Computer Organization and Architecture Laboratory
KGPminiRISC

Group 63

Members :

Pranav Nyati - 20CS30037
Shreyas Jena - 20CS30049

1. Instruction Set Architecture

Class	**Instruction**	**Usage**	**Meaning**	**Opcode**	**Func**
Arithmetic	Add Complement	add rs,rt comp rs,rt	rs ← (rs) + (rt) rs ← 2's comp (rt)	000000 000000	000000 000001
Logic	AND XOR	and rs,rt xor rs,rt	rs ← (rs) ∧ (rt) rs ← (rs) ⊕ (rt)	000011 000011	000000 000001
Shift	Shift Left Logical Shift Right Logical Shift Left Logical Variable Shift Right Logical Variable Shift Right Arithmetic Shift Right Arithmetic Variable	shll rs, sh shrl rs, sh shllv rs, rt shrlv rs, rt shra rs, sh shrav rs, rt	rs ← (rs) l-shift by sh rs ← (rs) r-shift by sh rs ← (rs) l-shift by (rt) rs ← (rs) r-shift by (rt) rs ← (rs) ar r-shift by sh rs ← (rs) r-shift by (rt)	000100 000100 000100 000100 000100 000100	000000 000001 000010 000011 000100 000101
Arithmetic Immediate	Add Immediate Complement Immediate	addi rs, imm compi rs, imm	rs ← (rs) + imm rs ← 2's comp of imm	000001 000010	NA NA
Memory	Load Word Store Word	lw rt, imm(rs) sw rt, imm(rs)	rt ← mem[(rs) + imm] mem[(rs) + imm] ← (rt)	000101 000110	NA NA

Branch	Branch on less than 0
	Branch on flag zero
	Branch on flag not zero
	Branch Register
	Unconditional Branch
	Branch and link
	Branch on Carry
	Branch on No Carry
		bltz rs, L
		bz rs, L
		bnz rs, L
		br rs
		b L
		bl L
		bcy L
		bncy L
			if (rs) < 0 then goto L
			if (rs) = 0 then goto L
			if (rs) ≠ 0 then goto L
			goto (rs)
			goto L
			goto L; $ra ← (PC) + 4
			if goto L if Carry = 1
			if goto L if Carry = 0
				000111
				001000
				001001
				001010
				001011
				001100
				001101
				001110
					NA
					NA
					NA
					NA
					NA
					NA
					NA
					NA

Complex	Diff	Diff	rs ← the LSB bit at which rs and rt differ	001111	000000

2. Instruction Format and Encoding

2.1 R-Format

Opcode	**rs**	**rt**	**Don't care**	**shamt**	**func**
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Instruction	**Opcode**	**Func**
add comp	000000	000000 000001
and xor	000011	000000 000001
shll shrl shllv shrlv shra shrav	000100	000000 000001 000010 000011 000100 000101
Diff	001111	000000

2.2 I-Format

Opcode	**rs**	**rt**	**Immediate**
6 bits	5 bits	5 bits	16 bits

Instruction	**Opcode**	**Func**
addi compi	000001 000010	NA NA
lw sw	000101 000110	NA NA

2.3 B-Formats

2.3.1 Type-1 (B1) (Register-based jump)

Opcode	**rs**	**Don't care**	**Immediate**
6 bits	5 bits	5 bits	16 bits

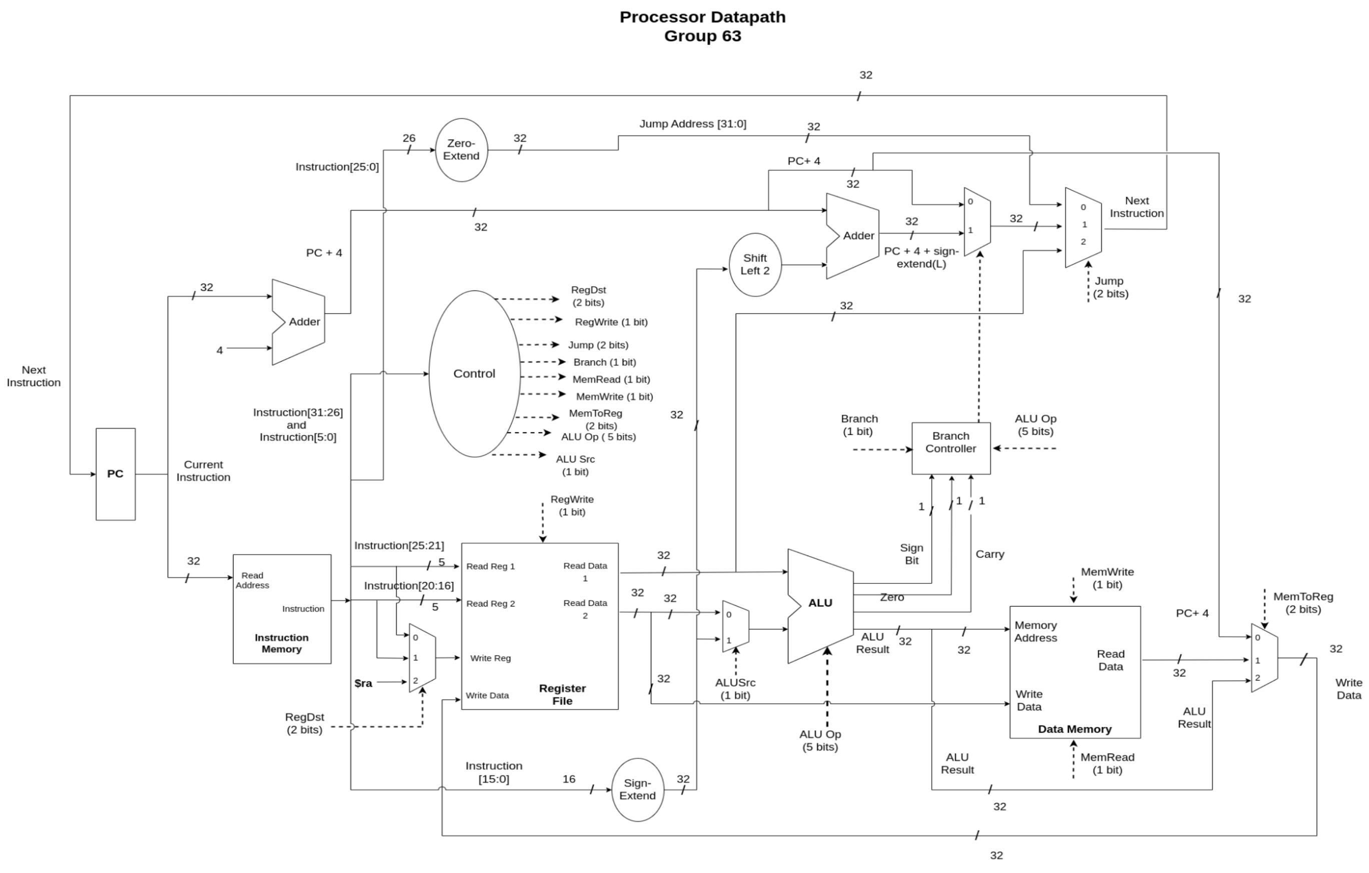
Instruction	**Opcode**	**Func**
bltz	000111	NA
bz	001000	NA
bnz	001001	NA
br	001010	NA

2.3.2 Type-2 (B2) (Unconditional and flag-based jump)

Opcode	**Label**
6 bits	26 bits

Instruction	**Opcode**	**Func**
b	001011	NA
bl	001100	NA
bcy	001101	NA
bncy	001110	NA

3. Processor Datapath Diagram



4. Control Signals Truth Table

Instr	**Opcode** (6 bits)	**Func** (6 bits)	**ALUop** (5 bits)	**ALUSrc** (1 bit)	**RegDst** (2 bits)	**Reg Write** (1 bit)	**Jump** (2 bits)	**Branch** (1 bit)	**Mem Read** (1 bit)	**Mem Write** (1 bit)	**MemToReg** (2 bits)
add	000000	000000	11111	0	00	1	01	0	0	0	10
comp	000000	000001	10000	0	00	1	01	0	0	0	10
addi	000001	NA	00001	1	00	1	01	0	0	0	10
compi	000010	NA	00010	1	00	1	01	0	0	0	10
AND	000011	000000	00011	0	00	1	01	0	0	0	10
XOR	000011	000001	00100	0	00	1	01	0	0	0	10
shll	000100	000000	10001	0	00	1	01	0	0	0	10
shrl	000100	000001	10010	0	00	1	01	0	0	0	10
shllv	000100	000010	10011	0	00	1	01	0	0	0	10
shrlv	000100	000011	10100	0	00	1	01	0	0	0	10
shra	000100	000100	10101	0	00	1	01	0	0	0	10
shrav	000100	000101	10110	0	00	1	01	0	0	0	10
lw	000101	NA	00101	1	01	1	01	0	1	0	01
sw	000110	NA	00110	1	11	0	01	0	0	1	11
bltz	000111	NA	00111	0	NA	0	01	1	0	0	NA
bz	001000	NA	01000	0	NA	0	01	1	0	0	NA
bnz	001001	NA	01001	0	NA	0	01	1	0	0	NA
br	001010	NA	01010	0	NA	0	10	1	0	0	NA
b	001011	NA	01011	NA	NA	0	00	1	0	0	NA
bl	001100	NA	01100	NA	10	1	00	1	0	0	00
bcy	001101	NA	01101	NA	NA	0	01	1	0	0	NA
bncy	001110	NA	01110	NA	NA	0	01	1	0	0	NA
Diff	001111	000000	01111	0	00	1	01	0	0	0	10

5. Description of Control Signals

- ***ALUop*** - Determines the type of operation to be carried out in the ALU.
- ***ALUsrc*** - Determines the content to pass through as the second input of the ALU. Possible options are :- the contents of a register (for R-type instructions), or the sign-extended result (for immediate operations).
- ***RegDst*** - Determines the register to which data will be written. Used to choose between rs, rt and $ra.
- ***RegWrite*** - Determines whether data should be written to a register or not.
- ***Jump*** - Selects the appropriate line to write to the next instruction address line. Possible options are :- zero-extended jump address, PC-relative address (obtained from 2:1 MUX) and address read from the register file.
- ***Branch*** - Determines whether an instruction is a branch instruction or not.
- ***MemRead*** - Determines whether data should be read from the data memory or not.
- ***MemWrite*** - Determines whether data should be written to the data memory or not.
- ***MemToReg*** - Selects the appropriate line to write to the register file. Used to choose between :- (PC) + 1 (for a bl instruction), the ALU result (for an R-type instruction) or data from the data memory (for a sw instruction).