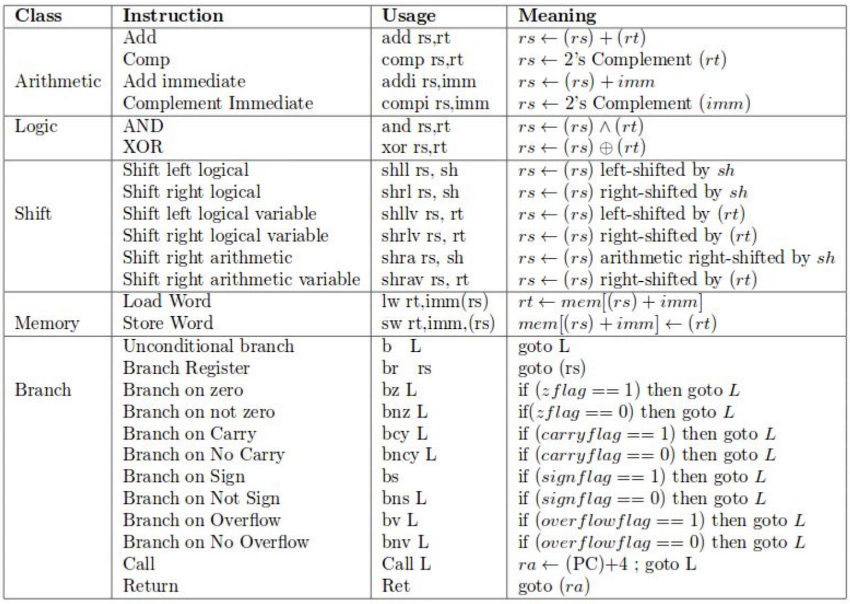
KGPRISC (Single Cycle CPU)

User Guide

- Nikhil Nayan Jha (16CS30022)
- Sarthak Chakraborty (16CS30044)

GROUP 32

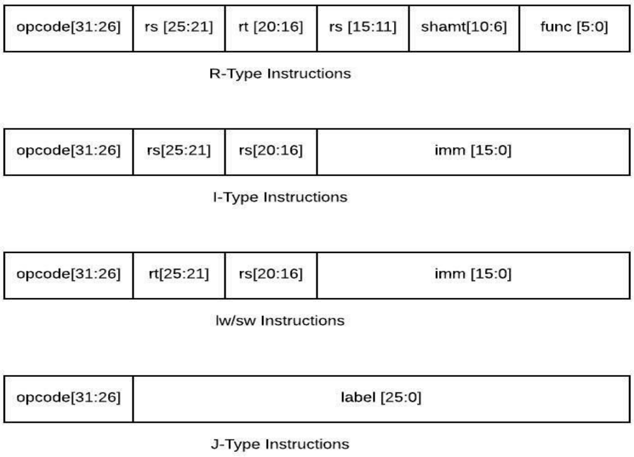
The following document is a guide for the Single Cycle CPU (RISC architecture) named KGPRISC that we designed. This document lists the Instruction Set Architecture (ISA) that the processor supports, format for instructions, instruction encoding and the architecture design.



Class	Instruction	Usage	Meaning
Arithmetic	Add	add rs,rt	$rs \leftarrow (rs) + (rt)$
	Comp	comp rs,rt	$rs \leftarrow$ 2's Complement (rt)
	Add immediate	addi rs,imm	$rs \leftarrow (rs) + imm$
	Complement Immediate	compi rs,imm	$rs \leftarrow$ 2's Complement (imm)
Logic	AND	and rs,rt	$rs \leftarrow (rs) \wedge (rt)$
	XOR	xor rs,rt	$rs \leftarrow (rs) \oplus (rt)$
Shift	Shift left logical	shll rs, sh	$rs \leftarrow (rs)$ left-shifted by sh
	Shift right logical	shrl rs, sh	$rs \leftarrow (rs)$ right-shifted by sh
	Shift left logical variable	shllv rs, rt	$rs \leftarrow (rs)$ left-shifted by (rt)
	Shift right logical variable	shrlv rs, rt	$rs \leftarrow (rs)$ right-shifted by (rt)
	Shift right arithmetic	shra rs, sh	$rs \leftarrow (rs)$ arithmetic right-shifted by sh
	Shift right arithmetic variable	shrav rs, rt	$rs \leftarrow (rs)$ right-shifted by (rt)
Memory	Load Word	lw rt,imm(rs)	$rt \leftarrow mem[(rs) + imm]$
	Store Word	sw rt,imm,(rs)	$mem[(rs) + imm] \leftarrow (rt)$
Branch	Unconditional branch	b L	goto L
	Branch Register	br rs	goto (rs)
	Branch on zero	bz L	if $(zflag == 1)$ then goto L
	Branch on not zero	bnz L	if$(zflag == 0)$ then goto L
	Branch on Carry	bcy L	if $(carryflag == 1)$ then goto L
	Branch on No Carry	bncy L	if $(carryflag == 0)$ then goto L
	Branch on Sign	bs	if $(signflag == 1)$ then goto L
	Branch on Not Sign	bns L	if $(signflag == 0)$ then goto L
	Branch on Overflow	bv L	if $(overflowflag == 1)$ then goto L
	Branch on No Overflow	bnv L	if $(overflowflag == 0)$ then goto L
	Call	Call L	$ra \leftarrow$ (PC)+4 ; goto L
	Return	Ret	goto (ra)

KGPRISC follows a set of instructions that are listed in the ISA shown beside. It can support only these instructions and any instruction different from the ones listed needs to be written in terms of these.

Eg: **sub rs, rt** can be written as **comp rt, rt** followed by **add rs, rt.**



opcode[31:26]	rs [25:21]	rt [20:16]	rs [15:11]	shamt[10:6]	func [5:0]

R-Type Instructions

opcode[31:26]	rs[25:21]	rs[20:16]	imm [15:0]

I-Type Instructions

opcode[31:26]	rt[25:21]	rs[20:16]	imm [15:0]

lw/sw Instructions

opcode[31:26]	label [25:0]

J-Type Instructions

Instruction format that KGPRISC follows are shown here. R-Type instructions include all the arithmetic and the logical and shift instructions. I-Type instructions are addi, and compi. J-Type instructions are the branch instructions.

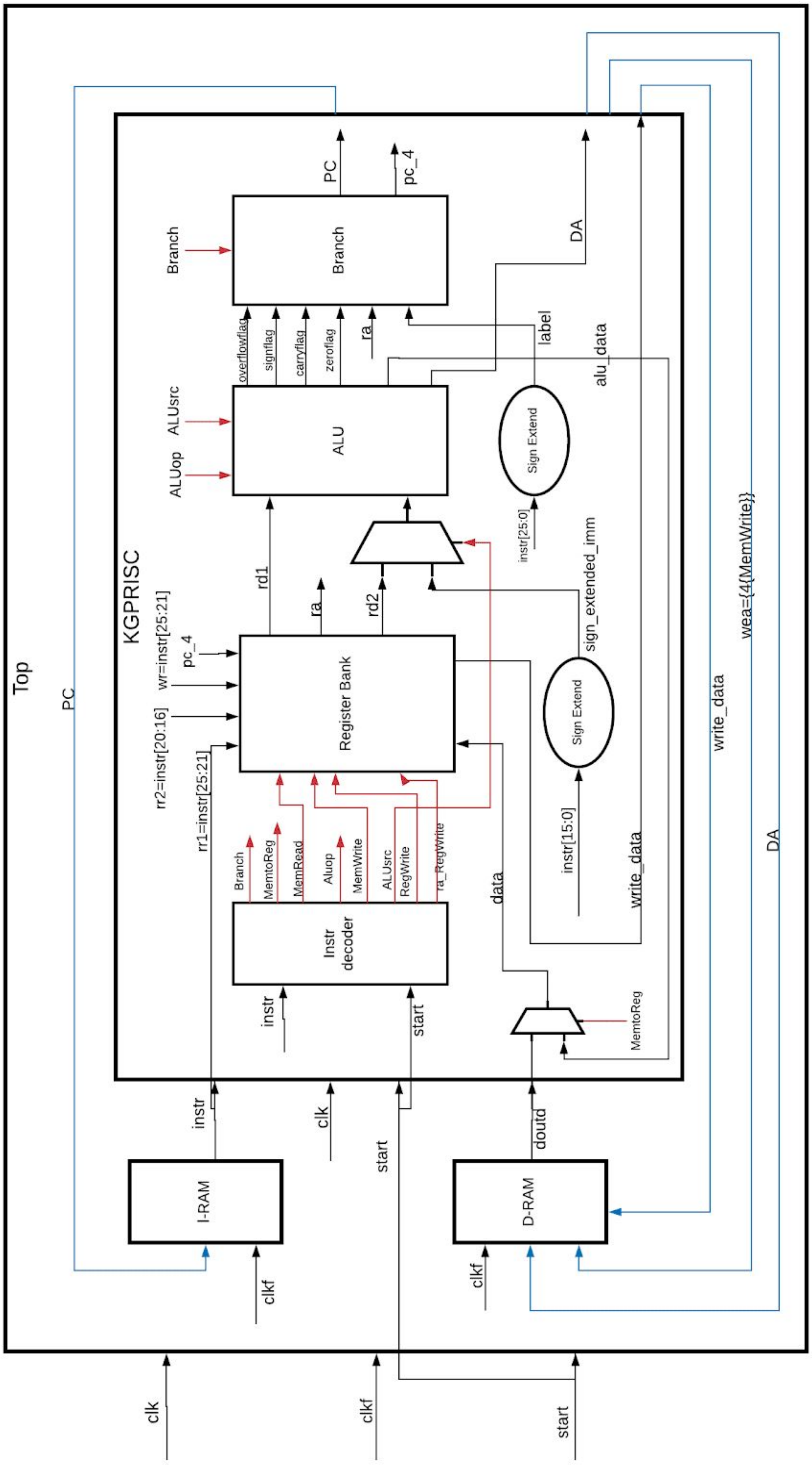
The instruction encoding followed for the instructions listed in the ISA for KGPRISC are shown below.

Instr	add	comp	and	xor	shll	shrl	shllv	shrlv	shra	shrav
opcode	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
func	000000	000001	000010	000011	000100	000101	000110	000111	001000	001001

Instr	addi	compi	lw	sw	b	br	bz	bnz
opcode	000100	000101	000010	000011	010000	010001	010010	010011

Instr	bcy	bncy	bs	bns	bv	bnv	Call	Ret
opcode	010100	010101	010110	010111	011000	011001	000110	000111

KGPRISC uses 32 registers



Top
KGPRISC
PC
I-RAM
D-RAM
instr
clk
start
doutd
clkf
clk
clkf
start
Instr decoder
Branch
MemtoReg
MemRead
Aluop
MemWrite
ALUsrc
RegWrite
ra_RegWrite
rr1=instr[25:21]
rr2=instr[20:16]
wr=instr[25:21]
pc_4
Register Bank
rd1
ra
rd2
ALUop
ALUsrc
ALU
overflowflag
signflag
carryflag
zeroflag
ra
Branch
Branch
PC
pc_4
DA
label
alu_data
Sign Extend
instr[25:0]
Sign Extend
sign_extended_imm
instr[15:0]
data
write_data
write_data
wea={4{MemWrite}}
DA
MemtoReg