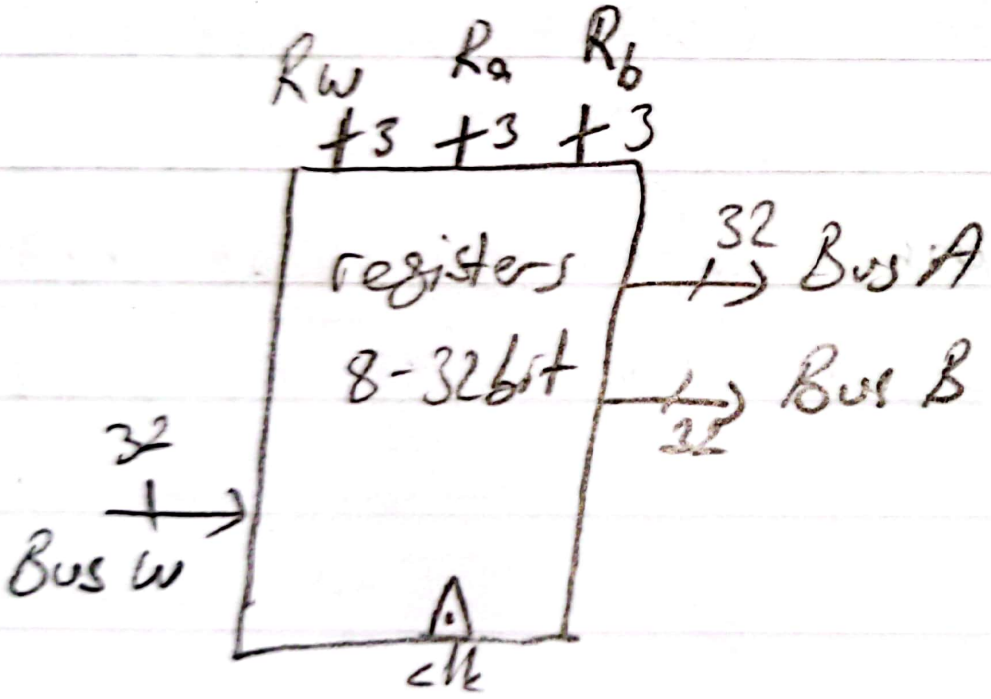
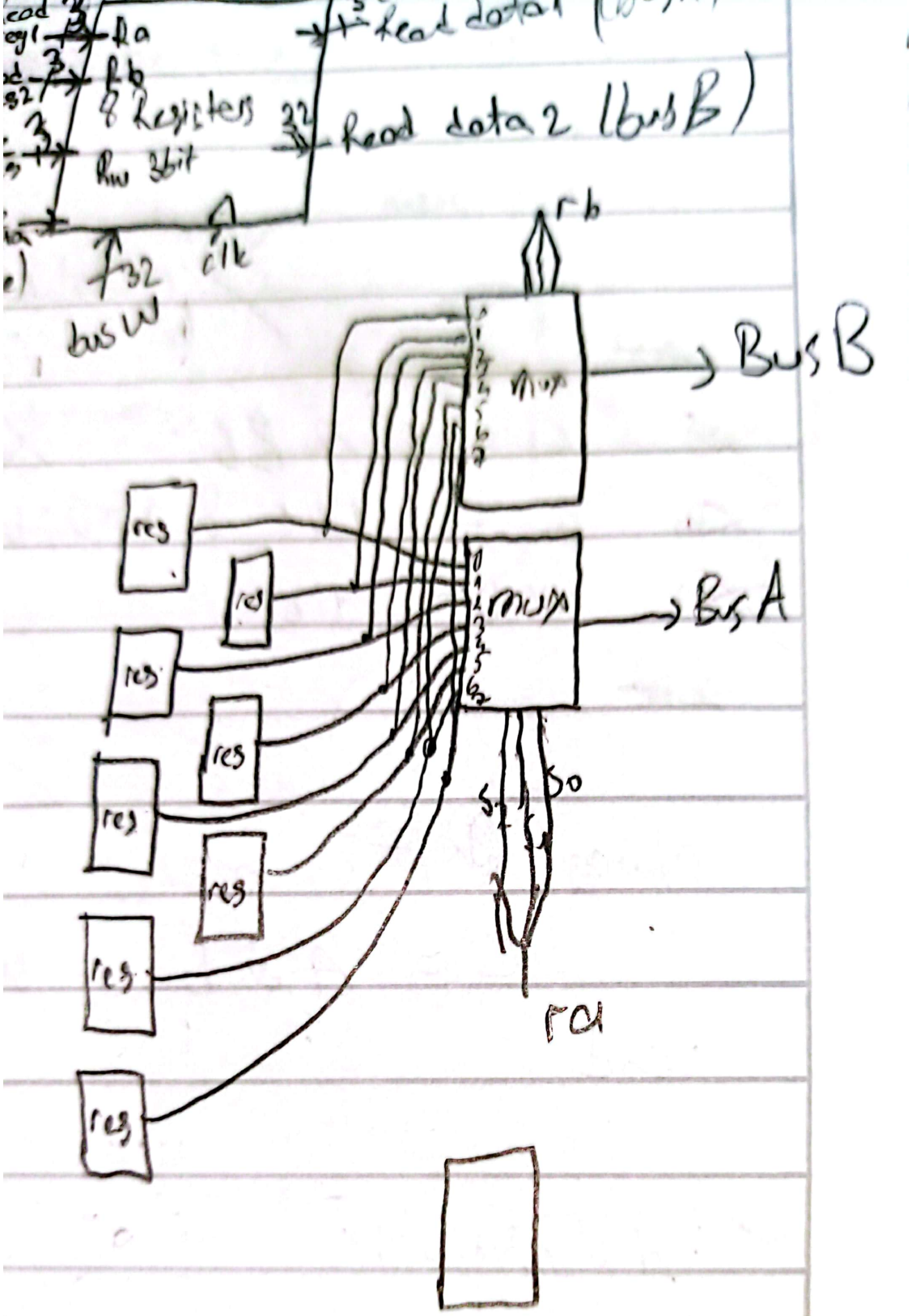
8 bit Register Design





I type Instructions

R [rt] = R [rs] (operation +-/*) SignExt Imm

ALUSrc = 1

memToReg = 0

RegDst = 0

RegWr = 1 (register enable)

MemWr = 0

branch = 0

ALUOp = operation (* - / +)

MemRd = 0

	Alu op	Alu OP (binary) $(P_2 P_1 P_0)$
R-type	2 (func'n bit)	010
addi	0 (add)	000
andi	3 (and	011
ori	4 (or)	100
nori	5 (nor)	101
slti	6 (slt)	110
beq	1 (sub)	001
bne	1 (sub)	001
lw	0 (add)	000
sw	0	000

$P_0 = bne + beq + nori + andi$

$P_1 = R\text{-type} + andi + slti$

$P_2 = ori + nori + slti$

ALU Control Unit

Alu (3bit

0 add 000
1 xor 001
2 sub
3 mult
4 slt
5 nor
6 and
7 or

		P_2	P_1	P_0	Func			Alu control			
		P_2	P_1	P_0	F_2	F_1	F_0	C_2	C_1	C_0	
	add 0	0	0	0	x	x	x	0	0	0	✓
	Sub 1	0	0	1	x	x	x	0	1	0	✓
	and 2	0	1	0	0	0	0	0	0	0	✓
	and 3	0	1	1	x	x	x	0	0	0	–
	or 4	1	0	0	x	x	x	1	1	1	
	nor 5	1	0	1	x	x	x	1	0	1	
	slt 6	1	1	0	x	x	x	1	0	0	–
	7	1	1	1	x	x	x				
	add 2	0	1	0	0	0	1	0	0	0	✓
	sub 2	0	1	0	0	1	0	0	1	0	✓
	xor 2	0	1	0	0	1	1	0	0	1	✓
	nor 2	0	1	0	1	0	0	1	0	1	✓
	or 2	0	1	0	1	0	1	1	0	1	✓

C_2

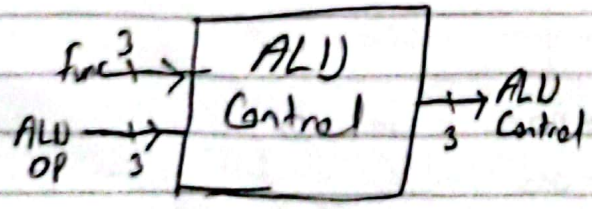
$$C_2 = P_2 + P_1 P_0' + P_1 F_2$$

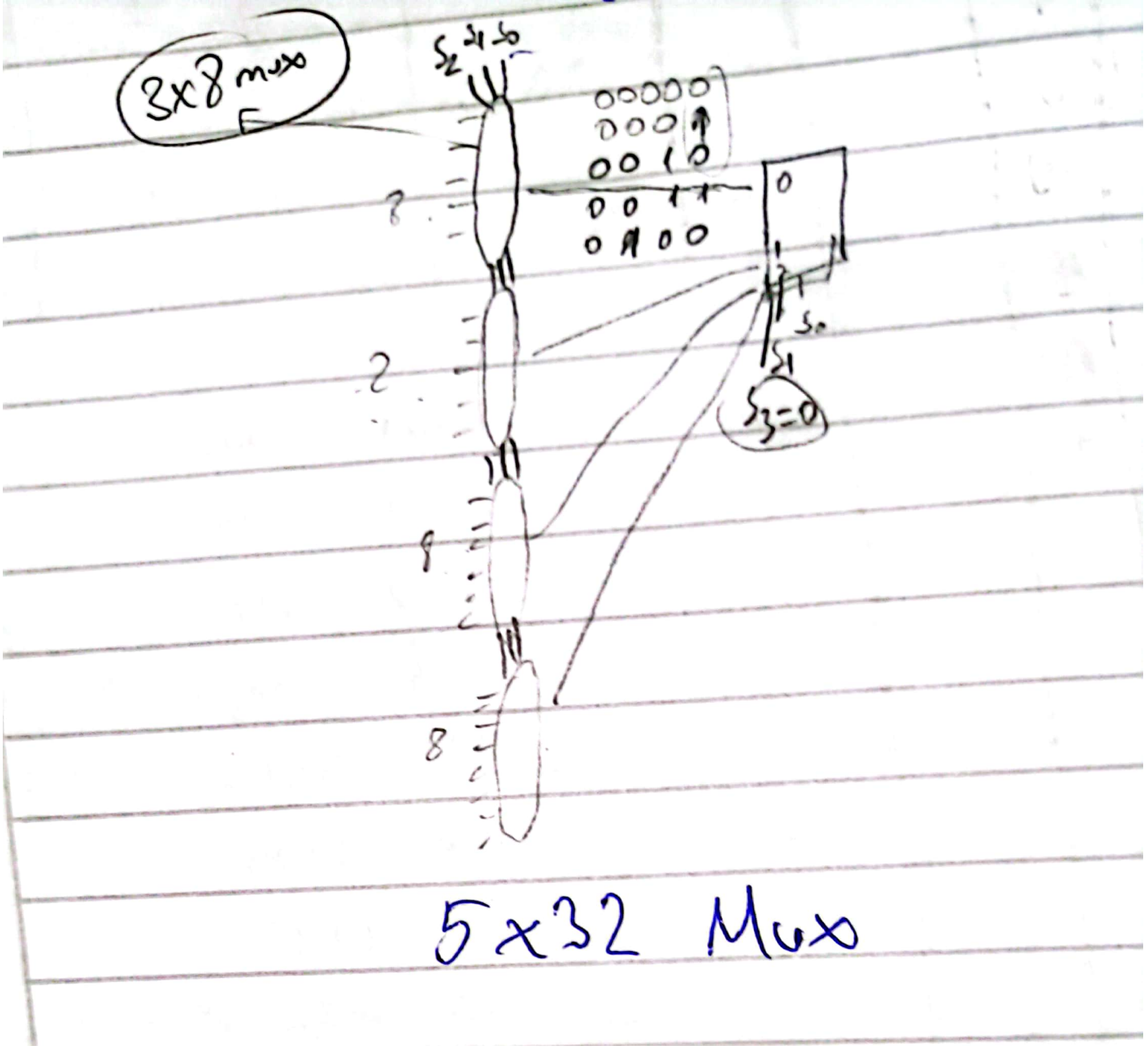
$$\boxed{C_2 = P_2 + P_1 (F_2 + P_0)}$$

$$C_1 = P_2' P_0 + P_2 P_1' P_0' + P_2' P_1 (F_2' F_1 F_0' + F_2 F_1' F_0)$$

$$\boxed{C_1 = P_2' P_0 + P_2 P_1' P_0' + P_2' P_1 (F_2' F_1 F_0' + F_2 F_1' F_0 + F_2' F_0')}$$

$$\boxed{C_0 = P_2 P_1' + P_2' P_1 P_0' (F_2 + F_1 F_0)}$$





5x32 Mux

110 (6) 010 (2) 100 (4) 011 (3) 111 (7)

100

		r0	r1	r2	r3	r4	r5	r6	r7
addi	$r_1 = r_0 + 3$		3						
addi	$r_2 = r_0 + 7$			7					
andi	$r_3 = r_1 \& 7$				3				
andi	$r_4 = r_2 \& 4$					4			
ori	$r_5 = r_4 \mid 7$						7		
ori	$r_6 = r_5 \mid 0$							7	
nori	$r_6 = r_6$ nor 4								
nori	$r_6 = r_6$ nor 4								
and	$r_5 = r_5 \& r_4$						4		
and	$r_5 = r_5 \& r_1$						0		
add	$r_5 = r_1 + r_2$						10		
add	$r_5 = r_4 + r_3$						4		
sub	$r_5 = r_5 - r_1$						1		
sub	$r_5 = r_5 - r_1$						1		
xor	$r_4 = r_4$ (xor) r_5					5			
xor	$r_4 = r_4$ (xor) r_5					4			
nor	$r_3 = r_3$ (nor) r_5			max-4					
nor	$r_2 = r_3$ (nor) r_2			0					
or	$r_2 = r_2 \mid r_1$			3					
or	$r_1 = r_2 \mid r_1$		3						
beq									
addi									
slti									
sw									
lw									

Instructions Plan for test bench

```
Rtypes
0000__A___B__OUT_FUNC
0000_101_100_101_000                    // and
0000_101_001_101_000                    // and
0000_001_010_101_001                    // add
0000_100_011_101_001                    // add
0000_101_001_101_010                    // sub
0000_101_011_101_010                    // sub

0000_100_101_100_011                    // xor
0000_100_101_100_011                    // xor
0000_011_101_011_100                    // nor
0000_011_010_010_100                    // nor
0000_010_001_010_101                    // or
0000_010_001_001_101                    // or



0000_001_010_011_000    // and -> r3 = r1 and r2
0000_001_010_011_001    // add -> r3 = r1 + r2
0000_001_010_011_010    // sub -> r3 = r1 - r2
0000_001_010_011_011    // xor -> r3 = r1 xor r2
0000_001_010_011_100    // nor -> r3 = r1 xor r2
0000_001_010_011_101    // or -> r3 = r1 xor r2



Itypes
_op___A__res_immed_
0001_000_001_000011     // addi -> r1 = r0 + 3
0001_000_010_000111     // addi -> r2 = r0 + 7
0010_001_011_000111     // andi
0010_010_100_000100     // andi
0011_100_101_000111     // ori
0011_101_110_000000     // ori
0100_110_110_000100     // nori
0100_110_110_000100     // nori
0111_000_001_000001     // slti

0001_001_001_000000
```

Beq

```
0101_001_001_000111          // r1==r1 -> pc=7
0101_001_001_000011          // r1==r1 -> pc=3
```

SW

```
1001_001_010_000001          // sw -> Mem[r1+imm(1)] = r2(7)
```

LW

```
1000_001_100_000001          // lw -> r4 = Mem[r1+imm(1)]
```