Computer Organization and Architecture Laboratory

IIT Kharagpur

Group 75

Yashwant Krishna, 20CS30036
Sarthak Nikumbh, 20CS30035

R-Format encoding and instructions:

opcode	rs	rt	shamt	func
6 bits	5 bits	5 bits	5 bits	11 bits

Instructions	opcode	func
add	000000	00000000000
comp	000000	00000000001
diff	000000	00000000010
and	000001	00000000000
xor	000001	00000000001
shll	000010	00000000000
shrl	000010	00000000001
shllv	000010	00000000010
shrlv	000010	00000000011
shrar	000010	00000000100
shrav	000010	00000000101

I-Format encoding and instructions:

opcode	rs	Don’t care	immediate
6 bits	5 bits	5 bits	16 bits

Instructions	opcode
addi	000011
compi	000100

Load and encoding and instructions:

opcode	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

Instructions	opcode
lw	000101
sw	000110

16-Label Jump encoding and instructions:

opcode	rs	Don’t care	immediate
6 bits	5 bits	5 bits	16 bits

Instructions	opcode
bltz	000111
bz	001000
bnz	001001

26-Label Jump encoding and instructions:

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

Instructions	opcode
br	001010

26-Label Jump encoding and instructions:

opcode	Label
6 bits	26 bits

Instructions	opcode
b	001011
bl	001100
bcy	001101
bncy	001110

Truth table for control signals:

Instr	RegDst	RegWrite	MemRead	MemWrite	MemToReg	ALUsrc	ALUop	ALUsel	Branch	Jump Addr	LblSell
add	00	1	0	0	10	0	00001	0	0	-	-
comp	00	1	0	0	10	0	00101	1	0	-	-
diff	00	1	0	0	10	0	11111	1	0	-	-
and	00	1	0	0	10	0	00010	0	0	-	-
xor	00	1	0	0	10	0	00011	0	0	-	-
shll	00	1	0	0	10	1	01010	0	0	-	-
shrl	00	1	0	0	10	1	01000	0	0	-	-
shllv	00	1	0	0	10	0	01010	0	0	-	-
shrlv	00	1	0	0	10	0	01001	0	0	-	-
shra	00	1	0	0	10	0	01001	0	0	-	-
shrav	00	1	0	0	10	0	01001	0	0	-	-
addi	00	1	0	0	10	1	00001	1	0	-	-
compi	00	1	0	0	10	1	00101	0	0	-	-
lw	01	1	1	0	01	1	10101	0	0	-	-
sw	-	0	0	1	-	1	10101	0	0	-	-
bltz	-	0	0	0	-	-	00000	-	1	0	1
bz	-	0	0	0	-	-	00000	-	1	0	1
bnz	-	0	0	0	-	-	00000	-	1	0	1
br		-	0	0	-	-	00000	-	1	1	-
b	-	0	0	0	-	-	00000	-	1	0	0
bl	10	1	0	0	00	-	00000	-	1	0	0
bcy	-	0	0	0	-	-	00000	-	1	0	0
bncy	-	0	0	0	-	-	00000	-	1	0	0

Where:

- ➔ **RegDst:** Chosen the register among **rs**, **rt**, and **$ra** to which the data is written
- ➔ **RegWrite: 1** if the data has to be written to the register, **0** if not
- ➔ **MemRead: 1** if the data is read from the memory, **0** if not
- ➔ **MemWrite: 1** if the data is written to the memory, **0** if not
- ➔ **MemToReg:** Selects the appropriate line to write to the register file
- ➔ **ALUSrc: 1** if chosen R-type instruction, and **0** if chosen I-type instruction
- ➔ **ALUop:**

- ALUop[4]: memory or R-type instruction
- ALUop[3]: **1** if there is a shift instruction
- ALUop[2]: **1** if there is a complement instruction
- ALUop[1]: **1** if left-shift and **0** if right-shift
- ALUop[0]: **1** if arithmetic shift and **0** if logical shift

➔ **ALUsel:** signal to differentiate between 2's complement operation and any other ALU operation

➔ **Branch: 1** if branch instruction, **0** if not

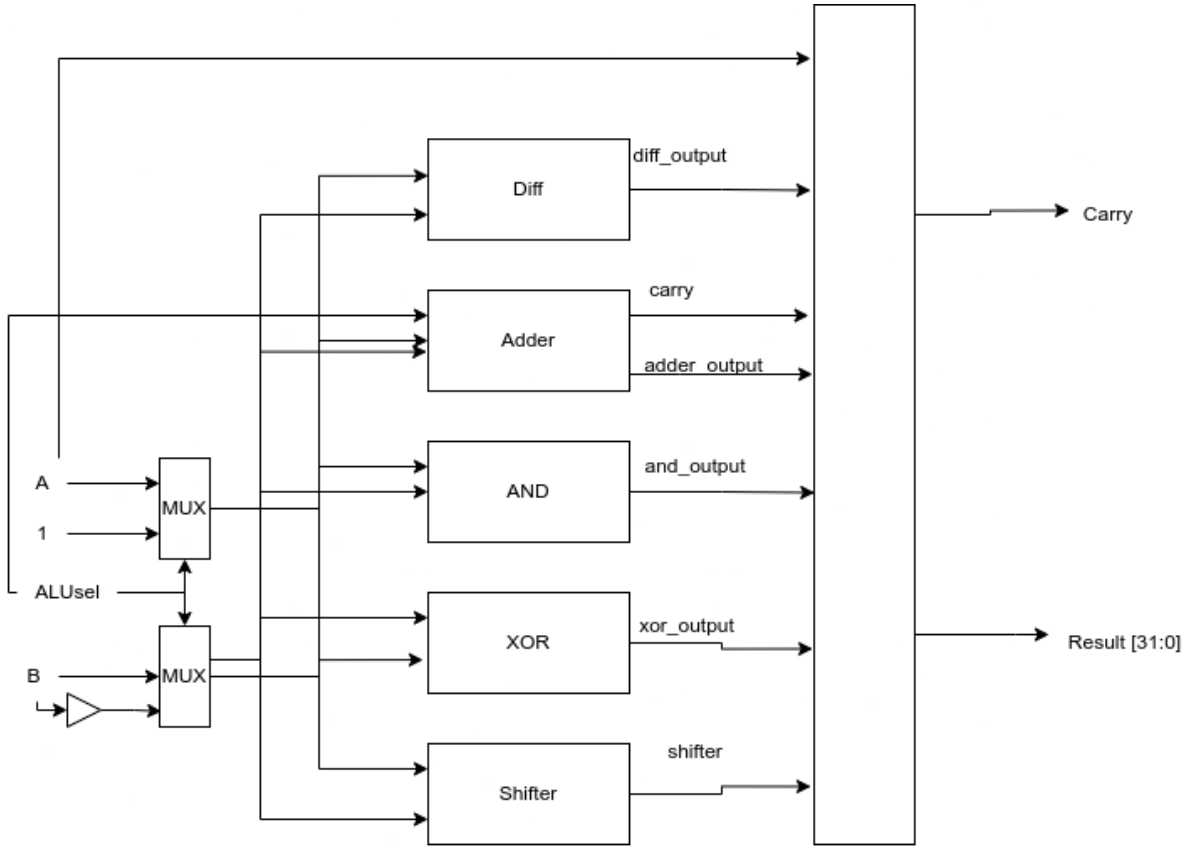
➔ **JumpAddr: 1** only for the **br** instruction, **0** for the others

➔ **LabelSel: 1** for 16-bit label instruction, **0** for 26-label instruction

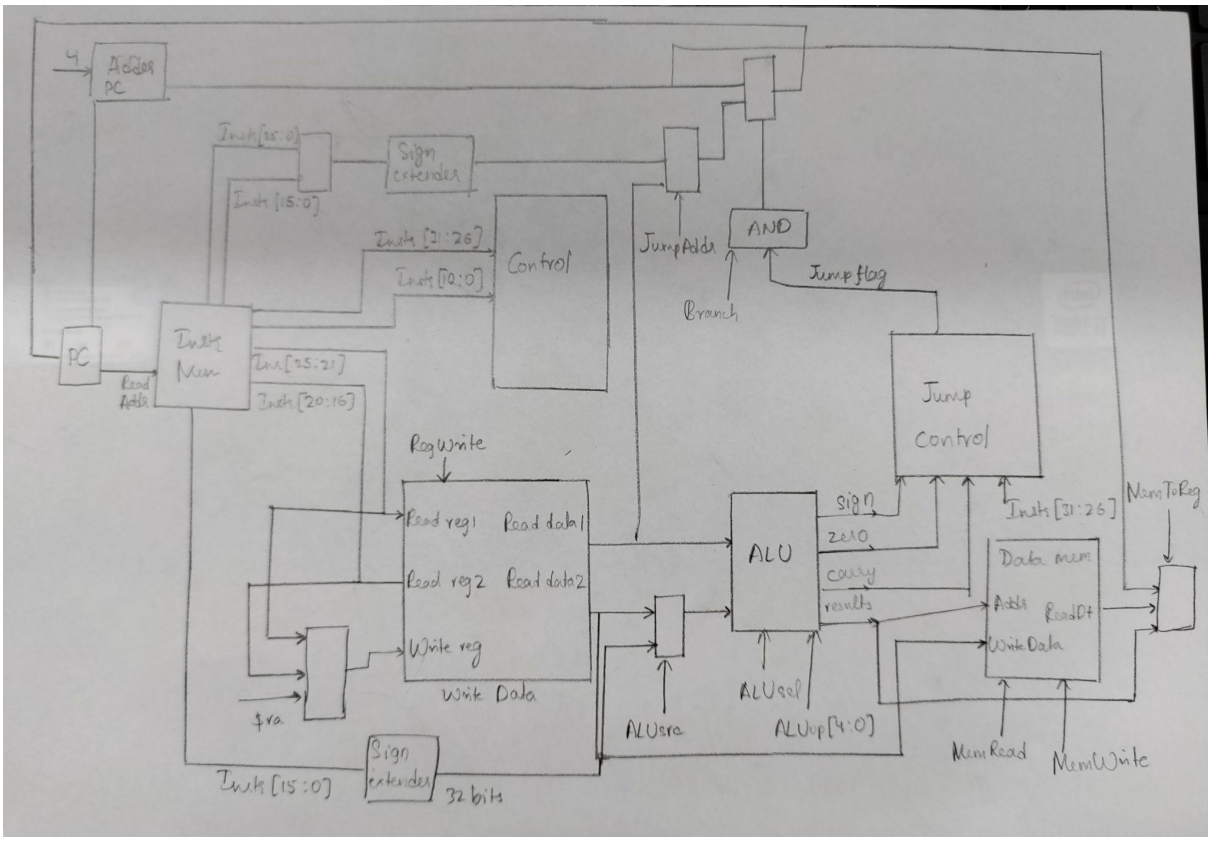
Truth table for jump control:

Instr	zero	sign	carry	can_jump
bltz	0	1	-	1
bz	1	0	-	1
bnz	0	-	-	1
br	-	-	-	1
b	-	-	-	1
bl	-	-	-	1
bcy	-	-	1	1
bncy	-	-	0	1

ALU diagram:


Diff
diff_output
Carry
carry
Adder
adder_output
A
1
MUX
AND
and_output
ALUsel
B
MUX
XOR
xor_output
Result [31:0]
shifter
Shifter

Datapath diagram:



Control: