KGP-RISC ISA

Group: 024
Members: Parth Jindal, Pranav Rajput

Instruction formats:

There are 3 types of instruction formats:

1. R-format (Register format)
2. I-format (Immediate format)
3. J-format (Jump format)

R-format

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

Currently, all R-format instructions are kept under the same op-code.

op-code: **000000**

Instruction	**func-code**
add	000001
cmp	000010
and	000011
xor	000100
shll	000101
shrl	000110
shllv	000111
shrlv	001000
shra	001001
shrav	001010

I-format

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

The following table enlists the op-code for all the above instructions

Instruction	**op-code**
lw	010000
sw	011000
addi	001000
cmpi	001001

J-format

Field size	6	26	Instructions
J-format	op	target address	b, br, bl, bcy, bncy

The following table enlists the op-code for all the above instructions

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

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								

Arithmetic Logic Unit Design

The proposed ALU has an internal adder module, and xor modules and a barrel shifter with lines for direction and type(logical/arithmetic).

A separate input bus is also present for shift-amount to be passed as input to shift module. The shift input can either come directly from the shift input bus or from the 5 LSB's of the register specified in the *shift-variable instructions* .
A 2-1 Mux is hence used to select either of these inputs.

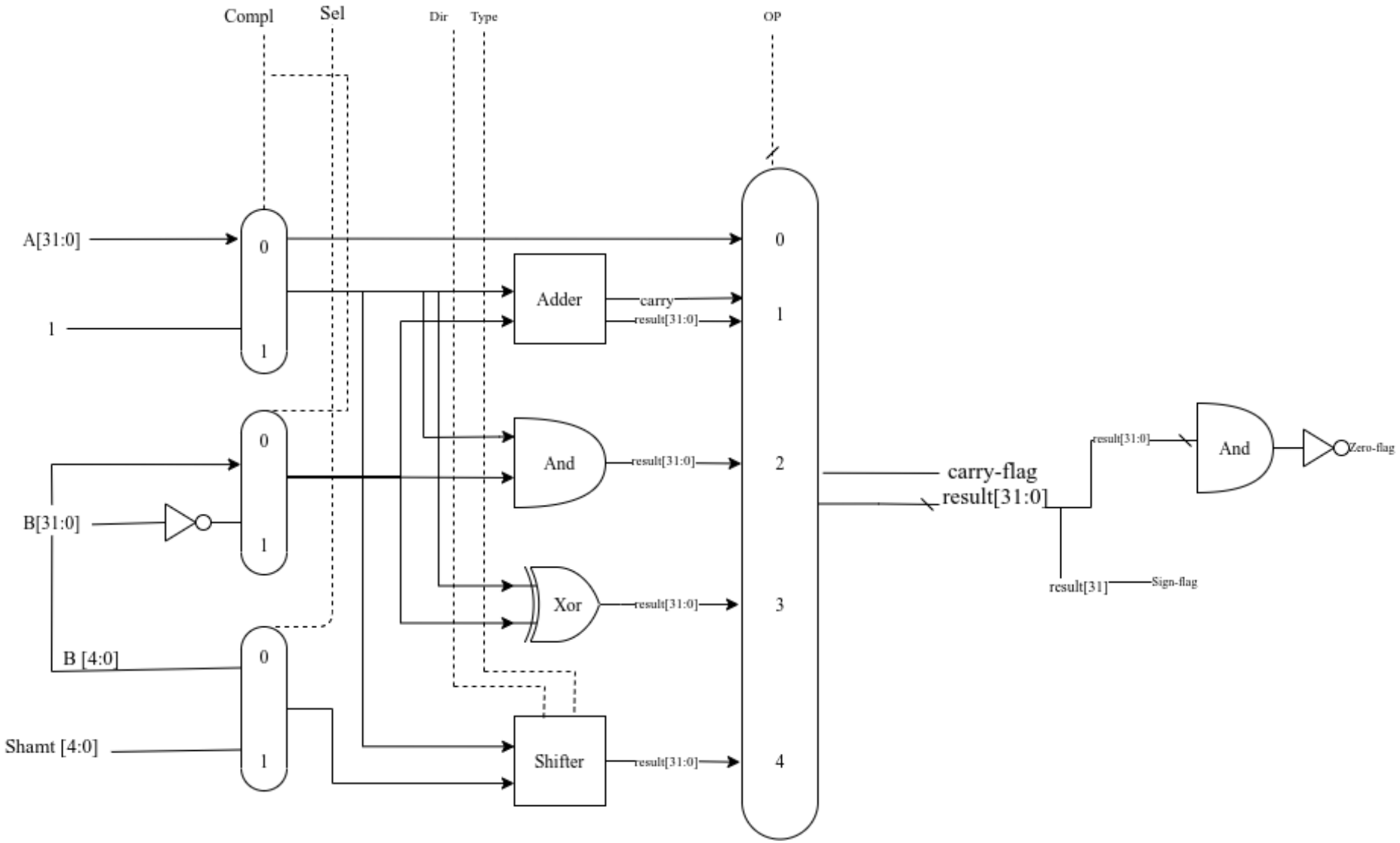
For simplicity, since the ISA doesn't have the subtract command natively, no carry-in has been provided rather for the complement instruction, a 2-1 MUX is used to switch input-1 with 32'd1 and the adder module is used to compute the complement.

Flags from ALU:

1. Carry
2. Zero result
3. Sign of result (0 for +, 1 for -)

The ALU design has been included in the documentation and also embedded here:

ALU DESIGN KGP-RISC



Datapath and control signals for the ISA

The given document (separate and embedded) shows the complete data paths and control signals for the Instruction set. The data-paths are shown using black lines and the control signals using the red lines. For simplicity the controller lines have not been joined to the modules themselves rather left open for better visual clarity.

The ISA contains 3 primary control modules:

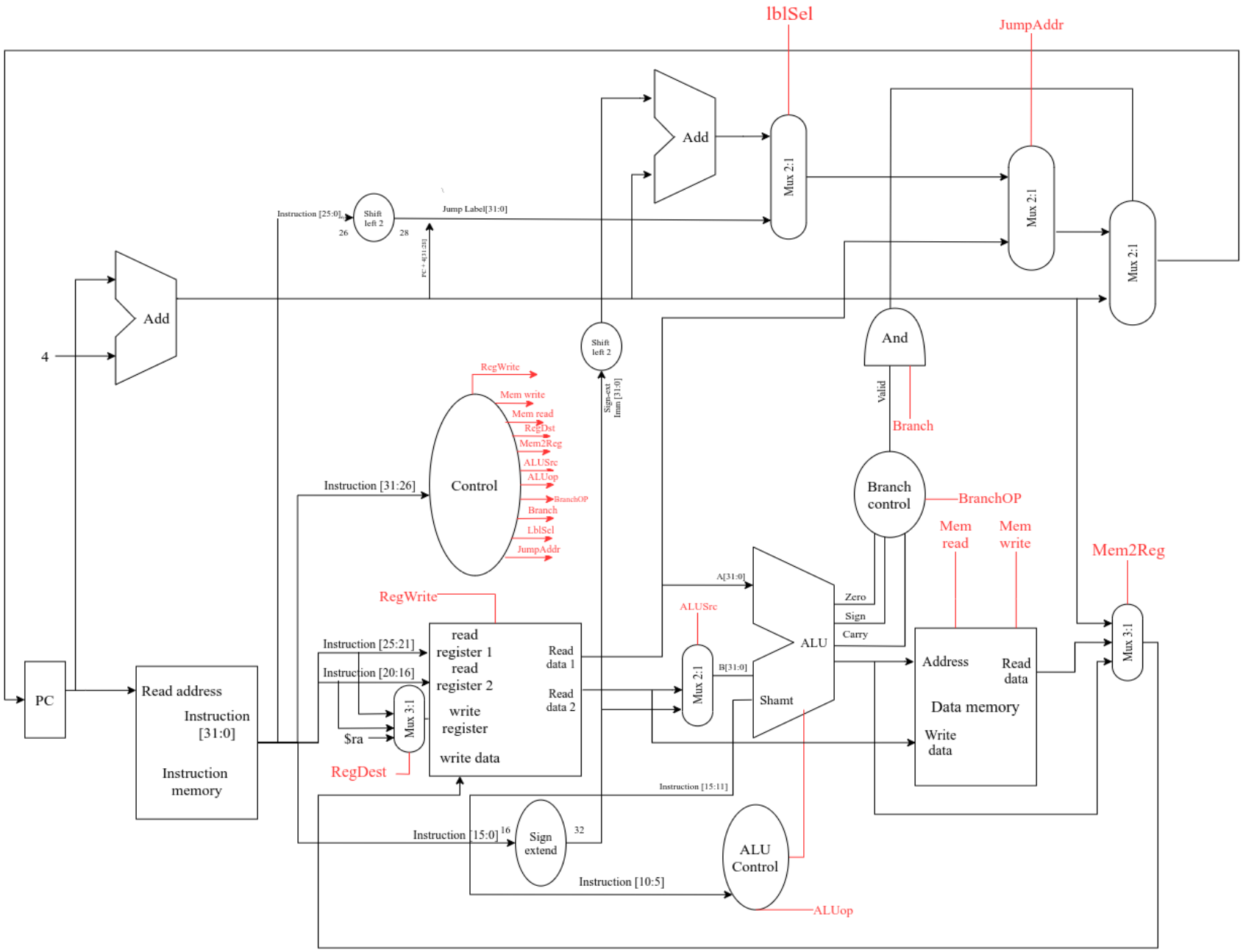
1. Controller: handles the primary signals to all modules
2. ALU-Controller: handles the ALU-specific control signals
3. Branch-Controller: handles the logic for branch on flag-signals and produces the output if it has to branch or not

There are 3 standard ways in which instruction memory can be referenced in the Instruction set:

1. Direct PC addressing
2. PC-relative addressing
3. (Pseudo)Direct jump addressing

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. *Branch*: whether to branch or not
8. *LblSel:* select type of addressing for PC-relative and PseudoDirect
9. *JumpAddr*: whether the jump address comes from a source reg (rs) or from a label


lblSel
JumpAddr
Add
Mux 2:1
Mux 2:1
Mux 2:1
Instruction [25:0]
Shift left 2
26
28
Jump Label[31:0]
PC + 4[31:28]
Add
4
Shift left 2
Sign-ext Imm[31:0]
And
Valid
Branch
RegWrite
Mem write
Mem read
RegDst
Mem2Reg
ALUSrc
ALUop
BranchOP
Branch
LblSel
JumpAddr
Instruction [31:26]
Control
Branch control
BranchOP
Mem read
Mem write
Mem2Reg
A[31:0]
ALUSrc
Zero
Sign
Carry
ALU
RegWrite
Instruction [25:21]
Instruction [20:16]
read register 1
read register 2
write register
write data
Read data 1
Read data 2
Mux 3:1
$ra
RegDest
Mux 2:1
B[31:0]
Shamt
Address
Read data
Data memory
Write data
Mux 3:1
PC
Read address
Instruction [31:0]
Instruction memory
Instruction [15:0]
16
Sign extend
32
Instruction [15:11]
Instruction [10:5]
ALU Control
ALUop

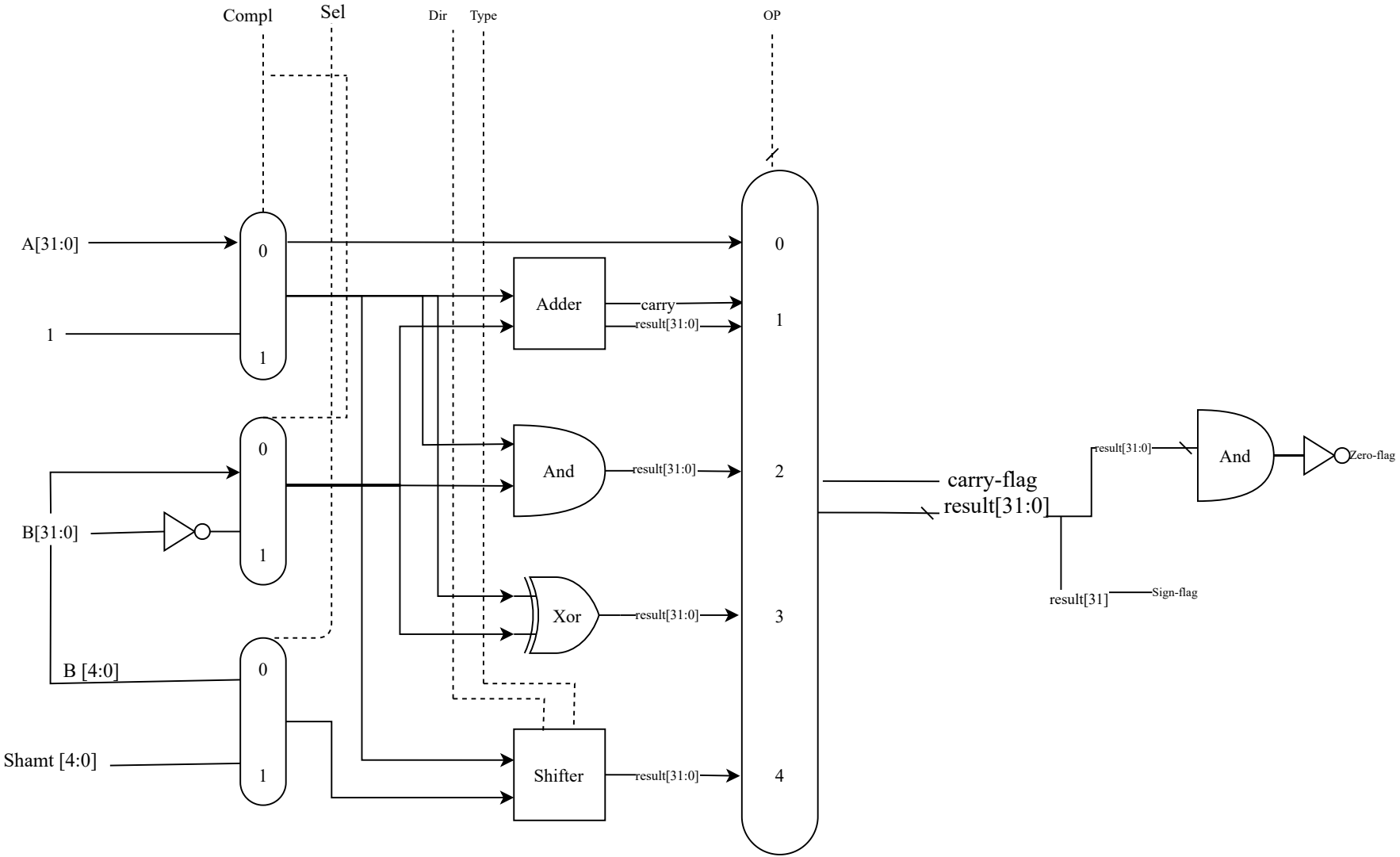
Truth table for control signals with associated op-codes and func-codes

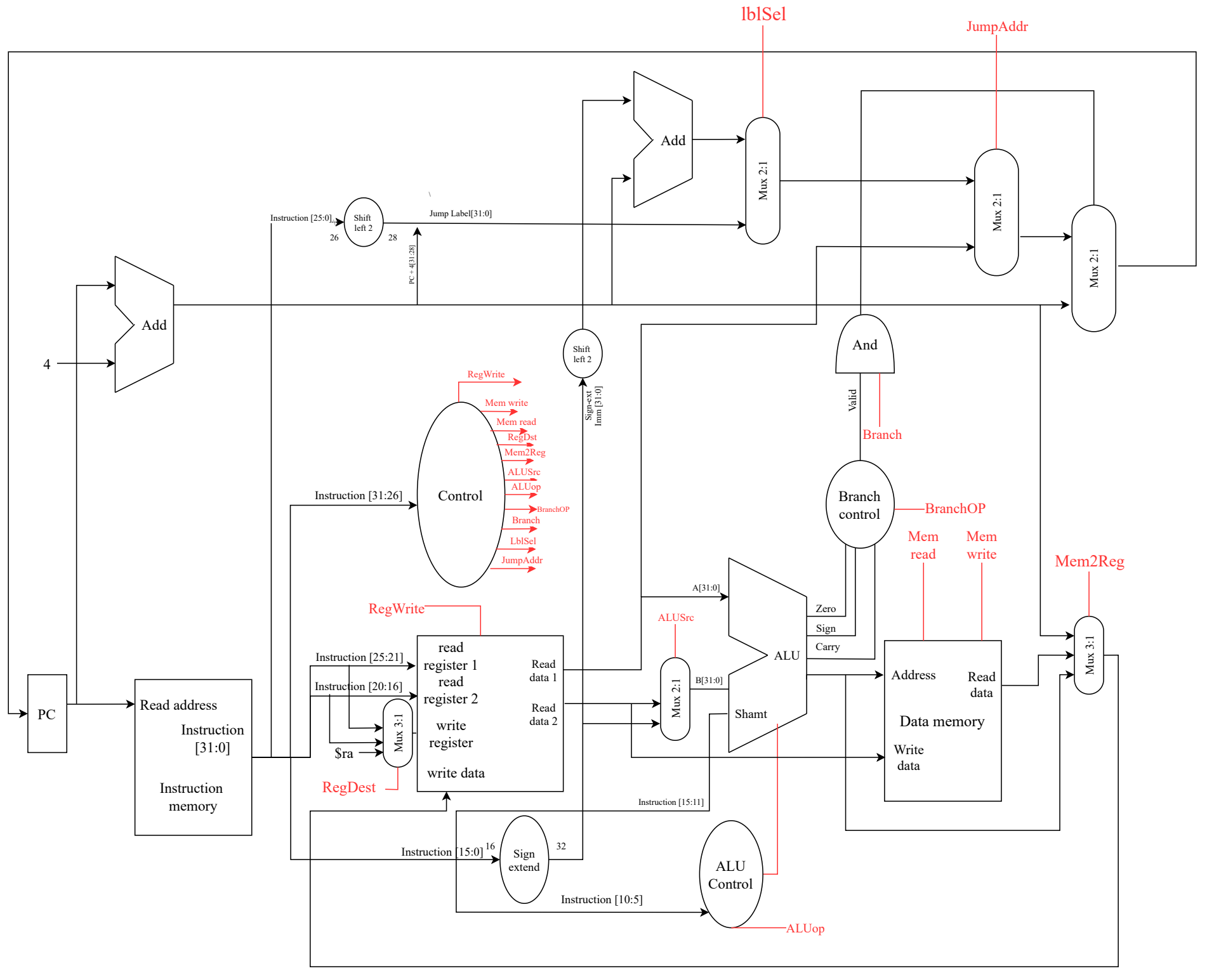
Op	Opcode	RegDst	RegWrite	ALUSrc	MemRead	MemWrite	Branch	Mem2Reg	LblSel	JumpAddr
add	000000	00	1	0	0	0	0	10/11	x	x
comp	000000	00	1	0	0	0	0	10/11	x	x
addi	001000	00	1	1	0	0	0	10/11	x	x
compi	001001	00	1	1	0	0	0	10/11	x	x
and	000000	00	1	0	0	0	0	10/11	x	x
xor	000000	00	1	0	0	0	0	10/11	x	x
shll	000000	00	1	x	0	0	0	10/11	x	x
shrl	000000	00	1	x	0	0	0	10/11	x	x
shllv	000000	00	1	x	0	0	0	10/11	x	x
shrlv	000000	00	1	x	0	0	0	10/11	x	x
shra	000000	00	1	x	0	0	0	10/11	x	x
shrav	000000	00	1	x	0	0	0	10/11	x	x
lw	010000	00	1	1	1	0	0	01	x	x
sw	011000	01	0	1	0	1	0	x	x	x
b	101000	x	0	x	0	0	1	x	0	0
br	100000	x	0	x	0	0	1	x	x	1
bltz	110000	x	0	x	0	0	1	x	1	0
bz	110001	x	0	x	0	0	1	x	1	0
bnz	110010	x	0	x	0	0	1	x	1	0
bl	101011	10/11	1	x	0	0	1	00	0	0
bcy	101001	x	0	x	0	0	1	x	0	0
bncy	101010	x	0	x	0	0	1	x	0	0

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

Operation	Opcode	Func-code	Compl	Sel	Dir	Type
add	000000	000001	0	x	x	x
comp	000000	000010	1	x	x	x
addi	001000	-	0	x	x	x
compi	001001	-	1	x	x	x
and	000000	000011	0	x	x	x
xor	000000	000100	0	x	x	x
shll	000000	000101	0	1	0	x
shrl	000000	000110	0	1	1	0
shllv	000000	000111	0	0	0	x
shrlv	000000	001000	0	0	1	0
shra	000000	001001	0	1	1	1
shrav	000000	001010	0	0	1	1
lw	010000	-	0	x	x	x
sw	011000	-	0	x	x	x
b	101000	-	0	x	x	x
br	100000	-	0	x	x	x
bltz	110000	-	0	x	x	x
bz	110001	-	0	x	x	x
bnz	110010	-	0	x	x	x
bl	101011	-	0	x	x	x
bcy	101001	-	0	x	x	x
bncy	101010	-	0	x	x	x

ALU DESIGN KGP-RISC




lblSel
JumpAddr
Add
Mux 2:1
Mux 2:1
Mux 2:1
Instruction [25:0]
Shift left 2
26
28
Jump Label[31:0]
PC + 4[31:28]
Add
4
Shift left 2
Sign-ext Imm [31:0]
And
Valid
Branch
RegWrite
Mem write
Mem read
RegDst
Mem2Reg
ALUSrc
ALUop
BranchOP
Branch
LblSel
JumpAddr
Instruction [31:26]
Control
Branch control
BranchOP
Mem read
Mem write
Mem2Reg
A[31:0]
Zero
Sign
Carry
ALUSrc
ALU
RegWrite
Instruction [25:21]
Instruction [20:16]
read register 1
read register 2
write register
write data
Read data 1
Read data 2
B[31:0]
Mux 2:1
Shamt
Address
Read data
Data memory
Write data
Mux 3:1
PC
Read address
Instruction [31:0]
Instruction memory
Mux 3:1
$ra
RegDest
Instruction [15:0]
16
Sign extend
32
Instruction [15:11]
Instruction [10:5]
ALU Control
ALUop