UNIVERSITY OF CALIFORNIA, LOS ANGELES UCLA

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO SANTA BARBARA • SANTA CRUZ

CS M151B / EE M116C

Midterm Exam

All work and answers should be written directly on these pages, use the backs of pages if needed.

This is an open book, open notes quiz – but you cannot share books or notes.

We will follow the departmental guidelines on reporting incidents of academic dishonesty – do not make us enforce the rules. Keep your eyes on your own exam!

NAME: Yu Li

ID: [redacted] 705063404

Do not write anything in the area below on this page:

Problem 1: 10 (10)

Problem 2: 20+9=29 (30)

Problem 3: 30 (30)

Problem 4: 25 (30)

Total: 94 (out of 100)

1. ***This One ISA Warmup (10 points)***: Consider the single cycle datapath we covered in class. The processor has a 3 GHz clock. This datapath will execute an application with 10 billion instructions. The application has the following instruction mix.

Instruction	% of Instructions
Load	30%
Store	10%
Simple R-type (i.e. adds, ands, shifts)	45%
BEQ/BNE	10%
Jump	5%

What is the CPI for this application running on this processor?

CPI: 1

2. ***Your Complaints Have Been Registered (30 points)***: A particular application is being run on a MIPS processor – this processor is not a single cycle datapath and is not a pipelined datapath. It only executes one instruction at a time, but instructions can take anywhere from 3 cycles to 7 cycles to execute. R type instructions take 5 cycles, LW instructions take 7 cycles, SW instructions take 6 cycles, and BEQ/BNE instructions take 3 cycles. For our application, 30% of instructions are LW, 10% of instructions are SW, 15% of instructions are BEQ/BNE, and 45% of instructions are R type. We will execute 10 billion instructions on a processor with a 2 GHz clock. Find the CPI and Execution Time for this application/processor combination. ***Show your work.***

CPI: 5.4

ET: 27s

$CPI = 0.3 \times 7 + 0.1 \times 6 + 0.15 \times 3 + 0.45 \times 5$

$= 5.4$

$ET = IC \times CPI \times CT$

$= 10 \times 10^9 \times 5.4 \times \frac{1}{2 \times 10^9} = 27s$

We are now going to modify the MIPS ISA and architecture. The particular application we are running requires large immediates – instead of 16 bits for the I type immediate field, we want to grow it to 18 bits. But we want to keep the instruction length fixed at 32 bits. Instead, we will reduce the register file from 32 registers to 16 registers. This will give us enough bits in the I type instruction to add two more bits to the immediate field. J type instructions will not change. R type instructions will also use fewer bits for registers – only four bits required for each address. We will use the extra three bits to increase the function field for R type, allowing us to add a richer set of operations we can perform for R type instructions. But, the smaller register file size comes with its own tradeoffs. The smaller register file will hurt us because we will spill to memory more often – that means more LW and potentially SW instructions to perform register spilling. One additional benefit from the smaller register is that it will have a lower latency to access – which could mean a faster clock. Use the following information to solve this question:

- We will not be using a single cycle datapath for this example. Instructions will take multiple cycles – but there is no pipelining. So an instruction must finish in the processor before another instruction comes along.
- The extra bits in the immediate field for I type instructions means that we will need fewer R type instructions in certain cases. Specifically, 50% of LW instructions will no longer need a prior add instruction to set up a larger immediate. In other words, half of the LW instructions will not need a single R type instruction before them – and for each of those LW instructions we can remove that single R type instruction from the code.
- The extra functions we can perform (thanks to the increase in function field bits for R type instructions) will allow us to reduce the number of R type instructions by 25%. This reduction is completely unrelated to the reduction from growing the immediate field for LW instructions – and so the 25% is relative to the original number of R type instructions.
- The smaller register file size results in a 40% increase in LW instructions and a 10% increase in SW instructions due to register spilling.
- The smaller register file size will reduce the cycle time by 30%, but will not affect how many cycles different instructions take.

Find the CPI and Execution Time for this application/processor combination for this new configuration. ***Show your work.***

30 LW → 45R

25% x 45R → -11.25R 18.75R

40% x 30LW → + 12LW 42LW

10% x 10SW → + 1SW 11SW

15 BEQ/BNE

$\frac{18.75}{86.75} \times 5 + \frac{42}{86.75} \times 7 + \frac{11}{86.75} \times 6 + \frac{15}{86.75} \times 3 = 20.125$

CPI: 5.75

ET: 20.125s.

$ET = IC \times CPI \times CT$

$= 10 \times 10^9 \times 5.75 \times \frac{1}{2 \times 10^9} + 0$

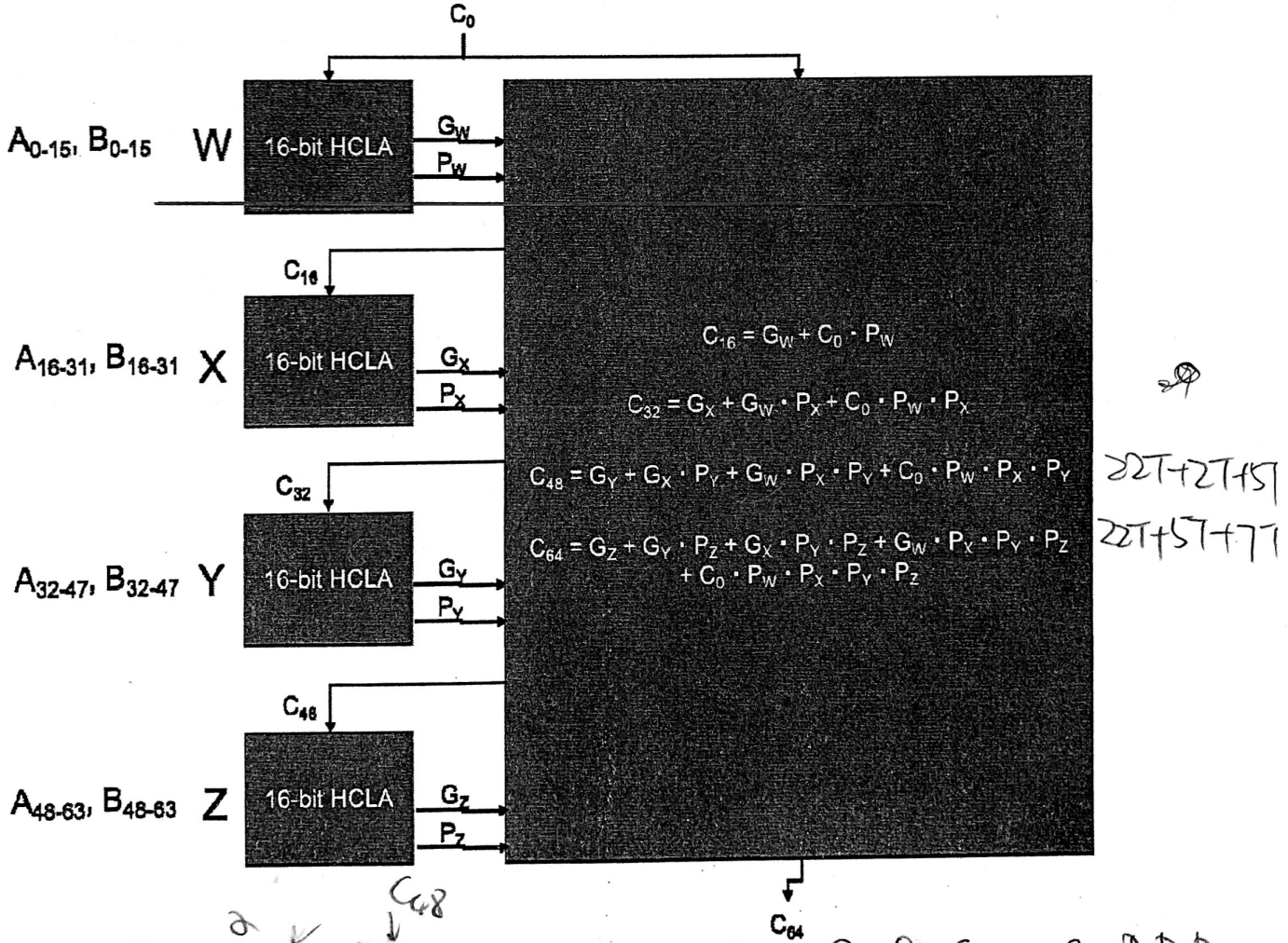
Total inst. not the same

3. ***Putting the CLA into UCLA (30 points)***: Assume for the rest of this problem that logic gates have the following delays:

Fan In	Delay AND/OR	Delay XOR
2	T	2T
3	2T	3T
4	5T	6T
5	7T	9T
6	10T	12T
7	13T	16T
8	15T	18T

So a 2-input AND gate would have delay T and a 4-input OR gate would have delay 5T. For simplicity, assume that mux's have delay 10T regardless of fan-in.

We are going to build a 64-bit adder. Your friend proposes the following design:



22T+27+5T

22T+5T+7T

A_{48-51}, B_{48-51} 45

A_{52-55}, B_{52-55} 45

A_{56-59}, B_{56-59} 45 → G_0

A_{60-63}, B_{60-63} 45 → P_0

$C_{64} = G_Y + G_B \cdot P_Y + G_2 P_B G_Y + C_{48} P_2 P_B P_Y$

29T + 5T

5T

29T

This is based on our hierarchical carry lookahead added from class. The four 16-bit HCLA's in this figure are labeled W, X, Y, and Z. Each handles 16 bits of the 64 bit computation. For example, HCLA W handles input bits $A_{0\text{-}15}$ and $B_{0\text{-}15}$ and produces output bits $S_{0\text{-}15}$. Your friend did not give the equations for the generates and propagates coming out of the four 16-bit HCLAs. First – provide the equations for:

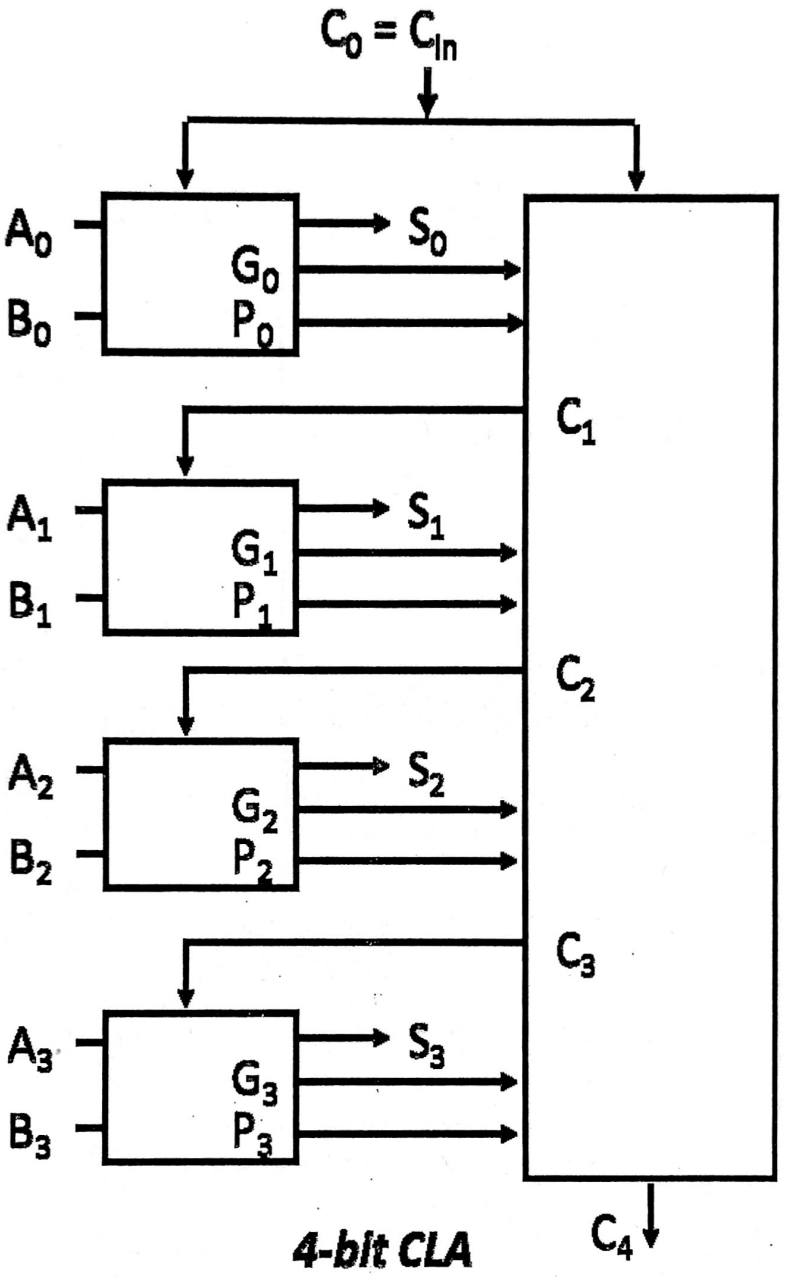
$G_W = G_\alpha + G_\beta P_\alpha + G_\gamma P_\alpha P_\beta + G_\delta P_\alpha P_\beta P_\gamma$ ✓ *(2 points)* (4)

$P_W = P_\alpha P_\beta P_\gamma P_\delta$ ✓ *(2 points)*

These equations should be in terms of G_alpha, P_alpha, G_beta, P_beta, G_gamma, P_gamma, G_delta, and P_delta, where G_alpha, P_alpha, G_beta, P_beta, G_gamma, P_gamma, G_delta, and P_delta are the Gs and Ps generated from the four 4-bit CLA's inside of the 16-bit HCLA labeled W.

Your task is to find the maximal delay of this design – i.e. determine the delays of $S_{0\text{-}63}$ and C_{64} – the maximal delay of these outputs will be the maximal delay of the entire design. To do this (and to help with possible partial credit) please use the diagrams on the following pages and fill in the tables in every page.

Single 4-bit CLA:



4-bit CLA

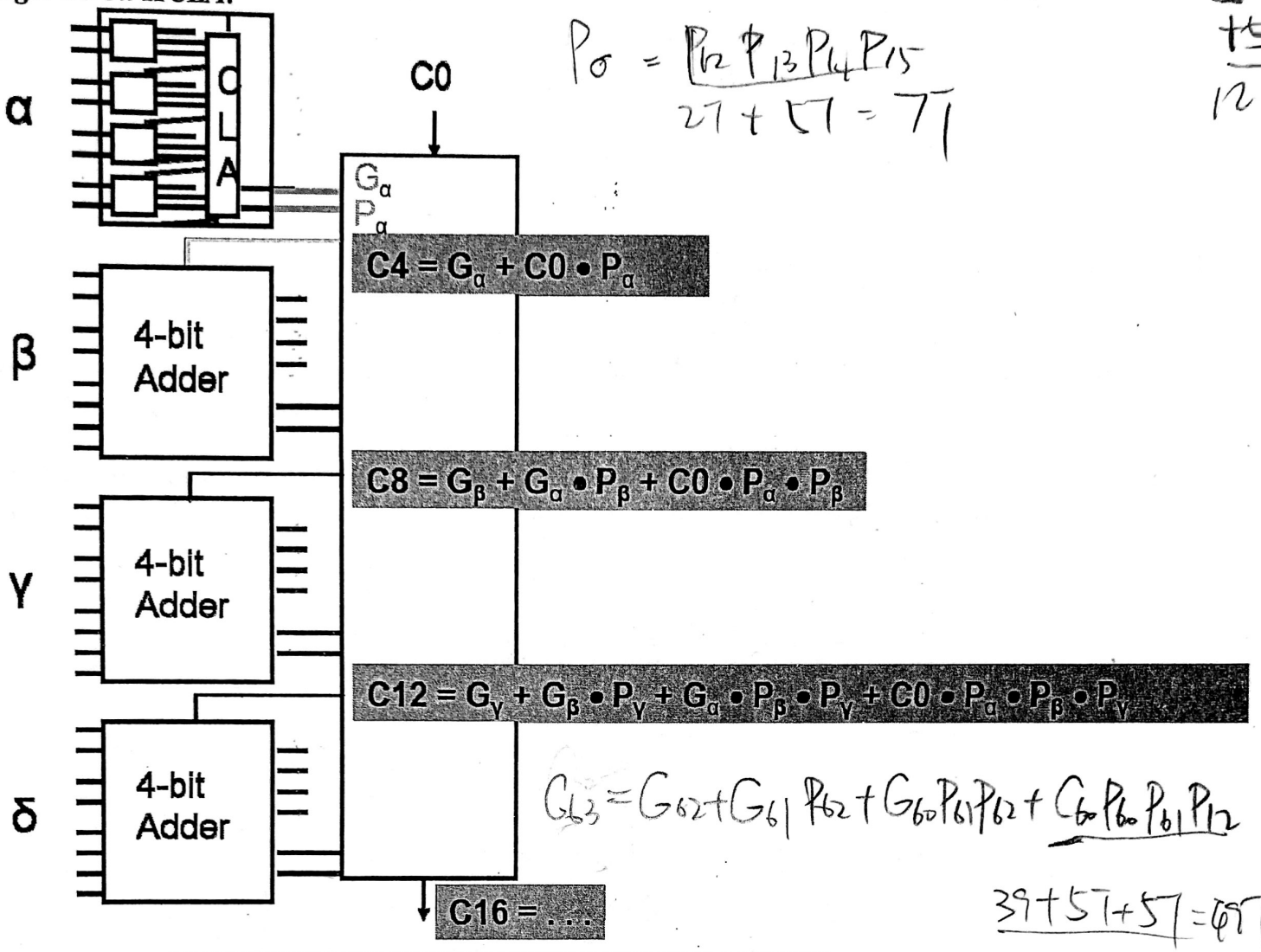
Output	Delay (in terms of T)	
G0	T ✓	*(2 points)*
P0	2T ✓	*(2 points)*
G3	T ✓	*(1 points)*
P3	2T ✓	*(1 points)*

(6)

$G_0 = A_0 B_0$ T $G_3 = A_3 B_3$

$P_0 = A_0 \wedge B_0$ 2T $P_3 = A_3 \wedge B_3$

Single 16-bit HCLA:



Output	Delay (in terms of T)	
G_δ	12T ✓	*(2 points)*
P_δ	7T ✓	*(2 points)*
Gw	22T ✓	*(2 points)*
Pw	12T ✓	*(2 points)*
Gz	22T ✓	*(2 points)*
Pz	12T ✓	*(2 points)*
C48	29T ✓	*(2 points)*
C64	34T ✓	*(2 points)*
C60	39T ✓	*(2 points)*
C63	49T ✓	*(2 points)*
S63	52T ✓	*(2 points)*

Find the maximum delay **in terms of T** of the 64-bit adder – take the maximum of all output bits – including the sum bits (S_0-S_{63}) and the final carry out (C_{64}).

Maximal Delay: 52T ✓ *(2 points)* Good

. ***It's PAR-ty Time (30 points)***: Consider the single-cycle processor implementation from class. Your task will be to augment this datapath with a new instruction – the predicated add register (PAR) instruction. The PAR instruction is an R type instruction. The PAR instruction will have the following pseudocode:

```
if (R[RT] != 0)
  R[RD]=M[R[RS]]+SE(SA)
```

Note that this instruction uses the SA field – this is the shift amount field of the R type instruction. Instead of being a shift amount though, the SA field is used as a 5-bit increment to the value stored in memory at the location specified by the RS register, like a mini immediate. So SE(SA) means to sign extend the shift amount field from 5 bits to a full 32-bit value. The value of the RD register should only change if the value in the RT register is nonzero.

Implement your solution on the following two pages. All other instructions must still work correctly after your modifications. You should not add any new ALUs, register file ports, or ports to memory.

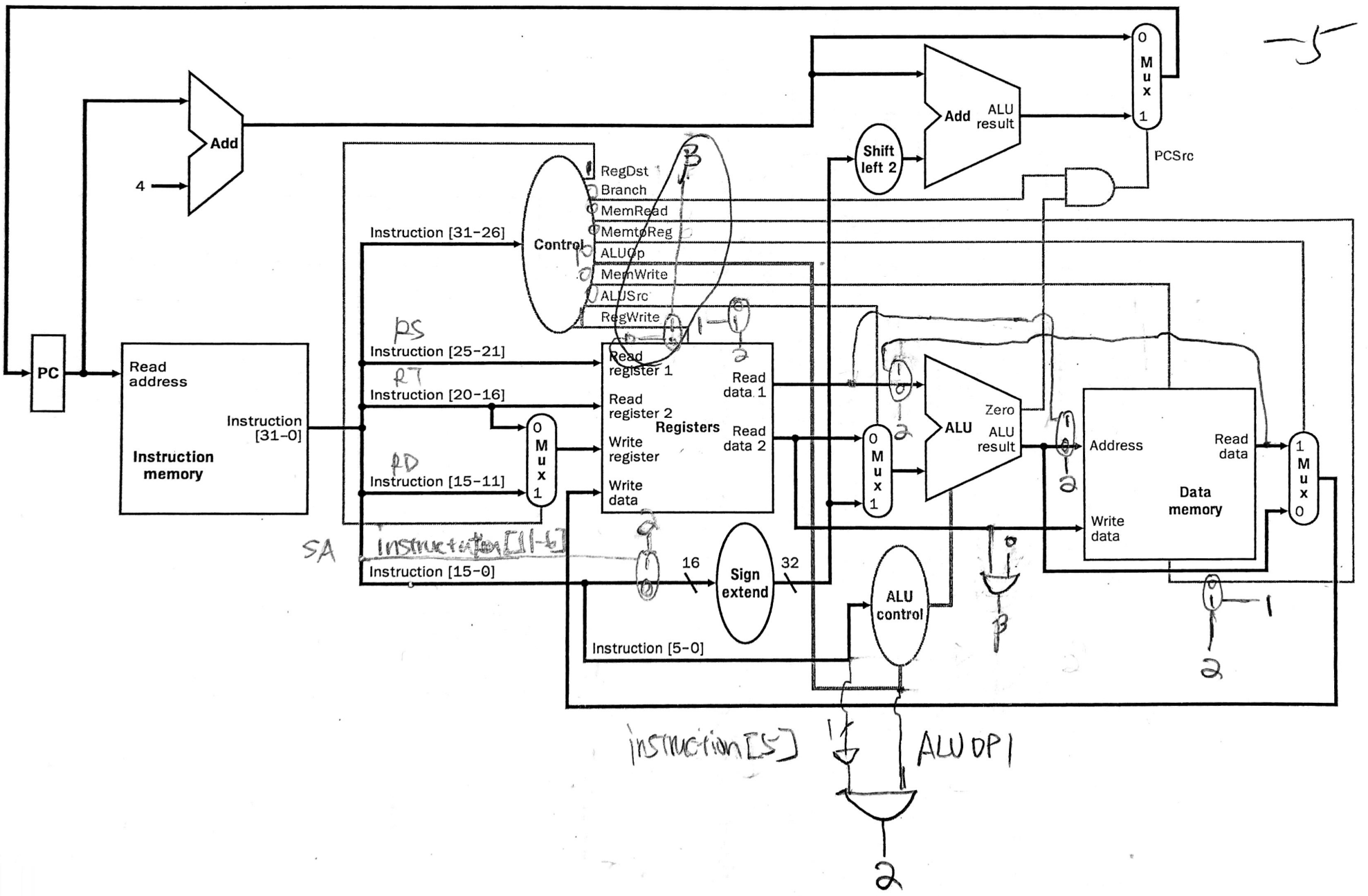
OP RS RT RD SA FUNC

6 5 5 5 5 6

31-26 25-21 20-16 15-11 10-6 5-0

if (R[RT] != 0)

R[RD] = M[R[RS]] + SE(SA)



Main Controller

nput or Output	Signal Name	R-format	lw	sw	Beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

ALU Controller

Opcode	**ALUOp**	**instruction**	**function**	**ALU Action**	**ALUCtrl**
Lw	00	load word	XXXXXX	add	010
Sw	00	store word	XXXXXX	add	010
Beq	01	branch equal	XXXXXX	subtract	110
R-type	10	add	100000	add	010
R-type	10	subtract	100010	subtract	110
R-type	10	AND	100100	AND	000
R-type	10	OR	100101	OR	001
R-type	10	SLT	101010	SLT	111

R-type 10 PAR 00|10 ADD 010