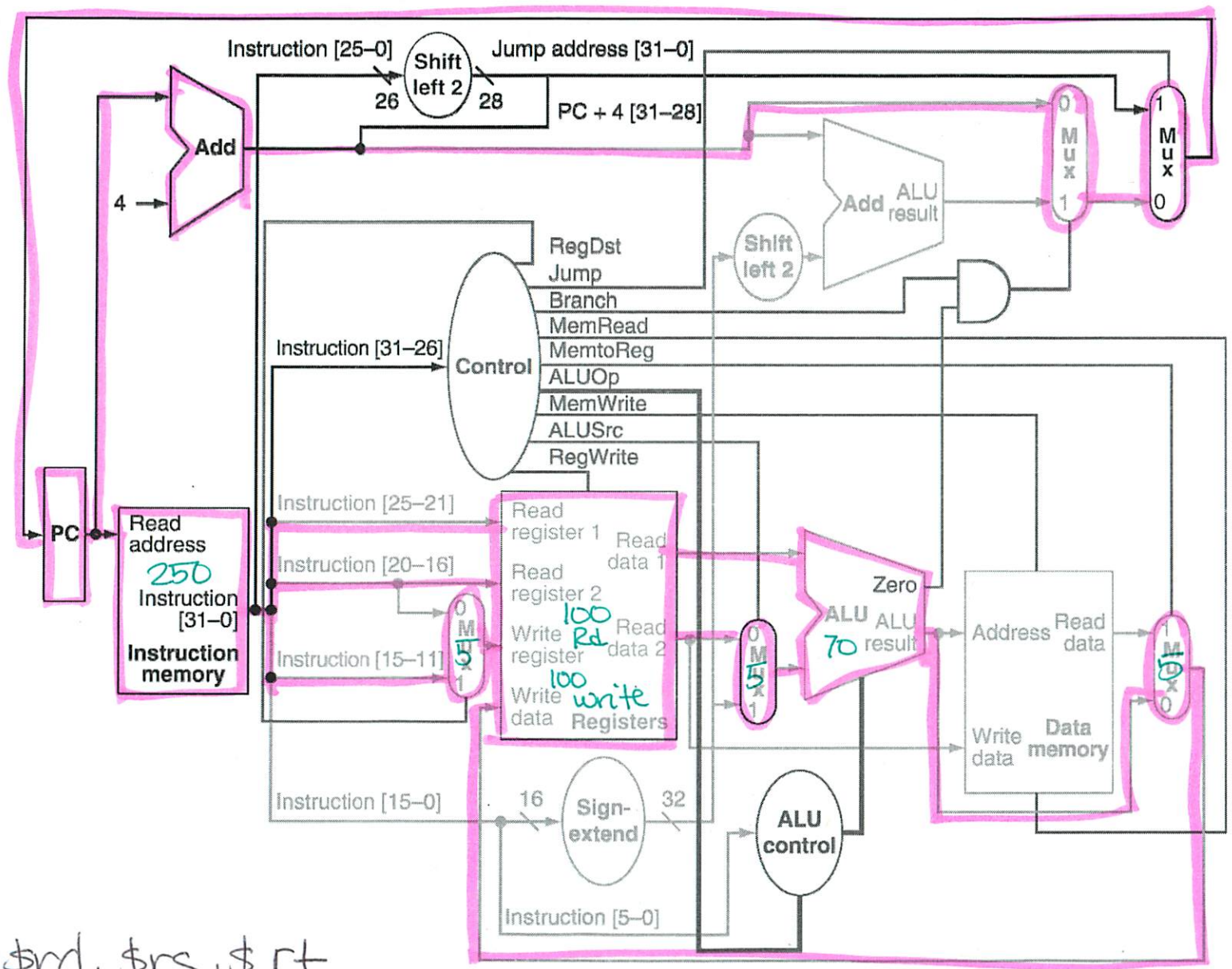
I-mem, Reg File, ALU, Data Mem
Read Write.



beq $rs, $rt, label

435ps

I-mem, Reg File Read, ALUSrc mux, ALU, Branch mux, Jump mux



slt $rd, $rs, $rt
add $rd, $rs, $rt
530ps

I-mem, RegFile Read, ALUSrc mux, ALU, MemtoReg Mux, RegFile Write

Problem 2: In class we covered the MIPS single-cycle implementation which handled only a subset of the MIPS instructions: R-type (add, sub, and, or, slt), memory references (lw, sw), conditional branch (beq) and jump (j). In this problem we will add to the implementation functionality for additional MIPS instructions. Use the datapath in (Figure 4.24) and control table below to specify your changes, add a row for each instruction and any additional columns (new control signals) to the control table as necessary. Be sure to mark don't cares in the control table whenever possible.

Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUp0	Jump
R	1	0	0	1	0	0	0	1	0	0
lw	0	1	1	1	1	0	0	0	0	0
sw	X	1	X	0	0	1	0	0	0	0
beq	X	0	X	0	0	0	1	0	1	0
j	X	X	X	0	X	0	X	X	X	1

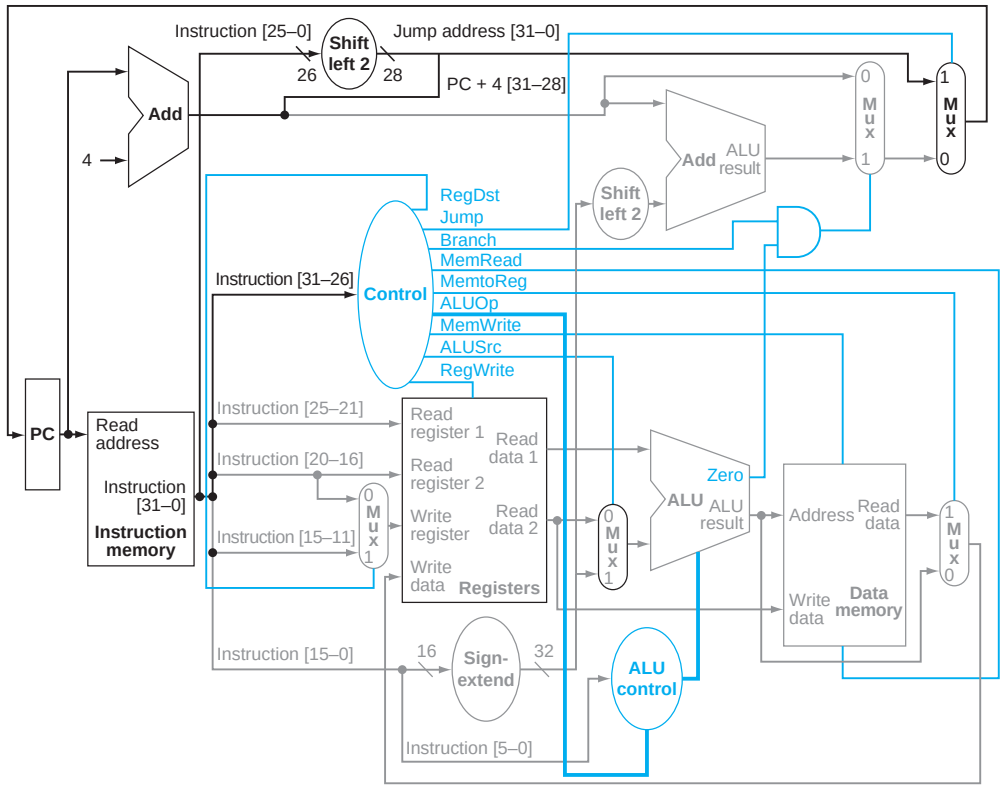


FIGURE 4.24 The simple control and datapath are extended to handle the jump instruction. An additional multiplexor (at the upper right) is used to choose between the jump target and either the branch target or the sequential instruction following this one. This multiplexor is controlled by the jump control signal. The jump target address is obtained by shifting the lower 26 bits of the jump instruction left 2 bits, effectively adding 00 as the low-order bits, and then concatenating the upper 4 bits of PC + 4 as the high-order bits, thus yielding a 32-bit address. Copyright © 2009 Elsevier, Inc. All rights reserved.

Problem Guidelines:

- When adding new instructions, don't break the operation of the standard ones.
- Avoid adding ALUs, adders, Reg Files, or memories to the datapath
- You can add MUXes, logic gates, etc. but try to do minimally. (these cost in terms of area, cycle time, etc)

a. Modify the datapath and the control table to implement the 'jal' instruction.
Reg[31] ← PC+4 # $ra register stores the return address
PC ← PC+4[31:28], (Instruction[25:0] << 2)

PC is the top 4 bits of the PC+4 value concatenated with the lowest 26 bits of the jump address shifted to the left by 2 bits

b. Modify the datapath and the control table to implement a new 'lw3r' (load word 3 registers) instruction.
lw3r Rd, Rs, Rt # Reg[Rd] = Mem[Reg[Rt] + Reg[Rs]]

c. Modify the datapath and the control table to implement a new 'lbu' (load byte unsigned) instruction.
lbu Rt, offset (Rs) # Reg[Rt] = (no sign extension) Mem[Reg[Rs]+offset]

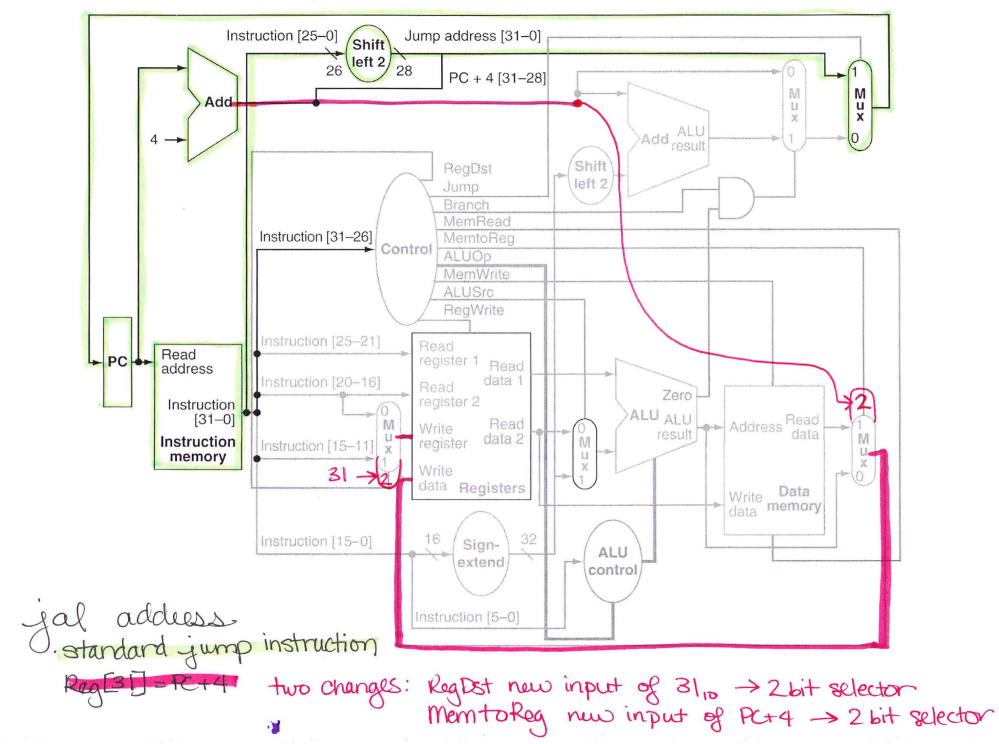
d. Modify the datapath and the control table to implement a new 'beven' (branch on even) instruction.
beven Rt, label # if Rt is even, PC $\leftarrow$ PC + 4 + (Instruction[15:0] << 2)

e. Modify the datapath and the control table to implement a new 'slt12' (set on less than 12) instruction.
slt12 Rt, Rs # if Rs $<$ 12, Rt = 1, else Rt = 0

f. Modify the datapath and the control table to implement a new 'beqi' (branch on equal to immediate) instruction.
beqi Rt, immed, label # if Rt == 5-bit 2's complement value in Rs field, PC $\leftarrow$ PC + 4 + (Instruction[15:0] << 2)

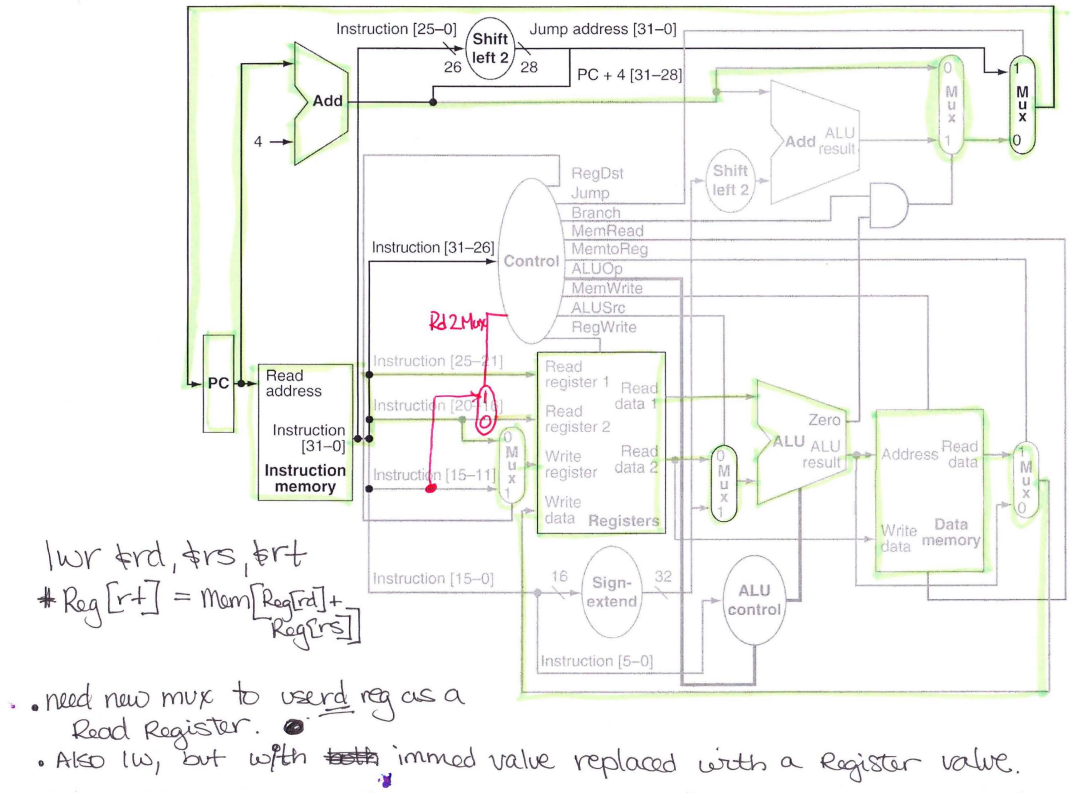
Problem 3: Using the timing for the units given in Problem 1, calculate the timing delays for each of the new instructions in Problem 2. Which instruction determines the clock cycle time?
ANSWER:

See control tables and timing at the end for each instruction INDEPENDENTLY.



Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUp0	Jump
R	01	0	00	1	0	0	0	1	0	0
lw	00	1	01	1	1	0	0	0	0	0
sw	XX	1	XX	0	0	1	0	0	0	0
beq	XX	0	XX	0	0	0	1	0	1	0
j	XX	X	XX	0	0	0	X	X	X	1
jal	10	X	10	0	0	0	X	X	X	1

jal: I-mem, Shift left 3, jump mux 230ps



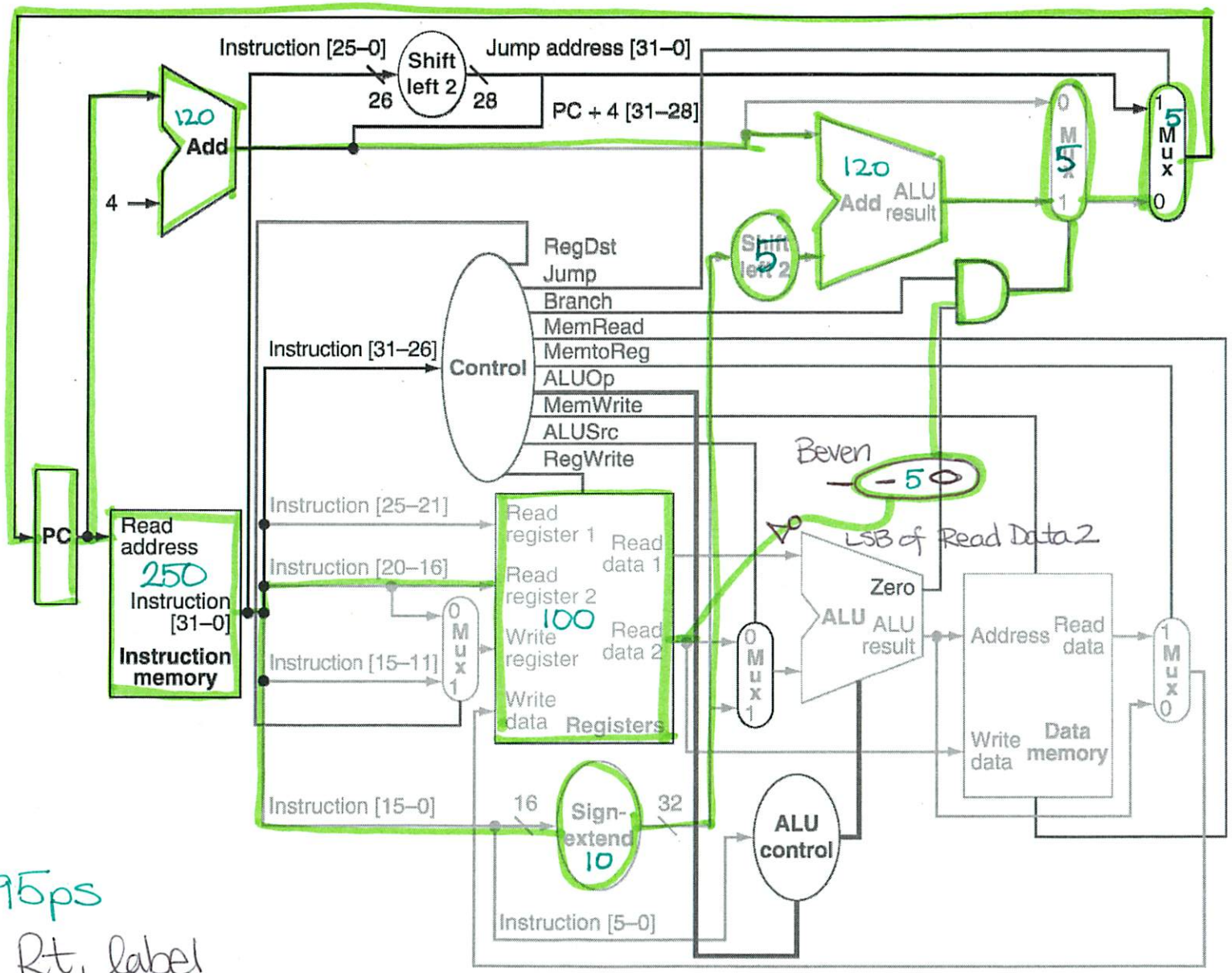
Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUp0	Jump	Rd2Mux
R	1	0	0	1	0	0	0	1	0	0	0
lw	0	1	1	1	1	0	0	0	0	0	0
sw	X	1	X	0	0	1	0	0	0	0	0
beq	X	0	X	0	0	0	1	0	1	0	0
j	X	X	X	0	0	0	X	X	X	1	0
lw3r	0	0	1	1	1	0	0	0	0	0	1

lw3r: I-mem, RegDst mux, Read RegFile, ALUSrc mux, ALU, D-mem, MemtoReg mux, Write RegFile 780ps

Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUp0	Jump	Beven
R	1	0	0	1	0	0	0	1	0	0	X
lw	0	1	1	1	1	0	0	0	0	0	X
sw	X	1	X	0	0	1	0	0	0	0	X
beq	X	0	X	0	0	0	1	0	1	0	0
j	X	X	X	0	0	0	X	X	X	1	X
beven	X	X	X	0	0	0	1	X	X	0	1

Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUp0	Jump
R	1	00	0	1	0	0	0	1	0	0
lw	0	01	1	1	1	0	0	0	0	0
sw	X	01	X	0	0	1	0	0	0	0
beq	X	00	X	0	0	0	1	0	1	0
j	X	XX	X	0	0	0	X	X	X	1
slt12	0	10	0	1	0	0	0	1	0	0

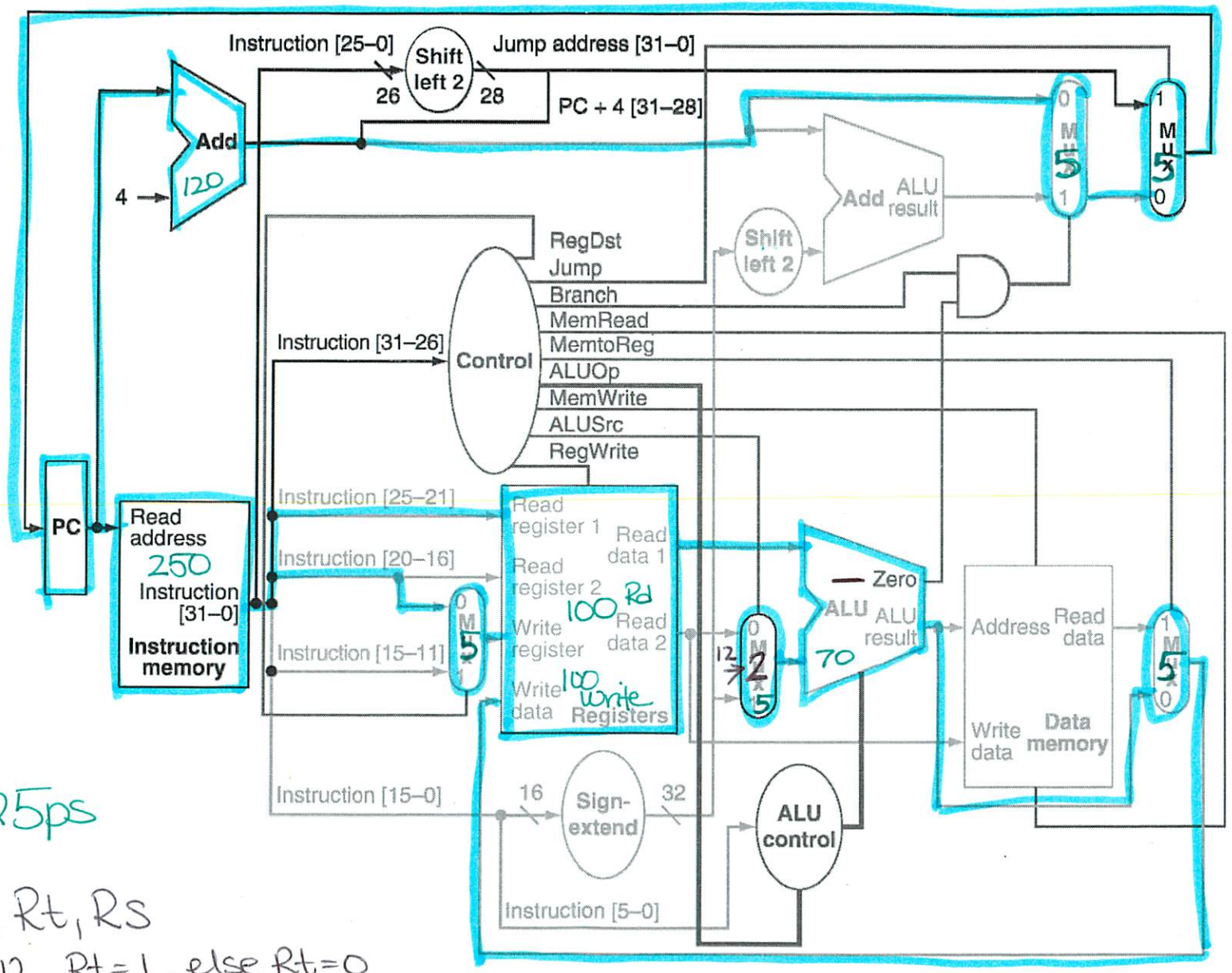
Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUp0	Jump	beqi
R	1	0	0	1	0	0	0	1	0	0	0
lw	0	1	1	1	1	0	0	0	0	0	0
sw	X	1	X	0	0	1	0	0	0	0	0
beq	X	0	X	0	0	0	1	0	1	0	0
j	X	X	X	0	0	0	X	X	X	1	X
beqi	X	0	X	0	0	0	1	0	1	0	1



395ps

beven Rt, label

-new control signal Beven. Set to 1 for Beven, set to zero for ~~all~~ beq. Don't Care for all other instructions

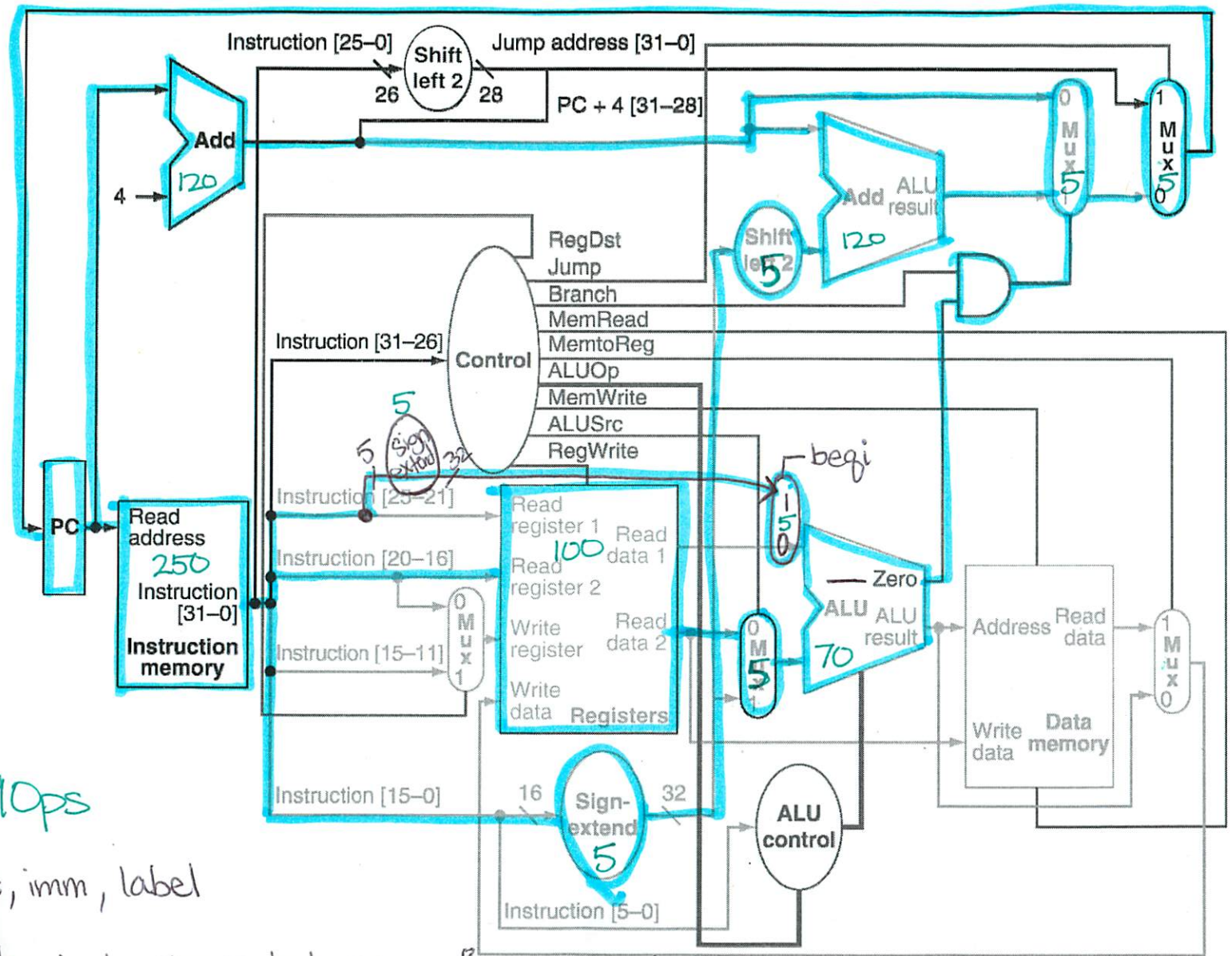


525ps

SLT12 Rt, RS

if RS<12, Rt=1, else Rt=0

- Modify ALUSrc to take additional input (value 12)



390ps

beqi Rt, imm, label

- new mux beqi to choose between 5 bits in rs or Reg[rs]

sign extended