Computer Organization and Architecture Laboratory

Assignment 6

KGP MiniRISC Processor

Group 53
Umang Singla - 20CS10068
Siddharth Vishwakarma - 20CS10082

Instruction Formats:

There are three types of instruction formats ➖

- R-format (Register format)
- J-format (Jump format)
- I-format (Immediate format)

R-format

Opcode for all R-format instructions are same

Field Size	6	5	5	5	6	5	Instructions
R-format	op	rs	rt	shamt	func	dc	add, cmp, and, xor, all-shift instructions

Op-code: 000000

Instruction	**func-code**
add	000001
cmp	000101
and	000010
xor	000011
shll	001100
shrl	001110
shllv	001000
shrlv	001010
shra	001111
shrav	001011
diff	000100

J-format

Field Size	6	26	Instructions
J-format	op	Target address	b, br, bl, bcy, bncy

The below table contains the op-code for all the above instructions

Instruction	**op-code**	**Format**
br	100000	op-code | rs | xxxxxxxx |
b	101000	op-code | label |
bcy	101001	op-code | label |
bncy	101010	op-code | label |
bl	101011	op-code | label |
bltz	110000	op-code | rs | xx | label |
bz	110001	op-code | rs | xx | label |
bn	110010	op-code | rs | xx | label |

I-format

Field Size	6	5	5	16	Instructions
R-format	op	rs	rt/dc	address/immediate	lw, sw, addi, cmpi

The below table contains the op-code for all the above instructions

Instruction	**op-code**
lw	010000

sw	011000
addi	001000
cmpi	001001

The following table summarizes all the instructions and the associated op-codes

LSB MSB	000	001	010	011	100	101	110	111
000	R-format							
001	addi	cmpi						
010	lw							
011	sw							
100	br							
101	b	bcy	bncy	bl				
110	bltz	bz	bnz					
111								

ALU Design

The proposed ALU has an internal adder module, xor modules, a barrel shifter with lines for direction and type (logical/arithmetic) and a diff module (which returns the position of lower most bit at which two numbers differ)

For the shift-amount to be given as input to the shift module, a separate input bus is also present. Depending on the shift-variable instructions, the 5 LSBs of the selected register or the shift input bus can be used as the shift input. Therefore, one of these inputs is chosen using a 2-1 Mux.

Because the ISA lacks a native subtract command, there is no carry-in instruction available for the complement instruction; instead, input-1 is switched with 32'd1 using a 2-1 MUX, and the adder module is utilised to compute the complement.

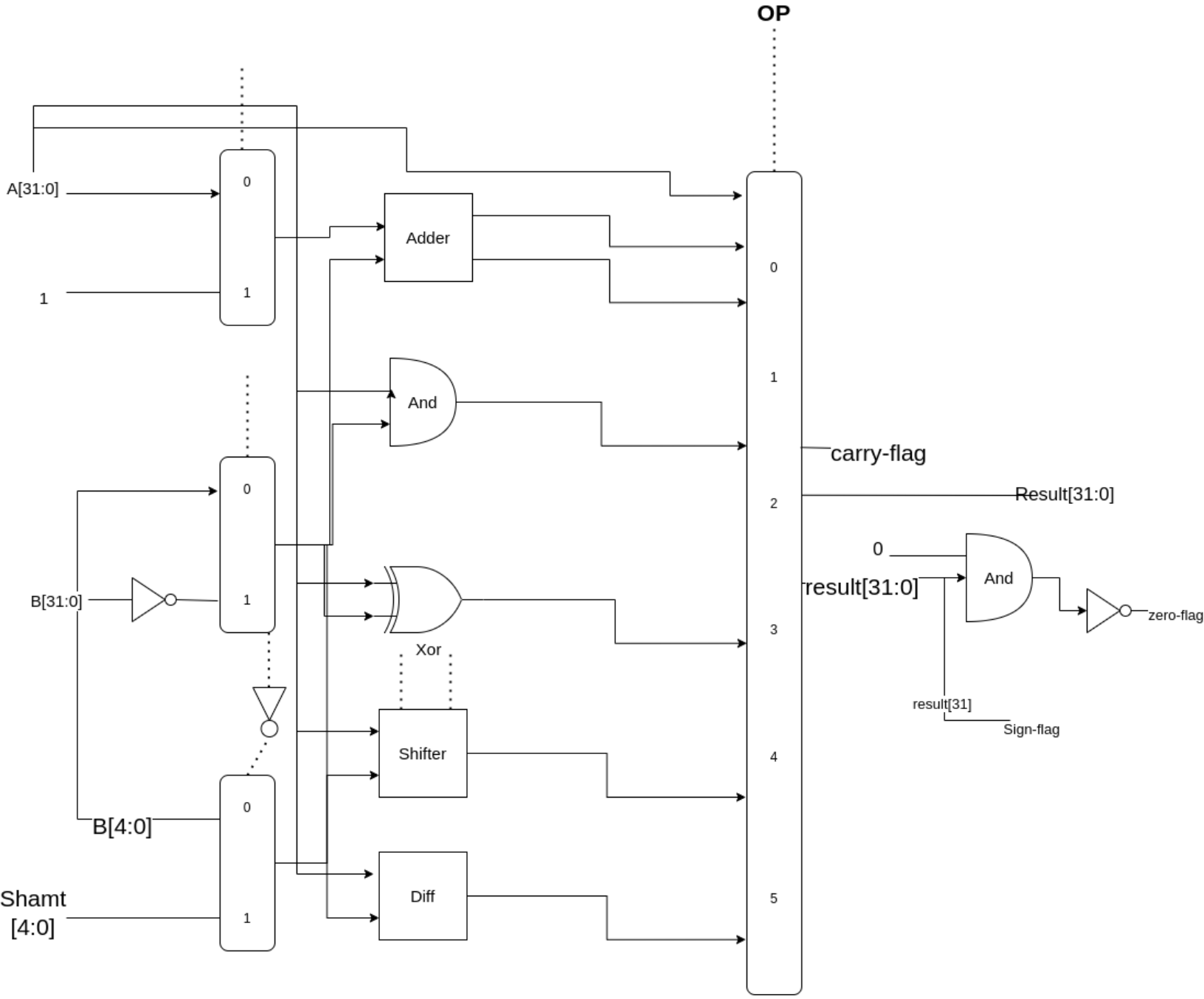
Flags from ALU:

1. Carry
2. Zero result
3. Sign of result

Truth table for ALU-control signals with associated op-codes and func-codes

Operation	Opcode	funcode	alucode
add	000000	000001	0001
comp	000000	000101	0101
addi	001000	-	0001
compi	001001	-	0101
and	000000	000010	0010
xor	000000	000011	0011
shll	000000	001100	1100
shrl	000000	001110	1110
shllv	000000	001000	1000
shrlv	000000	001010	1010
shra	000000	001111	1111
shrav	000000	001011	1011
lw	010000	-	0001
sw	011000	-	0001
b	101000	-	0000
br	100000	-	0000
bltz	110000	-	0000
bz	110001	-	0000
bnz	110010	-	0000
bl	101011	-	0000
bcy	101001	-	0000
bncy	101010	-	0000
diff	000000	000100	0100

ALU Diagram


OP
A[31:0]
1
0
1
Adder
And
B[31:0]
B[4:0]
Xor
Shifter
Shamt
[4:0]
Diff
0
1
2
3
4
5
carry-flag
Result[31:0]
0
result[31:0]
And
zero-flag
result[31]
Sign-flag

Truth table for control signals with associated op-codes and func-codes

Control signals:

1. **Regwrite**: whether to write into the register file or not
2. **RegDst[1:0]**: destination register for the write-register (can be $ra, rs, rt)
3. **ALUSrc:** Source for the 2nd input to the ALU (can be rt, sgn-extend(imm))
4. **MemRead:** whether to read from Data-memory or not
5. **MemWrite:** whether to write into the Data-memory or not
6. **Mem2Reg[1:0]:** write-data for the register files (can be PC+4, mem[], result_ALU)
7. **LblSel:** select type of addressing for PC-relative and PseudoDirect
8. **JumpAddr:** whether the jump address comes from a source reg (rs) or from a label

Operation	Opcode	RegDst	RegWrite	ALUSrc	MemRead	MemWrite	Mem2Reg	LblSel	Jump Sel
add	000000	00	1	0	0	0	00	x	x
comp	000000	00	1	0	0	0	00	x	x
addi	001000	00	1	1	0	0	00	x	x
compi	001001	00	1	1	0	0	00	x	x
and	000000	00	1	0	0	0	00	x	x
xor	000000	00	1	0	0	0	00	x	x
shll	000000	00	1	0	0	0	00	x	x
shrl	000000	00	1	0	0	0	00	x	x
shllv	000000	00	1	0	0	0	00	x	x
shrlv	000000	00	1	0	0	0	00	x	x
shra	000000	00	1	0	0	0	00	x	x
shrav	000000	00	1	0	0	0	00	x	x
lw	010000	01	1	1	1	0	01	x	x
sw	011000	x	0	1	0	1	x	x	x

b	101000	x	0	x	0	0	x	0	0
br	100000	x	0	x	0	0	x	x	1
bltz	110000	x	0	x	0	0	x	1	0
bz	110001	x	0	x	0	0	x	1	0
bnz	110010	x	0	x	0	0	x	1	0
bl	101011	10	1	x	0	0	10	0	0
bcy	101001	x	0	x	0	0	x	0	0
bncy	101010	x	0	x	0	0	x	0	0
diff	000000	00	1	0	0	0	00	0	1

Datapath Diagram