Computer Organization and Architecture Laboratory
KGP-RISC

Group 30
Vanshita Garg - 19CS10064
Ashutosh Kumar Singh - 19CS30008

1 Instruction Set Architecture

Class	Instruction	Usage	Meaning	Opcode	Function Code
Arithmetic	Add	add rs, rt	$rs \leftarrow (rs) + (rt)$	000000	00000
	Complement	comp rs, rt	$rs \leftarrow$ 2's Complement of (rt)	000000	00001
Logic	AND	and rs, rt	$rs \leftarrow rs \wedge rt$	000001	00000
	XOR	xor rs, rt	$rs \leftarrow rs \oplus rt$	000001	00001
Shift	Shift Left Logical	shll rs, sh	rs left-shifted by sh	000010	00000
	Shift Right Logical	shrl rs, sh	rs right-shifted by sh	000010	00001
	Shift Left Logical Variable	shllv rs, rt	rs left-shifted by rt	000010	00010
	Shift Right Logical Variable	shrlv rs, rt	rs right-shifted by rt	000010	00011
	Shift Right Arithmetic	shra rs, sh	rs arithmetic right-shifted by sh	000010	00100
	Shift Right Arithmetic Variable	shrav rs, rt	rs arithmetic right-shifted by rt	000010	00101
Arithmetic Immediate	Add Immediate	addi rs, imm	$rs \leftarrow (rs) + imm$	000011	NA
	Complement Immediate	compi rs, imm	$rs \leftarrow$ 2's Complement of imm	000100	NA
Memory	Load Word	lw rt, imm(rs)	$rt \leftarrow mem[(rs) + imm]$	000101	NA
	Store Word	sw rt, imm(rs)	$mem[(rs) + imm] \leftarrow (rt)$	000110	NA
Branch	Branch on less than zero	bltz rs, L	if $(rs) < 0$ then goto L	000111	NA
	Branch on flag zero	bz rs, L	if $(rs) = 0$ then goto L	001000	NA
	Branch on flag not zero	bnz rs, L	if $(rs) \neq 0$ then goto L	001001	NA
	Branch Register	br rs	goto (rs)	001010	NA
	Unconditional Branch	b L	goto L	001011	NA
	Branch and Link	bl L	goto L; $\$ra \leftarrow$ (PC) + 4	001100	NA
	Branch on Carry	bcy L	if goto L if Carry = 1	001101	NA
	Branch on no Carry	bncy L	if goto L if Carry = 0	001110	NA

2 Instruction Format and Encoding

2.1 R-Format Instructions

opcode	rs	rt	shamt	Don't care	func
6 bits	5 bits	5 bits	5 bits	6 bits	5 bits

Instruction	opcode	func
add	000000	00000
comp		00001
and	000001	00000
xor		00001
shll	000010	00000
shrl		00001
shllv		00010
shrlv		00011
shra		00100
shrav		00101

2.2 I-Format Instructions

opcode	rs	Don't care	imm
6 bits	5 bits	5 bits	16 bits

Instruction	opcode
addi	000011
compi	000100

2.3 Memory Access Instructions

opcode	rs	rt	imm
6 bits	5 bits	5 bits	16 bits

Instruction	opcode
lw	000101
sw	000110

2.4 J1-Format Instructions (Conditional Jump to Label Depending on Register Value)

opcode	rs	Don't care	L
6 bits	5 bits	5 bits	16 bits

Instruction	opcode
bltz	000111
bz	001000
bnz	001001

2.5 J2-Format Instruction(s) (Unconditional Jump to Address at Register Content)

opcode	rs	Don't care
6 bits	5 bits	21 bits

Instruction	opcode
br	001010

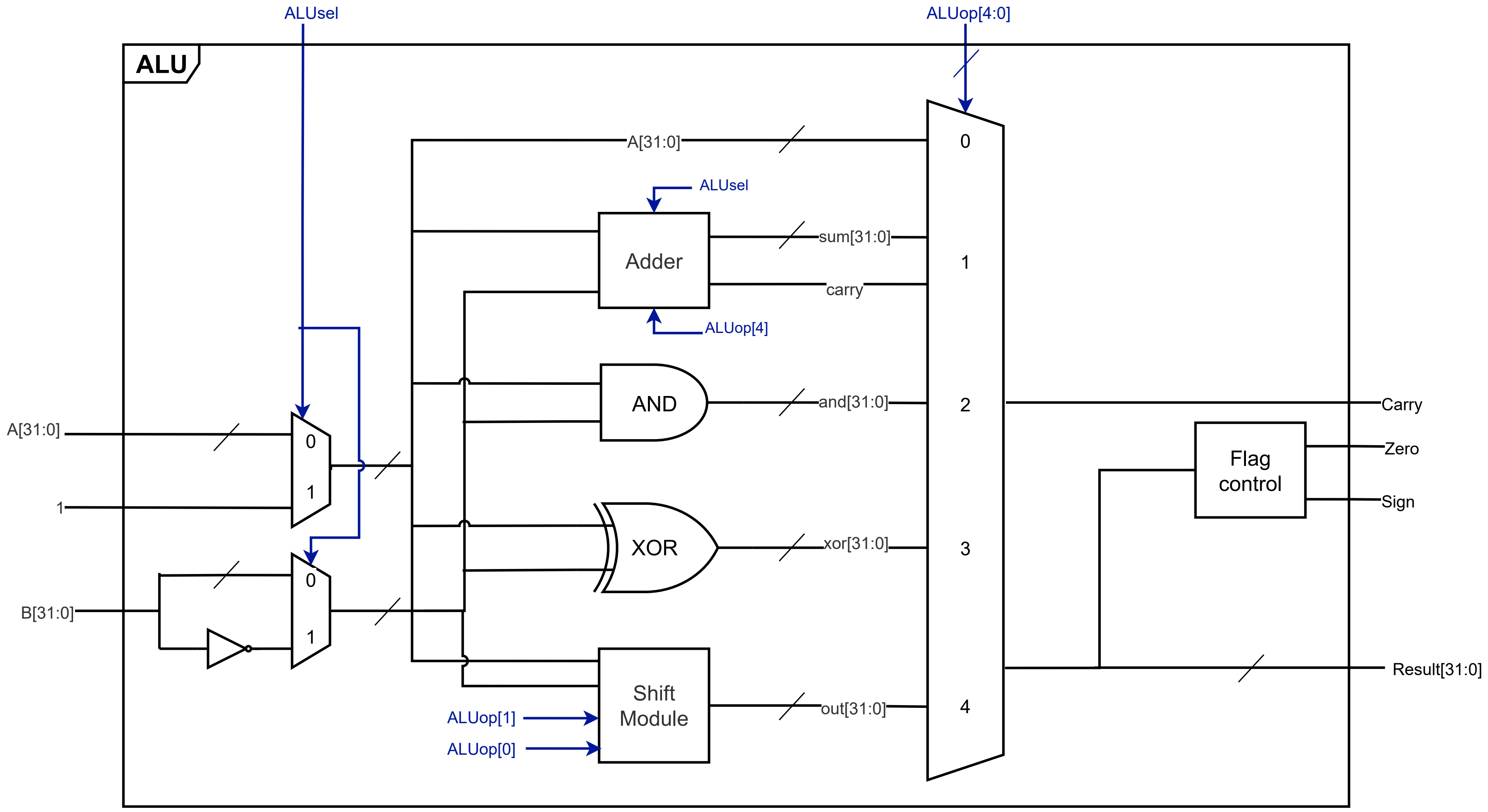
2.6 J3-Format Instructions (Jump to Label Unconditionally or Based on Flags)

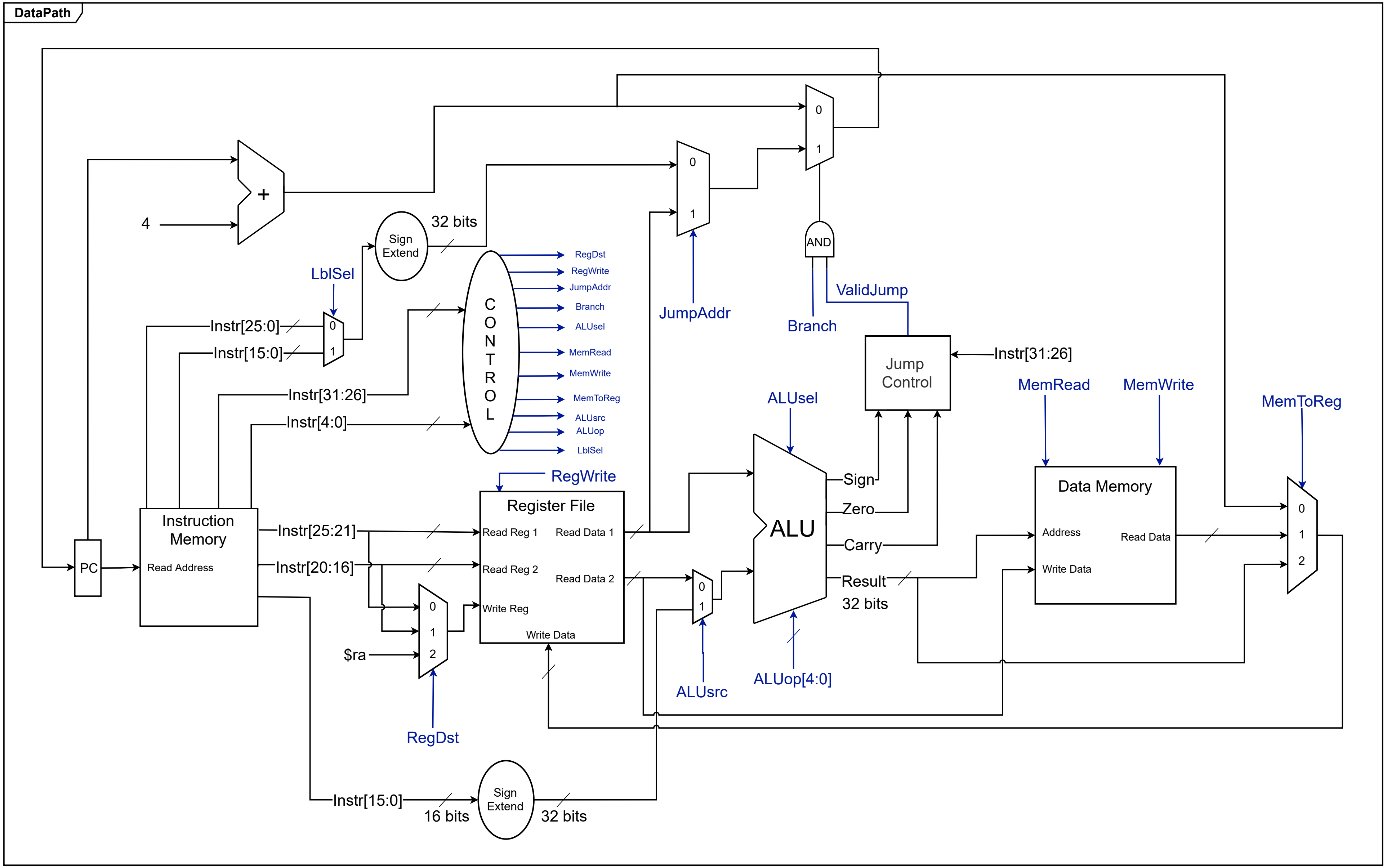
opcode	L
6 bits	26 bits

Instruction	opcode
b	001011
bl	001100
bcy	001101
bncy	001110

3 Register Usage Convention

Register	Function	Register Number	Register Code
$zero	Zero Register, always stores the constant value 0	0	00000
$pc	Program Counter	1	00001
$v0 - $v1	Return values from functions	2 - 3	00010 - 00011
$a0 - $a3	Arguments for function calls	4 - 7	00100 - 00111
$t0 - $t11	Temporaries (not preserved across function calls)	8 - 19	01000 - 10011
$s0 - $s9	Saved variables (preserved across function calls)	20 - 29	10100 - 11101
$sp	Stack Pointer	30	11110
$ra	Return Address (used during function calls)	31	11111


ALUsel
ALUop[4:0]
ALU
A[31:0]
0
ALUsel
Adder
sum[31:0]
1
carry
ALUop[4]
AND
and[31:0]
2
Carry
A[31:0]
0
1
1
Flag
control
Zero
Sign
XOR
xor[31:0]
3
B[31:0]
0
1
Shift
Module
out[31:0]
4
Result[31:0]
ALUop[1]
ALUop[0]


DataPath
+
4
Sign Extend
32 bits
LblSel
Instr[25:0]
Instr[15:0]
Instr[31:26]
Instr[4:0]
C O N T R O L
RegDst
RegWrite
JumpAddr
Branch
ALUsel
MemRead
MemWrite
MemToReg
ALUsrc
ALUop
LblSel
JumpAddr
AND
Branch
ValidJump
Jump Control
Instr[31:26]
ALUsel
MemRead
MemWrite
MemToReg
PC
Instruction Memory
Read Address
Instr[25:21]
Instr[20:16]
$ra
RegDst
RegWrite
Register File
Read Reg 1
Read Reg 2
Write Reg
Write Data
Read Data 1
Read Data 2
ALUsrc
ALU
ALUop[4:0]
Sign
Zero
Carry
Result
32 bits
Data Memory
Address
Write Data
Read Data
Instr[15:0]
16 bits
Sign Extend
32 bits

4 Truth Table for Control Signals

Instr	opcode	func	Reg Dst	Reg Write	Mem Read	Mem Write	MemTo Reg	ALU src	ALUop	ALU sel	Branch	Jump Addr	Lbl Sel
add	000000	00000	00	1	0	0	10	0	00001	0	0	X	X
comp	000000	00001	00	1	0	0	10	0	00101	1	0	X	X
and	000001	00000	00	1	0	0	10	0	00010	0	0	X	X
xor	000001	00001	00	1	0	0	10	0	00011	0	0	X	X
shll	000010	00000	00	1	0	0	10	1	01010	0	0	X	X
shrl	000010	00001	00	1	0	0	10	1	01000	0	0	X	X
shllv	000010	00010	00	1	0	0	10	0	01010	0	0	X	X
shrlv	000010	00011	00	1	0	0	10	0	01000	0	0	X	X
shra	000010	00100	00	1	0	0	10	1	01001	0	0	X	X
shrav	000010	00101	00	1	0	0	10	0	01001	0	0	X	X
addi	000011	NA	00	1	0	0	10	1	00001	0	0	X	X
compi	000100	NA	00	1	0	0	10	1	00101	1	0	X	X
lw	000101	NA	01	1	1	0	01	1	10101	0	0	X	X
sw	000110	NA	X	0	0	1	X	1	10101	0	0	X	X
bltz	000111	NA	X	0	0	0	X	X	00000	X	1	0	1
bz	001000	NA	X	0	0	0	X	X	00000	X	1	0	1
bnz	001001	NA	X	0	0	0	X	X	00000	X	1	0	1
br	001010	NA	X	0	0	0	X	X	00000	X	1	1	X
b	001011	NA	X	0	0	0	X	X	00000	X	1	0	0
bl	001100	NA	10	1	0	0	00	X	00000	X	1	0	0
bcy	001101	NA	X	0	0	0	X	X	00000	X	1	0	0
bncy	001110	NA	X	0	0	0	X	X	00000	X	1	0	0

Description of Control Signals

- **RegDst** - Determines the register to which data will be written. Used to choose between rs, rt and $\$ra$.
- **RegWrite** - Determines whether any data should be written to a register or not.
- **MemRead** - Determines whether any data should be read from the data memory or not.
- **MemWrite** - Determines whether anything should be written to the data memory or not.
- **MemToReg** - Selects the appropriate line to write to the register file. Used to choose between the ALU result (for an R-type instruction), data from the data memory (for a sw instruction), or (PC) + 4 (for a bl instruction).
- **ALUsrc** - Controls the second input of the ALU. Used to choose between the contents of a register (for R-type instructions), or the sign-extended result (for immediate operations).
- **ALUop** - Determines the type of operation to be carried out in the ALU. ALUop[4] distinguishes between a memory and R-type instruction, ALUop[3] is 1 only for a shift instruction, ALUop[2] is 1 only for a complement instruction, ALUop[1] indicates direction for shift operations (0 for right shift, 1 for left shift), and ALUop[0] indicates type for shift operations (0 for logical shift, 1 for arithmetic shift).
- **ALUsel** - Control signal to distinguish between any other ALU operation and a 2's complement operation.
- **Branch** - Determines whether an instruction is a branch instruction or not.
- **JumpAddr** - Controls the kind of label (address) to be selected during a jump operation. It is 1 only for the br instruction, where the address is the content of a register, otherwise it is 0.
- **LblSel** - Controls the length of the label (address) selected for the branch operation. It is 1 for the instructions with a 16-bit label (bltz, bz, bnz), and 0 for the instructions with a 26-bit label (b, bl, bcy, bncy).

5 Truth Table for Jump Control

Instr	opcode	Zero	Sign	Carry	ValidJump
bltz	000111	0	1	X	1
bz	001000	1	0	X	1
bnz	001001	0	X	X	1
br	001010	X	X	X	1
b	001011	X	X	X	1
bl	001100	X	X	X	1
bcy	001101	X	X	1	1
bncy	001110	X	X	0	1

All other input combinations lead to ValidJump = 0.

The ValidJump signal is always 1 in case of an unconditional branch instruction (like br, b, bl). For conditional jumps based on some flags or the contents of a register it is 1 only when the condition is satisfied, otherwise it is 0.