INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMPUTER ORGANIZATION LABORATORY
(CS39001)

Abhay Kumar Keshari (20CS10001)
Hardik Pravin Soni (20CS30023)

November 9, 2022

Contents

1 Instruction Set Architecture

Table 1: 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(rt)
	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	$rs \leftarrow rs \& rt$
	XOR	xor rs,rt	$rs \leftarrow rs \oplus rt$
Shift	Shift Left Logical	shll rs, sh	$rs \leftarrow (rt)$ left-shifted by sh
	Shift Right Logical	shrl rs,sh	$rs \leftarrow (rt)$ right-shifted by sh
	Shift Left Logical Variable	shllv rs, rt	$rs \leftarrow (rt)$ left-shifted by (rt)
	Shift Right Logical Variable	shrlv rs, rt	$rs \leftarrow (rt)$ right-shifted by (rt)
	Shift Right Arithmetic	shra rs, sh	$rs \leftarrow (rs)$ arithmetic right-shifted by sh
	Shift Right Arithmatic Variable	shrav rs, sh	$rs \leftarrow (rs)$ arithmetic 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*
Complex	Diff	diff rs, rt	$rs \leftarrow$ *the LSB bit at which rs and rt differ*

2 Instruction Format and Encoding

2.1 R-Format Instruction:

add, comp, and, xor, shll, shllv, shrl, shrlv, shra, shrav, diff

Table 2: R-Format Encoding

opcode	rs	rt	shamt	Don't Care	funccode
6 bits	5bits	5 bits	5 bits	6 bits	5 bits

2.2 I-Format Instruction: lw, sw (Memory Access Instructions)

Table 3: I-Format Encoding for lw and sw

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

2.3 I-format: addi, compi

Table 4: I-Format Encoding for addi and compi

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

2.4 J1-Format:- bltz, bz, bnz, br

Table 5: J1-Format Encoding

opcode	rs	Don't Care	label
6 bits	5 bits	5 bits	16 bits

2.5 J2-Format:- b,bl,bcy,bncy

Table 6: J2-Format Encoding

opcode	L
6 bits	26 bits

3 Control Signals

Table 7: Control Signal Configuration

Instruction	op-code	func-code	RegDst	RegWrite	MemToReg	MemRead	MemWrite	ALUSel	ALUsrc	ALUop	LblSel	BranchAddrSel	isBranch
add	000000	00000	00	1	10	0	0	0	0	00001	x	x	0
comp	000000	00001	00	1	10	0	0	1	0	00101	x	x	0
AND	000001	00000	00	1	10	0	0	0	0	00010	x	x	0
XOR	000001	00001	00	1	10	0	0	0	0	00011	x	x	0
shll	000010	00000	00	1	10	0	0	0	1	01010	x	x	0
shllv	000010	00010	00	1	10	0	0	0	0	01010	x	x	0
shrl	000010	00001	00	1	10	0	0	0	1	01000	x	x	0
shrlv	000010	00011	00	1	10	0	0	0	0	01000	x	x	0
shra	000010	00100	00	1	10	0	0	0	1	01001	x	x	0
shrav	000010	00101	00	1	10	0	0	0	0	01001	x	x	0
lw	000101	xxxxx	01	1	01	1	0	0	1	10101	x	x	0
sw	000110	xxxxx	xx	0	xx	1	1	0	1	10101	x	x	0
b	001011	xxxxx	xx	0	xx	0	0	X	x	00000	0	0	1
bl	001100	xxxxx	10	1	00	0	0	X	x	00000	0	0	1
bcy	001101	xxxxx	xx	0	xx	0	0	X	x	00000	0	0	1
bncy	001110	xxxxx	xx	0	xx	0	0	X	x	00000	0	0	1
br	001010	xxxxx	xx	0	xx	0	0	X	x	00000	x	1	1
bltz	000111	xxxxx	xx	0	xx	0	0	X	x	00000	1	0	1
bz	001000	xxxxx	xx	0	xx	0	0	X	x	00000	1	0	1
bnz	001001	xxxxx	xx	0	xx	0	0	X	x	00000	1	0	1
addi	000011	xxxxx	00	1	10	0	0	0	1	00001	x	x	0
compi	000100	xxxxx	00	1	10	0	0	1	1	00101	x	x	0
diff	001111	00000	00	1	10	0	0	0	0	00100	x	x	0

3.1 Description of Control Signals for KGP-RISC

- **RegDst:-** Flag Bits to make the choice for the register to which data will be written. Used to make the choice between rs, rt and $ra.
- **RegWrite:-** The Flag that controls whether or NOT data is to be written to a register.
- **MemToReg:-** Selects the suitable line to write to the register file. Used to make a choice between (PC) + 4 (for a bl instruction) or the ALU result (for an R-type instruction), data from the data memory (for a sw instruction)
- **MemRead:-** The Flag that controls whether or not data is to be read from a given data-Memory address or NOT.
- **MemWrite:-** The Flag that controls whether or not data is to be written to a given data-memory address or NOT.

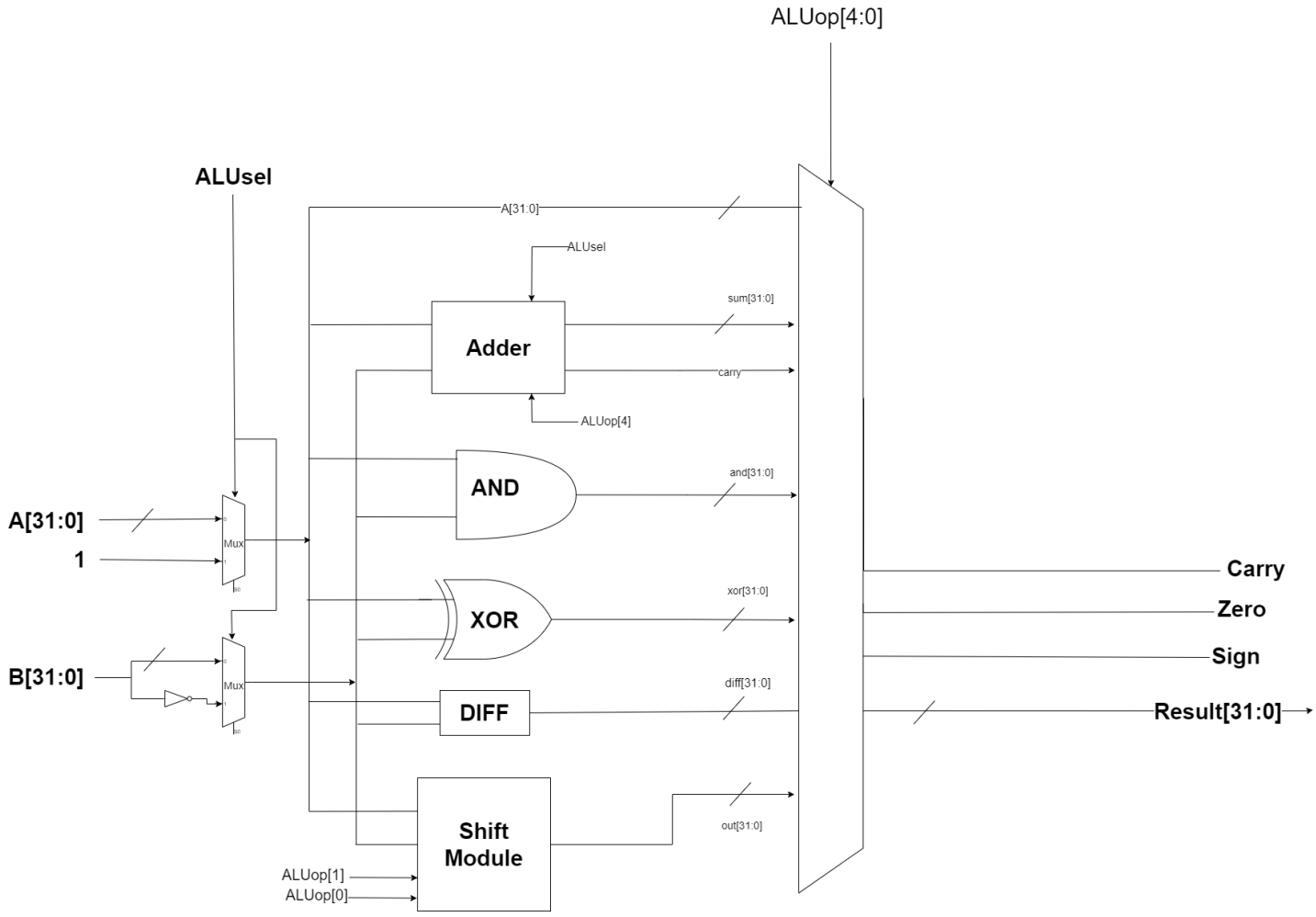

Figure 1: ALU for KGP-RISC

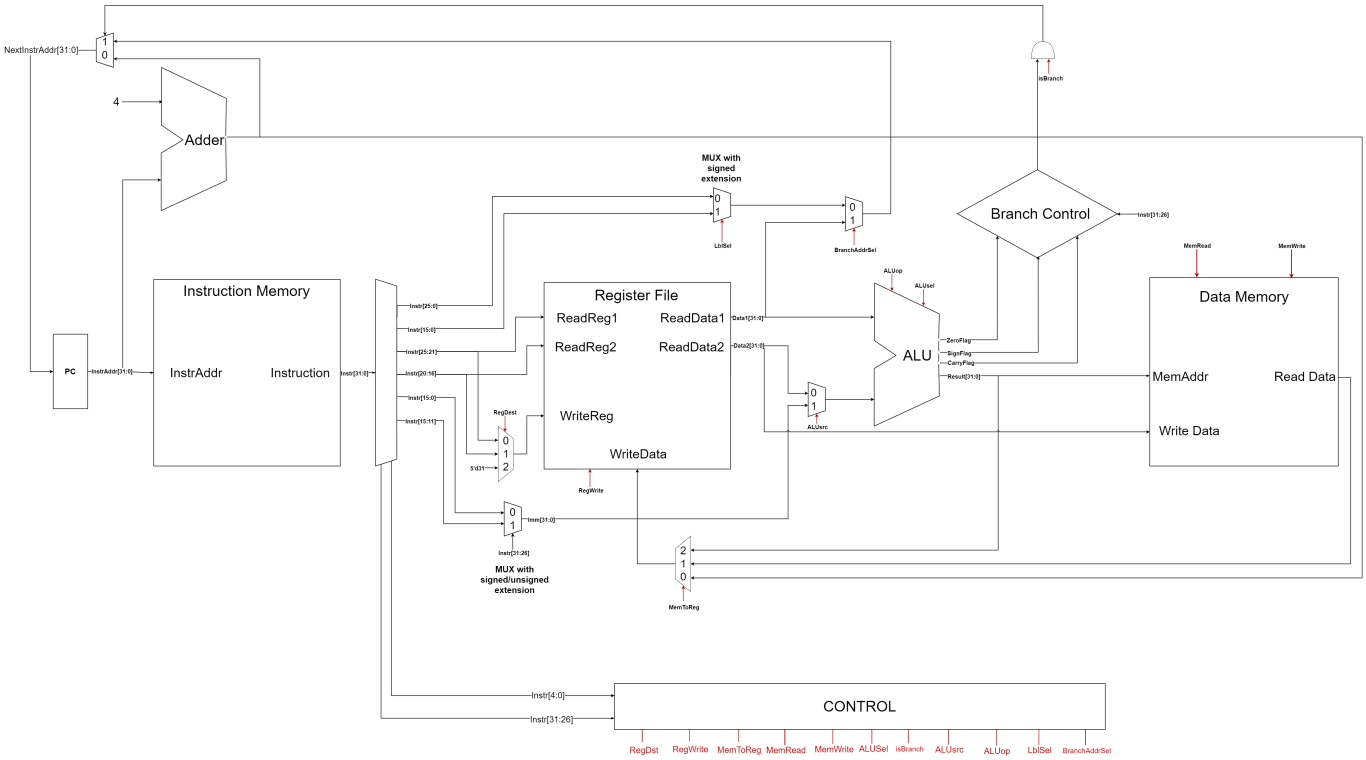

Figure 2: Datapath for KGP-RISC

- **ALUsrc:-** The Flag bit to control the choice between choose between the sign-extended result (for immediate operations) or the contents of a register (for R-type instructions), this acts as the second input to the ALU.
- **ALUop:-** Determines the type of operation to be carried out in the ALU.
- **LblSel:-** Controls the length of the label (address) selected for the branch operation. It is 0 for the instructions with a 26-bit label (b, bl, bcy, bncy) and 1 for the instructions with a 16-bit label (bltz, bz, bnz), and
- **BranchAddrSel:-** 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.
- **isBranch:-** Determines whether an instruction is a branch instruction or not(a valid jump is performed or NOT).

Table 8: Instructions in Detail

Instruction	**ALUop**	a	b	**result**	**Carry**	**Sign**	**Zero**
add	add a, b	R[rs]	R[rt]	$a + b$	c_{out}	X	X
comp	comp b	R[rs]	R[rt]	$\neg b + 1$	no change	X	X
AND	and a, b	R[rs]	R[rt]	$a \wedge b$	no change	X	X
XOR	xor a, b	R[rs]	R[rt]	$a \oplus b$	no change	X	X
diff	diff a, b	R[rs]	R[rt]	$(a \oplus b) \wedge (-(a \oplus b))$	no change	X	X
shll	sll a, b	R[rs]	imm	$a \ll b$	no change	X	X
shllv	sll a, b	R[rs]	R[rt]	$a \ll b$	no change	X	X
shrl	srl a, b	R[rs]	imm	$a \gg b$	no change	X	X
shrlv	srl a, b	R[rs]	R[rt]	$a \gg b$	no change	X	X
shra	sra a, b	R[rs]	imm	{b*(a[31]),$a \gg b$}	no change	X	X
shrav	sra a, b	R[rs]	R[rt]	{b*(a[31]),$a \gg b$}	no change	X	x
lw	add a, b	R[rs]	imm	a+b	no change	X	X
sw	add a, b	R[rs]	imm	a+b	no change	X	X
b	none	X	X	X	no change	X	X
bl	none	X	X	X	no change	X	X
bcy	none	X	X	X	no change	X	X
bncy	none	X	X	X	no change	X	X
br	none	R[rs]	X	a	no change	X	X
bltz	update(sign)	R[rs]	X	a	no change	a[31]	X
bz	update(zero)	R[rs]	X	a	no change	X	a==0
bnz	update(zero)	R[rs]	X	a	no change	X	a==0
addi	add a, b	R[rs]	imm	a+b	c_{out}	X	X
compi	comp b	R[rs]	imm	$\neg b + 1$	no change	X	X

3.2 Branch Unit

All other input combinations lead to isBranch = 0. The isBranch 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.

Table 9: ALUop for Each Instruction

Operation	**ALUop**
add a, b	00001
comp b	00101
and a, b	00010
xor a, b	00011
diff a, b	00100
shll a, b	01010
shrl a, b	01000
shra a, b	01001
update(a,s)	00000
update(a,z)	00000
none	00000

Table 10: Truth Table for Jump Control

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