Computer Organisation and Architecture Laboratory

Design of a KGP-miniRISC Processor

November 9, 2022

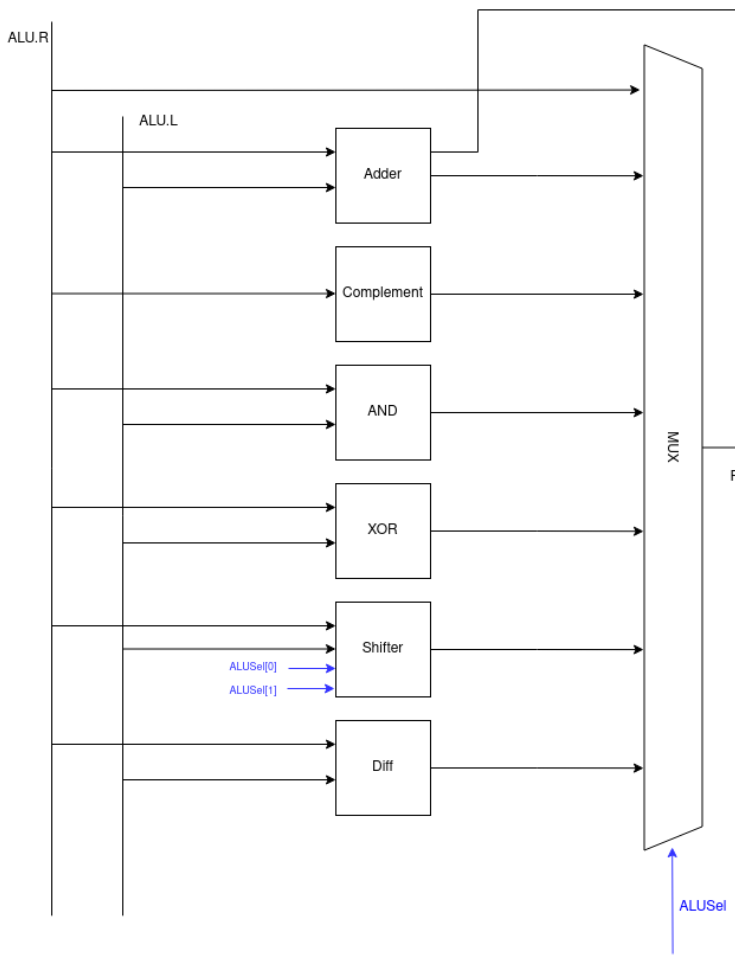
Group 68

Likhith Reddy Morreddigari 20CS10037
Shivansh Shukla 20CS10057

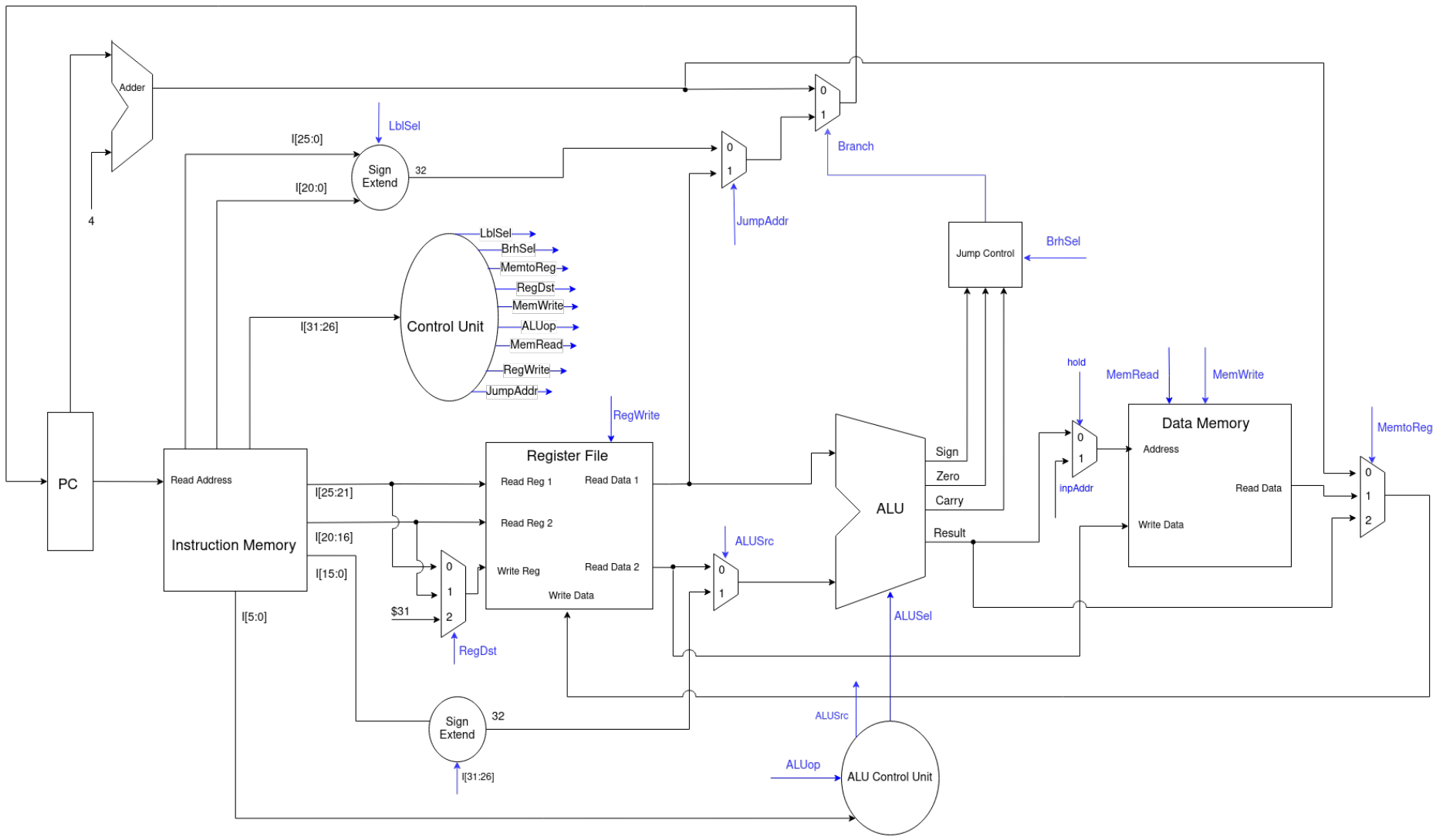
1 Instruction Set Architecture

Class	Instruction	Usage	Meaning	Opcode	Function code
Arithmetic	Add	add rs, rt	$rs \leftarrow (rs) + (rt)$	010000	000000
	Complement	comp rs, rt	rs $\leftarrow$ 2's complement of (rt)	010000	000001
Logic	AND	and rs, rt	$rs \leftarrow (rs) \wedge (rt)$	010000	000010
	XOR	xor rs, rt	$rs \leftarrow (rs) \oplus (rt)$	010000	000011
Shift	Shift left logical	shll rs, sh	$r \leftarrow (rs)$ left-shifted by sh	010000	000100
	Shift right logical	shrl rs, sh	$r \leftarrow (rs)$ right-shifted by sh	010000	000101
	Shift left logical variable	shllv rs, rt	$r \leftarrow (rs)$ left-shifted by (rt)	010000	000110
	Shift right logical	shrlv rs, rt	$r \leftarrow (rs)$ right-shifted by (rt)	010000	000111
	Shift right arithmetic	shra rs, sh	$r \leftarrow (rs)$ arithmetic right-shifted by sh	010000	001000
	Shift right arithmetic variable	shrav rs, rt	$r \leftarrow (rs)$ right-shifted by (rt)	010000	001001
Complex	Diff	diff rs, rt	$rs \leftarrow$ the LSB bit at which rs and rt differ	010000	001010
Arithmetic Immediate	Add immediate	addi rs,imm	$rs \leftarrow (rs) + imm$	110010	NA
	Complement immediate	compi rs,imm	rs $\leftarrow$ 2's complement of (imm)	110001	NA
Memory	Load Word	lw rt, imm(rs)	$rt \leftarrow mem[(rs) + imm]$	000000	NA
	Store Word	sw rt, imm(rs)	$mem[(rs) + imm] \leftarrow rt$	000001	NA
Branch	Unconditional branch	b L	goto L	100000	NA
	Branch Register	br rs	goto (rs)	100001	NA
	Branch on less than 0	bltz rs, L	if (rs) < 0 then goto L	100010	NA
	Branch on flag zero	bz rs, L	if (rs) $= 0$ then goto L	100011	NA
	Branch on flag not zero	bnz rs, L	if (rs) $\neq 0$ then goto L	100100	NA
	Branch and Link	bl L	goto L and $31 \leftarrow (PC) + 4$	100101	NA
	Branch and Carry	bcy L	goto L if carry $= 1$	100110	NA
	Branch and No Carry	bncy L	goto L if carry $= 0$	100111	NA

2 ALU



3 Datapath



4 Instruction Module and Encoding

R-Format Instructions

opcode	rs	rt	don't care	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

I-Format Instructions

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

Base Addressing Instructions

opcode	rs	rt	offset
6 bits	5 bits	5 bits	16 bits

Branch 1-Format Instructions

opcode	rs	L
6 bits	5 bits	21 bits

Branch 2-Format Instructions

opcode	L
6 bits	26 bits

5 Control Signals

Instr	Type	opcode	func	Reg Dst	Reg Write	Mem Read	Mem Write	MemTo Reg	Jump Addr	Lbl Sel	Brh Sel	ALU Src	ALUOp	ALUSel
add	R	010000	000000	00	1	0	0	10	X	X	0000	0	000	0000
comp	R	010000	000001	00	1	0	0	10	X	X	0000	0	000	0001
and	R	010000	000010	00	1	0	0	10	X	X	0000	0	000	0010
xor	R	010000	000011	00	1	0	0	10	X	X	0000	0	000	0011
shll	R	010000	000100	00	1	0	0	10	X	X	0000	1	000	0100
shrl	R	010000	000101	00	1	0	0	10	X	X	0000	1	000	0101
shllv	R	010000	000110	00	1	0	0	10	X	X	0000	0	000	0100
shrlv	R	010000	000111	00	1	0	0	10	X	X	0000	0	000	0101
shra	R	010000	001000	00	1	0	0	10	X	X	0000	1	000	0110
shrav	R	010000	001001	00	1	0	0	10	X	X	0000	0	000	0110
diff	R	010000	001010	00	1	0	0	10	X	X	0000	0	000	1001
lw	B	000000	NA	01	1	1	0	01	X	X	0000	1	011	0000
sw	B	000001	NA	00	0	0	1	X	X	X	0000	1	011	0000
addi	I	110010	NA	00	1	0	0	10	X	X	0000	1	100	0000
compi	I	110001	NA	00	1	0	0	10	X	X	0000	1	010	0001
b	Br 2	100000	NA	X	0	0	0	X	0	0	1000	1	001	1000
br	Br 1	100001	NA	X	0	0	0	X	1	X	1001	1	001	1000
bltz	Br 1	100010	NA	X	0	0	0	X	0	1	1010	1	001	1000
bz	Br 1	100011	NA	X	0	0	0	X	0	1	1011	1	001	1000
bnz	Br 1	100100	NA	X	0	0	0	X	0	1	1100	1	001	1000
bl	Br 2	100101	NA	10	1	0	0	00	0	0	1101	1	001	1000
bcy	Br 2	100110	NA	X	0	0	0	X	0	0	1110	1	001	1000
bncy	Br 2	100111	NA	X	0	0	0	X	0	0	1111	1	001	1000

R: R-Format Instructions, I: I-Format Instructions, B: Base Addressing Instructions, Br 1: Branch 1-Format Instructions, Br 2: Branch 2-Format Instructions, X: Don't Care

Control Signals Description

- **RegDst :** Determines the register to which data will be written. Used to choose between rs, rt and $31.
- **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 any data should be written to the data memory or not.
- **MemToReg :** Determines which data should be written to a register. Used to choose between data from Data memory, PC+4 and the result of ALU.
- **JumpAddr :** Determines the address to be selected during a jump operation. Only in case of *brrs* instruction, we need to jump to address stored in a register.
- **LblSel :** Determines which needs to be sign extended out of two types of branch instructions. It is 1 for Branch 1-Format instructions and 0 for Branch 2-Format instructions.
- **BrhSel :** Determines the kind of branch instructions and distinguish between them.
- **ALUSrc :** Determines the 2^{nd} input of ALU. Used to choose between the content of other register and sign extended result in case of immediate operations.
- **ALUOp :** We get the values of ALUOp from *opcode* and $func$ in control unit. For non-R type instructions which have no $func$, it helps to distinguish which kind of ALU operations need to be performed.

- **ALUSel :** Determines which kind of ALU operation needs to be done. Used to choose between Adder, Complement, AND, XOR, Shifer and Diff. Also used to distinguish between various kinds of shift operations, ie, logical left shift, logical right shift and arithmetic right shift.
- **hold :** Determines the input address for data memory. Initially we need to keep hold = 0, and once the execution is done, we use hold = 1 to receive the value stored in particular address in data memory through $InpAddr$.

6 Jump Control

Instr	**opcode**	**Zero**	**Sign**	**Carry**	**ValidJump**
b	100000	X	X	X	1
br	100001	X	X	X	1
bltz	100010	X	1	X	1
bz	100011	1	X	X	1
bnz	100100	0	X	X	1
bl	100101	X	X	X	1
bcy	100110	X	X	1	1
bncy	100111	X	X	0	1