

CS39001 - COMPUTER ORGANIZATION LABORATORY

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Design of the KGP-miniRISC Processor

Group Number - 12

Authors:

Ashish Rekhani (20CS10010)

Kartik Pontula (20CS10031)

Date: November 9, 2022

Introduction

In this final assignment we design the processor KGP-miniRISC with the provided instruction set architecture.

The Instruction Encoding Format

R-format Instructions

Bit Ranges:	[31-26]	[25-21]	[20-16]	[15-6]	[5-0]
Fields:	opcode 6 bits	rs 5 bits	rt 5 bits	shamt 5+5 bits (leftmost 5 bits are redundant)	funct 6 bits

Figure 1: R-format encoding

opcode and *funct* uniquely identify the operation used.

rs and *rt* store the indices of the source and terminal registers. Range: 0 to 31.

shamt consists of 5 redundant bits (15-11) followed by the shift amount (10-6) as an unsigned integer. Range: 0 to 31.

This format encodes the following operations:

- *add, comp, and, xor, shllv, shrlv, shrav, diff.* (*shamt* is unused here)
- *shll, shrl, shra.* (*rt* is unused here)

I-format Instructions

Bit Ranges:	[31-26]	[25-21]	[20-16]	[15-0]
Fields:	opcode 6 bits	rs 5 bits	rt 5 bits	immediate 16 bits

Figure 2: I-format encoding

Here *immediate* stores a signed integer value.

Range: -2^{15} to $2^{15} - 1$ i.e. -32,768 to 32,767.

This format encodes:

-
- *addi, compi*. (*rt* is unused here)
 - *lw, sw*.

B-format Instructions

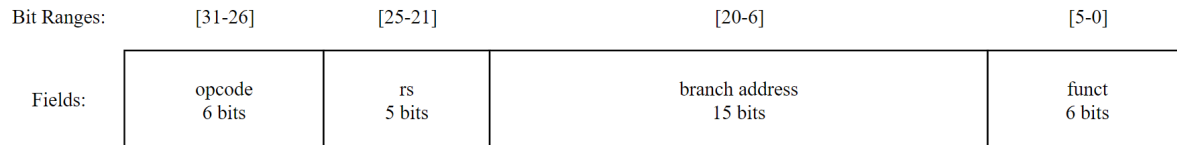


Figure 3: B-format encoding

branch address stores the address provided in the instruction.
This format encodes:

- *b, bl, bcy, bncy*. (*rs* is unused here)
- *br, bltz, bz, bnz*.

Assignment of *opcode* and *funct*

Operation	<i>opcode</i>	<i>funct</i>
<i>add</i>	000000	000000
<i>comp</i>	000000	000001
<i>diff</i>	000000	000010
<i>and</i>	000001	000000
<i>xor</i>	000001	000001
<i>shll</i>	000010	000000
<i>shrl</i>	000010	000001
<i>shllv</i>	000010	000010
<i>shrlv</i>	000010	000011
<i>shra</i>	000010	000100
<i>shrav</i>	000010	000101
<i>br</i>	000011	000000
<i>bltz</i>	000011	000001
<i>bz</i>	000011	000010
<i>bnz</i>	000011	000011
<i>b</i>	000100	000000
<i>bcy</i>	000100	000001
<i>bncy</i>	000100	000010
<i>lw</i>	000101	XXXXXX
<i>sw</i>	000110	XXXXXX
<i>addi</i>	000111	XXXXXX
<i>compi</i>	001000	XXXXXX
<i>bl</i>	001001	000000

Table 1: *opcode* and *funct* for each operation

<i>opcode</i>	ALUSrc	ALUOp	MemToReg	RegWrite	MemRead	MemWrite	Branch
000000	1	001	00	10	0	0	00
000001	1	010	00	10	0	0	00
000010	1	011	00	10	0	0	00
000011	1	000	00	00	0	0	01
000100	1	000	00	00	0	0	10
000101	0	100	01	11	1	0	00
000110	0	101	00	00	0	1	00
000111	0	110	00	10	0	0	00
001000	0	111	00	10	0	0	00
001001	1	000	10	01	0	0	11

Table 2: Main Control Unit

ALUSrc - Controls what the 2nd input to the ALU should be (immediate value if 0, data of 2nd register if 1)

ALUOp - This is fed to the ALU Control Unit along with *funct* to determine the control signal to be sent to the ALU.

MemToReg - Determines which data is to be written to the register file (00 to choose ALU Output, 01 for RAM output, 10 for PC+4)

RegWrite - This signal chooses when and where data should be written for the register file (00 to not write, 01 to write to register 31, 10 for *rs*, 11 for *rs*)

MemRead & MemWrite - Truth value to decide whether RAM should be read from or written to.

Branch - Used to decide which instruction to branch to next.

ALUOp	<i>funct</i>	Control Signal
001	000000	0000
001	000001	0001
001	000010	0010
010	000000	0011
010	000001	0100
011	000000	0101
011	000001	0110
011	000010	1001
011	000011	1000
011	000100	0111
011	000101	1010
110	XXXXXX	0000
111	XXXXXX	0001

Table 3: ALU Control Unit

Processor Datapath

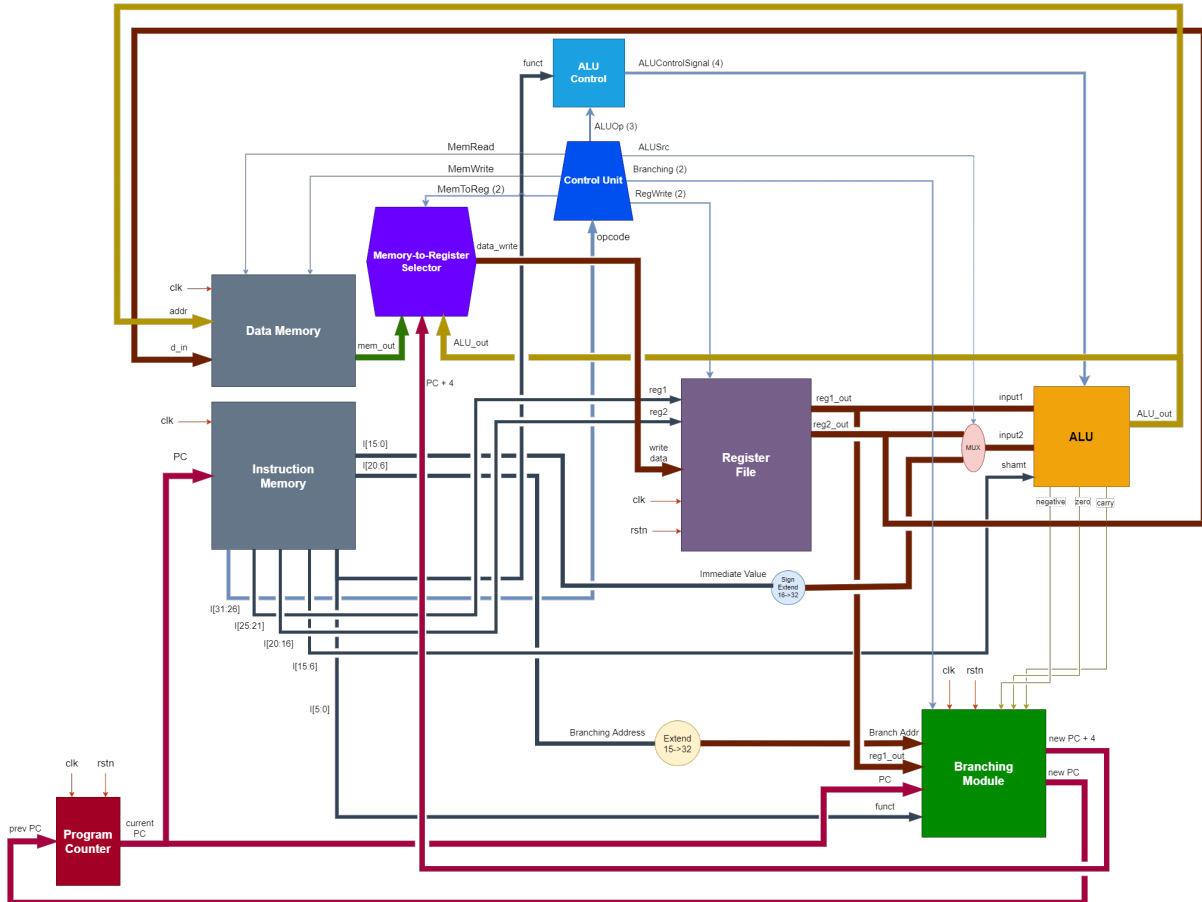


Figure 4: Datapath of the KGP-miniRISC Processor.

ALU Diagram

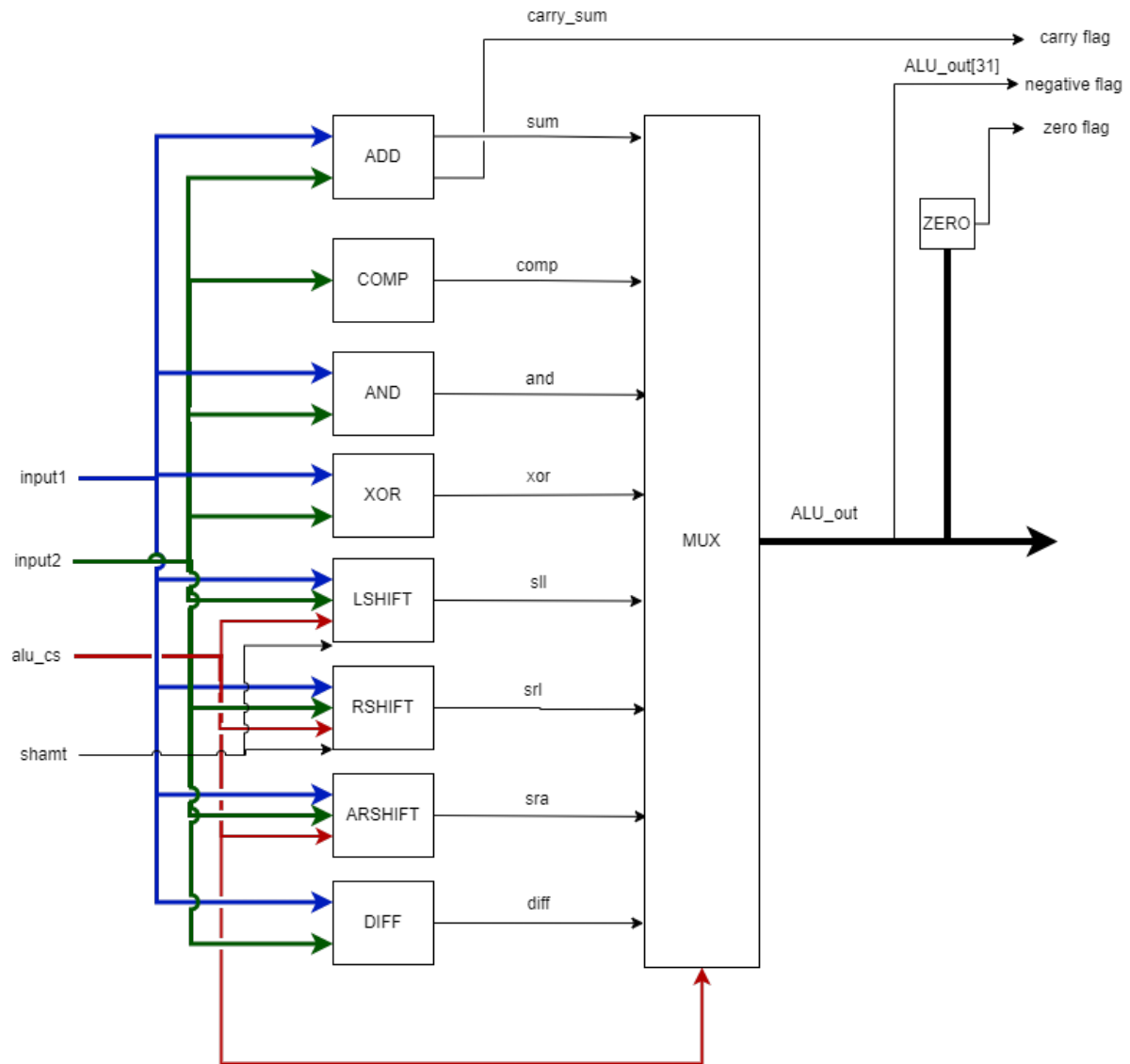


Figure 5: ALU Design.