Instruction Set Architecture

Class	Instruction	Usage	Meaning
Arithmetic	Add	add rs,rt	$rs \leftarrow (rs) + (rt)$
	Comp	comp rs,rt	$rs \leftarrow$ 2's Complement (rs)
	Add immediate	addi rs,imm	$rs \leftarrow (rs) + imm$
	Complement Immediate	compi rs,imm	$rs \leftarrow$ 2's Complement (imm)
Logic	AND	and rs,rt	$rs \leftarrow (rs) \wedge (rt)$
	XOR	xor rs,rt	$rs \leftarrow (rs) \oplus (rt)$
Shift	Shift left logical	shll rs, sh	$rs \leftarrow (rs)$ left-shifted by sh
	Shift right logical	shrl rs, sh	$rs \leftarrow (rs)$ right-shifted by sh
	Shift left logical variable	shllv rs, rt	$rs \leftarrow (rs)$ left-shifted by (rt)
	Shift right logical	shrl rs, rt	$rs \leftarrow (rs)$ right-shifted by (rt)
	Shift right arithmetic	shra rs, sh	$rs \leftarrow (rs)$ arithmetic right-shifted by sh
	Shift right arithmetic variable	shrav rs, rt	$rs \leftarrow (rs)$ right-shifted by (rt)
Memory	Load Word	lw rt,imm(rs)	$rt \leftarrow mem[(rs) + imm]$
	Store Word	sw rt,imm,(rs)	$mem[(rs) + imm] \leftarrow (rt)$
Branch	Unconditional branch	b L	goto L
	Branch Register	br rs	goto (rs)
	Branch on less than 0	bltz rs,L	if$(rs) < 0$ then goto L
	Branch on flag zero	bz rs,L	if $(rs) = 0$ then goto L
	Branch on flag not zero	bnz rs,L	if$(rs) \neq 0$ then goto L
	Branch and link	bl L	goto L; 31 $\leftarrow$ (PC)+4
	Branch on Carry	bcy L	goto L if Carry = 1
	Branch on No Carry	bncy L	goto L if Carry = 0

Instruction Formats:

1. R-Format

   Instruction : add, comp, and, xor, shllv, shrlv, shrav

   No. of functions available : 16

   No. of Functions currently used : 7

Opcode(3)	rs(5)	rt(5)	Don't care(15)	funct(4)

Opcode	Funct(decimal)	Instruction
000	0	add
	1	comp
	2	and

	3	xor
	4	shllv
	5	shrlv
	6	shrav

2. I-Format

Instruction : addi, compi, shll, shrl, shra
No. of functions available : 16
No. of Functions currently used : 5

Opcode(3)	rs(5)	Immediate(15)	Don’t Care(5)	funct(4)

Opcode	Funct(decimal)	Instruction
001	1	addi
	2	compi
	3	shll
	4	shrl
	5	shra

3. LS-Format

Instructions : lw, sw
No. of functions available : 16
No. of Functions currently used : 2

Opcode(3)	rs(5)	rt(5)	Immediate(15)	funct(4)

Opcode	Funct(decimal)	Instruction
010	0	lw
	1	sw

4. B1-Format

Instructions : b, bl, bcy, bncy

No. of functions available : 16
No. of Functions currently used : 4

Opcode(3)	Immediate/Label(15)	Don’t Care(10)	funct(4)

Opcode	Funct(decimal)	Instruction
011	0	b
	1	bl
	2	bcy
	3	bncy

5. B2-Format

Instructions : br
No. of functions available : 16
No. of Functions currently used : 1

Opcode(3)	rs(5)	Don’t care(20)	funct(4)

Opcode	Funct(decimal)	Instruction
100	0	br

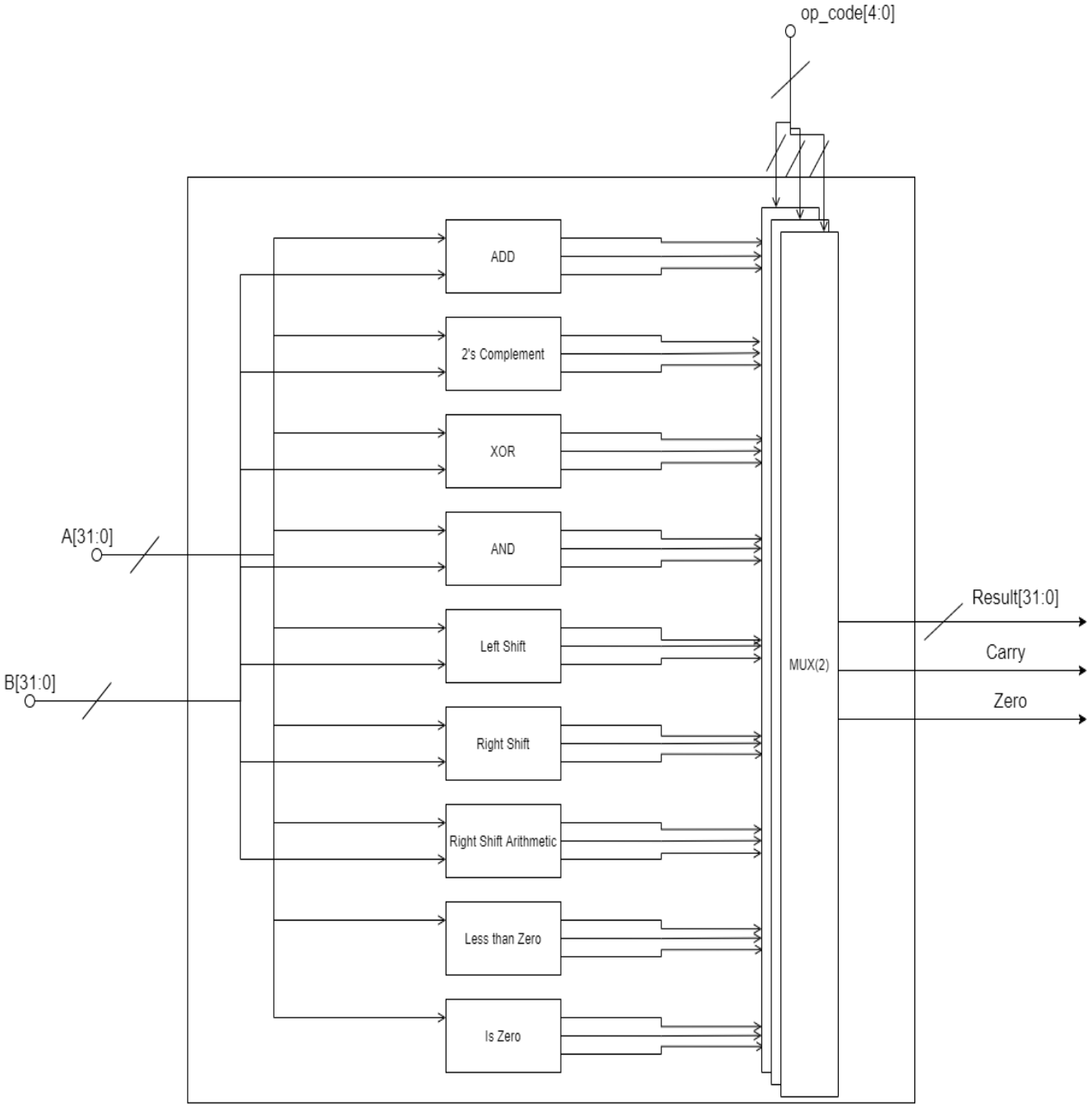
6. B3-Format

Instructions : bltz, bz, bnz
No. of functions available : 16
No. of Functions currently used : 3

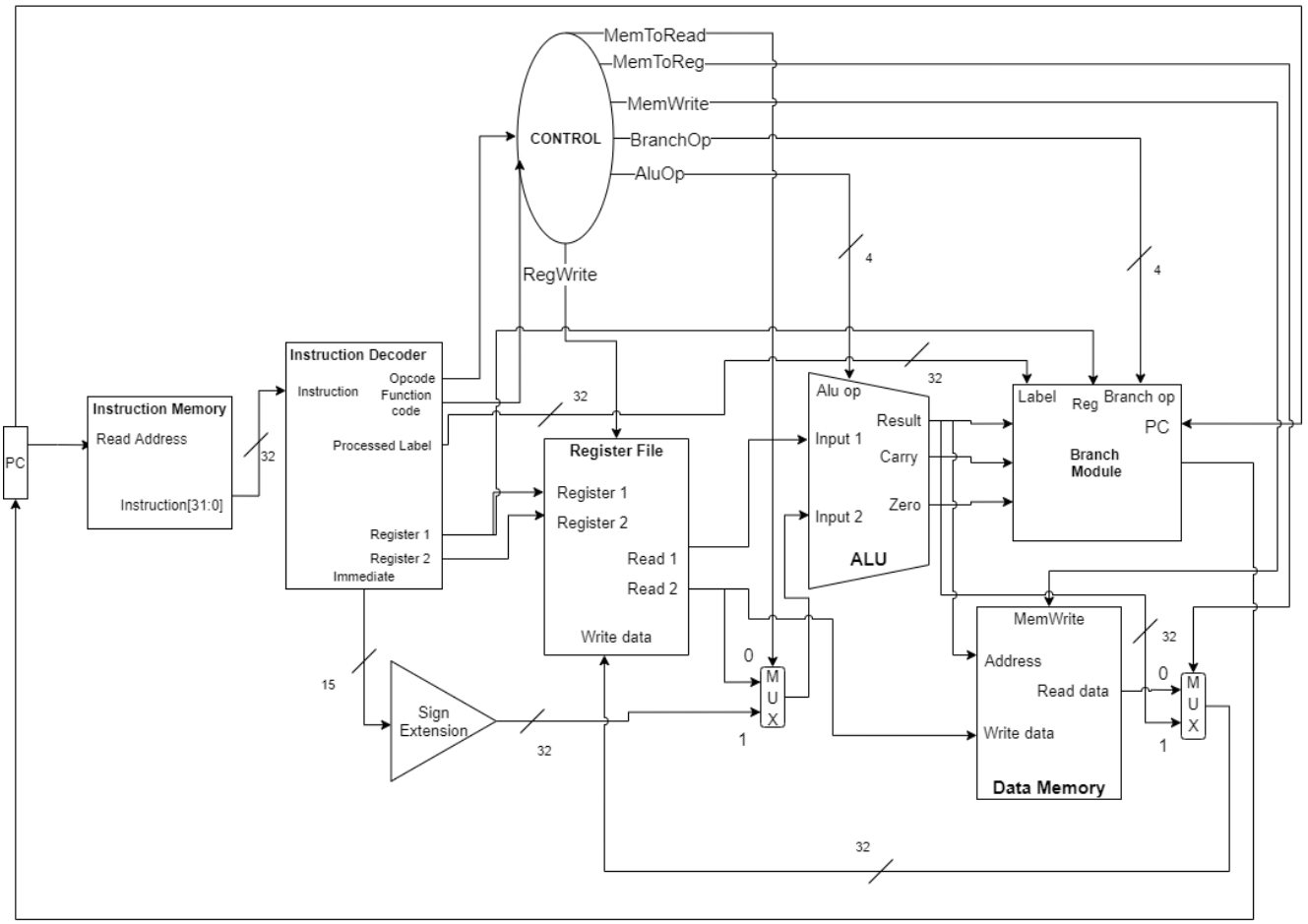
Opcode(3)	rs(5)	Immediate/Label(15)	Don’t Care(5)	funct(4)

Opcode	Funct(decimal)	Instruction
101	0	bltz
	1	bz
	2	bnz

ALU Design


op_code[4:0]
ADD
2's Complement
XOR
AND
Left Shift
Right Shift
Right Shift Arithmetic
Less than Zero
Is Zero
MUX(2)
A[31:0]
B[31:0]
Result[31:0]
Carry
Zero

Data Path Design



Control Unit:

Instruction	OP code	Func code	RegWrite	MemToRead	MemToReg	MemWrite	BranchOp	AluOp
add	000	0000	1	0	1	0	0000	0000
comp	000	0001	1	0	1	0	0000	0001
and	000	0010	1	0	1	0	0000	0011
xor	000	0011	1	0	1	0	0000	0010
shllv	000	0100	1	0	1	0	0000	0100
shrlv	000	0101	1	0	1	0	0000	0101
shrav	000	0110	1	0	1	0	0000	0110
addi	001	0001	1	1	1	0	0000	0000
compi	001	0010	1	1	1	0	0000	0001
shll	001	0011	1	1	1	0	0000	0100
shrl	001	0100	1	1	1	0	0000	0101
shra	001	0101	1	1	1	0	0000	0110
lw	010	0000	1	1	0	0	0000	0000
sw	010	0001	0	1	X	1	0000	0000
b	011	0000	0	X	X	0	0001	XXXX
bl	011	0001	0	X	X	0	0010	XXXX
bcy	011	0010	0	X	X	0	0011	XXXX
bncy	011	0011	0	X	X	0	0100	XXXX
br	100	0000	0	X	X	0	0101	XXXX
bltz	101	0000	0	0	X	0	0110	0111
bz	101	0001	0	0	X	0	0111	1000
bnz	101	0010	0	0	X	0	1000	1000

Branch Module

Inputs:

- Result, Carry and Zero from the ALU unit
- Label and Reg from the Instruction Decoder Unit
- BranchOp code from the control unit

Outputs: The next Program Counter (PC)

The Branch module receives inputs from various units and processes them according to the BranchOp code that is given by the controller. The Module’s main purpose is to set the program counter (PC) to the correct value after every instruction. For example, Upon receiving BranchOP code as 0000 the module simply adds 4 to the PC.