Indian Institute of Technology Kharagpur

Design of KGP-minRISC Processor

By

Ritwik Ranjan Mallik (20CS10049)
Utsav Basu (20CS30057)

1. Instruction Format

Instructions handled by the KGP-miniRISC processor can appear in five different formats. These are given below:

1. R-Type (Register Type)
2. I-Type (Immediate Type)
3. M-Type (Memory Type)
4. BA-Type (Branch Address Type)
5. BR-Type (Branch Register Type)

The opcodes have been decided such that more instructions can be added at a later time. The opcodes and func

1.1. R-type

op (6)	**rs (5)**	**rt (5)**	**shamt (10)**	**func (6)**

The higher 5 bits of shamt are ignored by the ALU while performing the shift operations.

Instruction Name	**opcode**	**func**
add	000000	000000
comp	000000	000001
and	001010	000000
xor	001010	000001
shll	010100	000000
shrl	010100	000001
shra	010100	000010

shllv	010101	000000
shrlv	010101	000001
shrav	010101	000010
diff	110010	000000

1.2. I-Type

op (6)	**rs (5)**	**imm (15)**	**fn (6)**

Instruction Name	**opcode**	**func**
addi	000001	000000
compi	000001	000001

1.2. M-Type

op (6)	**rs (5)**	**rt (5)**	**imm (16)**

Instruction Name	**opcode**	**func**
lw	011110	NA
lw	011111	NA

1.3. BA-Type

op (6)	addr (26)

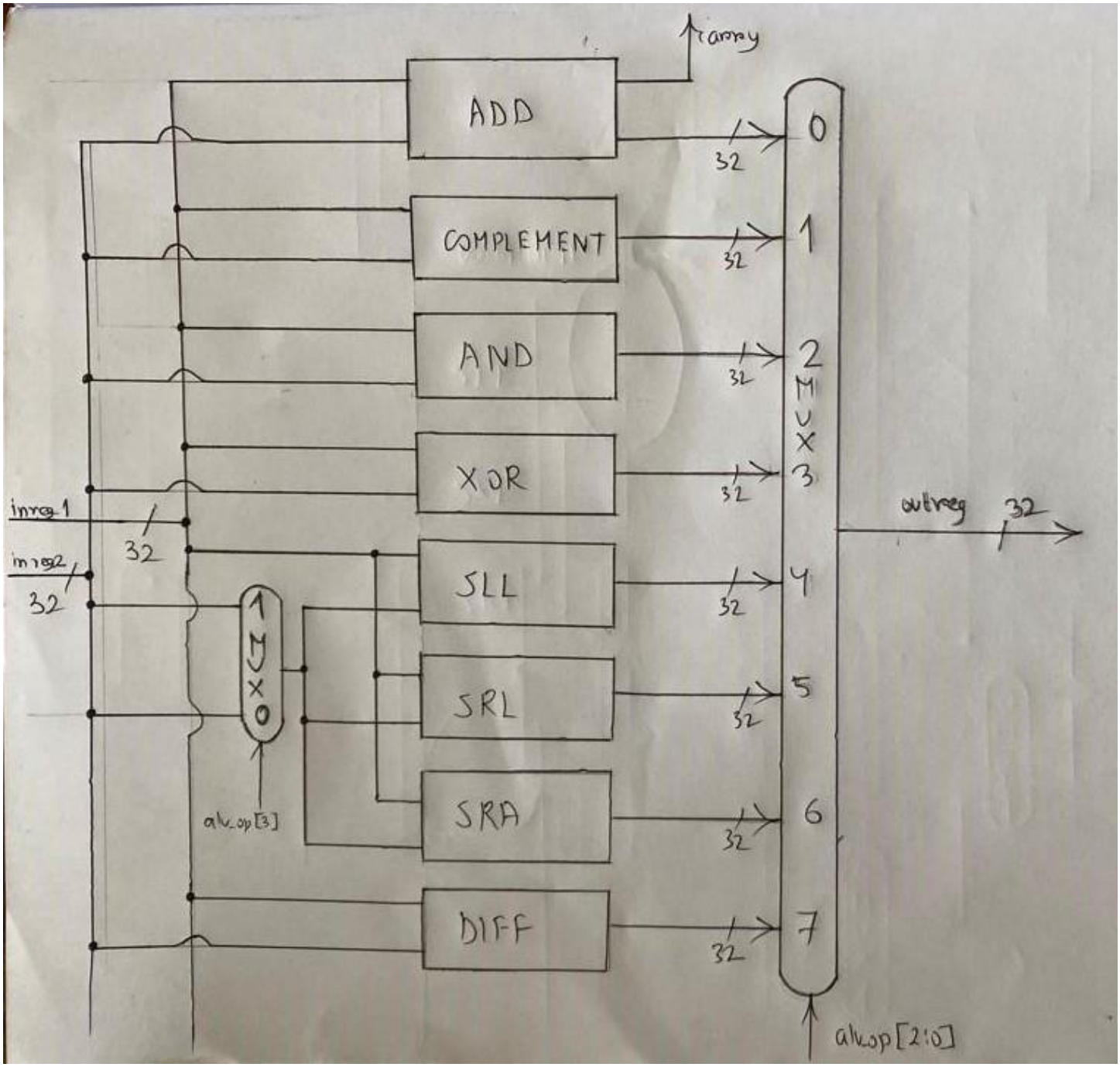
Instruction Name	opcode	func
b	101000	NA
bl	101001	NA
bcy	101010	NA
bncy	101011	NA

1.4. BR-Type

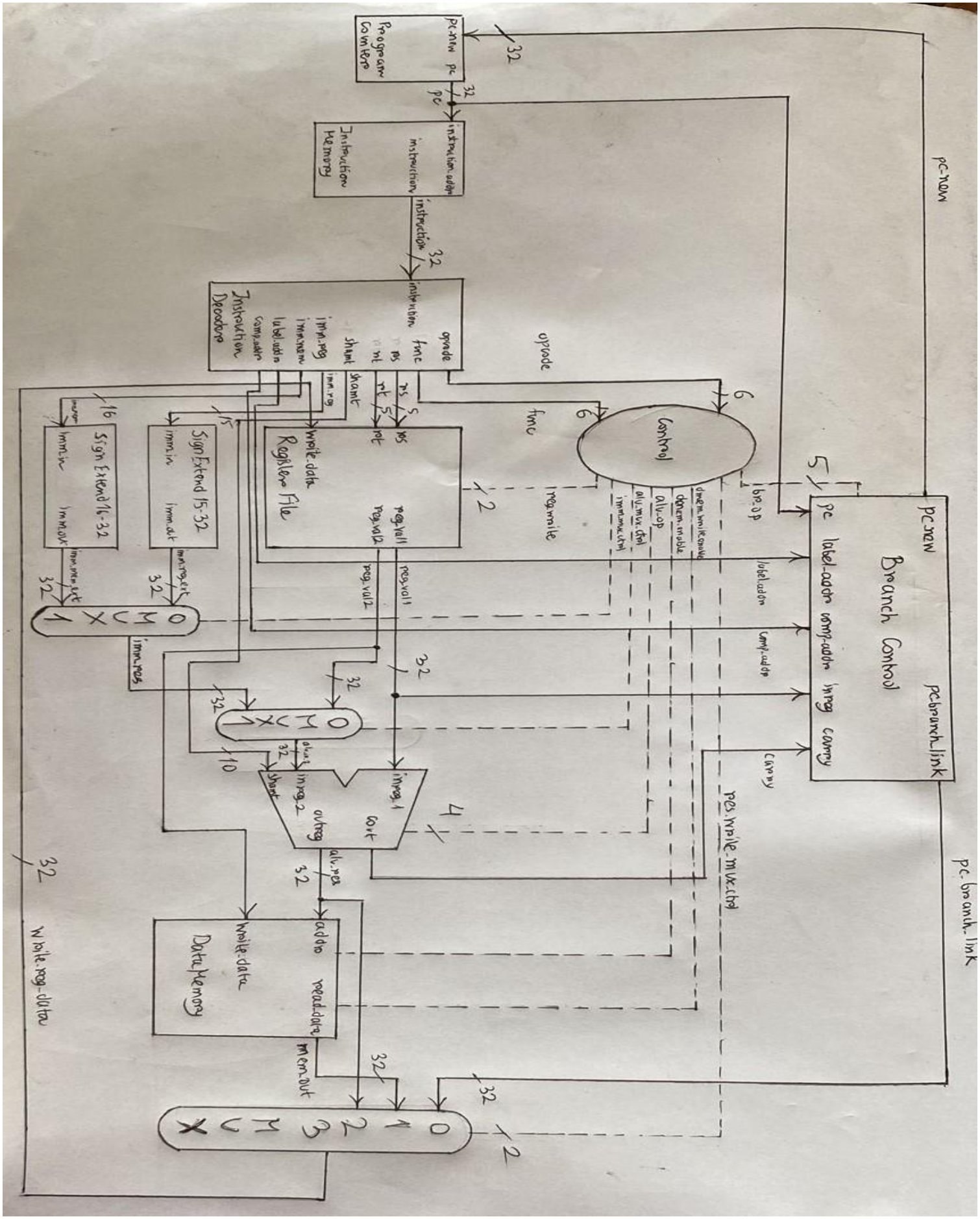
op (6)	rs (5)	addr (21)

Instruction Name	opcode	func
br	101100	NA
bltz	101101	NA
bz	101110	NA
bnz	101111	NA

2. ALU Architecture



3. Complete Processor Architecture



4. Control Signals

The single control unit generates all the 8 control signals. The first 4 control signals does the following:

1. **reg_write**: This controls where the writing will take place - 01 for writing to rs, 10 for writing to rt, 11 for writing to $31 and 00 for no writing.
2. **imm_mux_ctrl**: This mux controls which imm is implied by the instruction. 0 means imm for I-type instructions whereas 1 means imm for M-type instructions.
3. **alu_mux_ctrl**: This determines the second input to the ALU. 0 means the input is rt whereas 1 means imm value.
4. **alu_op**: This 4 bit control signal determines the output of the ALU. The MSB determines whether the input to the shift operations is from shamt or from rt. The last three bits determine the final result - 000 for sum, 001 for complement, 010 for bitwise and, 011 for bitwise xor, 100 for shift left logical, 101 for shift right logical, 110 for shift right arithmetic and 111 for diff.

op	**opcode**	**func**	**reg_write**	**imm_ mux_ ctrl**	**alu_ mux_ ctrl**	**alu_op**
add	000000	000000	01	0	0	0000
comp	000000	000001	01	0	0	0001
addi	000001	000000	01	0	1	0000
compi	000001	000001	01	0	1	0001
and	001010	000000	01	0	0	0010
xor	001010	000001	01	0	0	0011
shll	010100	000000	01	0	0	0100
shrl	010100	000001	01	0	0	0101

shra	010100	000010	01	0	0	0110
shllv	010101	000000	01	0	0	1100
shrlv	010101	000001	01	0	0	1101
shrav	010101	000010	01	0	0	1110
lw	011110	NA	10	1	1	0000
sw	011111	NA	00	1	1	0000
b	101000	NA	00	0	0	0000
br	101100	NA	00	1	0	0000
bltz	101101	NA	00	0	0	0000
bz	101110	NA	00	0	0	0000
bnz	101111	NA	00	0	0	0000
bl	101001	NA	11	0	0	0000
bcy	101010	NA	00	0	0	0000
bncy	101011	NA	00	0	0	0000
diff	110010	000000	01	0	0	00111

The next four control signal perform the following tasks:

1. **dmem_enable**: This signals turns the data memory on or off - 0 for off and 1 for on.
2. **dmem_write_enable**: This signals whether writing in the data memory is turned on or off - 0 for off and 1 for on.
3. **reg_write_mux_ctrl**: This signals which output is to be fed to the write_data port of the Register File to be written into an appropriate register - 10 for ALU's output, 01 for memory's output and 00 for return address.
4. **br_op**: This 5 bit signal determines the value of pc_next. Last three bits denote the kind of branching - 000 for normal branching (pc = pc + 4), 001 for unconditional branching (b), 010 for branch register (br), 011 for comparison based branching (bltz, bz and bnz), 100 for carry based branching and 101 for

branch and link (bl). The higher two bits determine the specifics of the branching specified by the lower three bits. For comparison based branching, these two bits determine what kind of comparison is to be done - 00 for rs < 0 comparison, 01 for rs == 0 comparison and 10 for rs != 0 comparison. For carry based branching, they determine the kind of carry comparison - 00 for carry == 1 comparison and 01 for carry == 0 comparison.

op	**opcode**	**func**	**dmem_ enable**	**demem_ write_ enable**	**reg_write_ mux_ctrl**	**br_op**
add	000000	000000	0	0	10	0000
comp	000000	000001	0	0	10	00000
addi	000001	000000	0	0	10	00000
compi	000001	000001	0	0	10	00000
and	001010	000000	0	0	10	00000
xor	001010	000001	0	0	10	00000
shll	010100	000000	0	0	10	00000
shrl	010100	000001	0	0	10	00000
shra	010100	000010	0	0	10	00000
shllv	010101	000000	0	0	10	00000
shrlv	010101	000001	0	0	10	00000
shrav	010101	000010	0	0	10	00000
lw	011110	NA	1	0	01	00000
sw	011111	NA	1	1	00	00000
b	101000	NA	0	0	00	00001
br	101100	NA	0	0	00	00010
bltz	101101	NA	0	0	00	00011

bz	101110	NA	0	0	00	01011
bnz	101111	NA	0	0	00	10011
bl	101001	NA	0	0	00	00101
bcy	101010	NA	0	0	00	00100
bncy	101011	NA	0	0	00	01100
diff	110010	000000	0	0	10	00000