DOCUMENTATION FOR KGP-RISC PROCESSOR

K P S M Surya 19CS10041

Pavan Sai 19CS10023

Group 51

Class	Instruction	Op code	Function code	Usage	Meaning
Arithmetic	Add	0000	0000	add rs,rt	rs ← (rs) + (rt)
	Comp		0001	comp rs,rt	rs ← 2's Complement (rs)
	Add immediate	0001	0000	addi rs,imm	rs ← (rs) + imm
	Complement Immediate		0001	compi rs,imm	rs ← 2's Complement (imm)
Logic	AND	0000	0010	and rs,rt	rs ← (rs) ∧ (rt)
	XOR		0011	xor rs,rt	rs ← (rs) ⊕ (rt)
Shift	Shift left logical	0001	0010	shll rs, sh	rs ← (rs) left-shifted by sh
	Shift right logical		0011	shrl rs, sh	rs ← (rs) right-shifted by sh
	Shift right arithmetic		0100	shra rs, sh	rs ← (rs) arithmetic right-shifted by sh
	Shift left logical variable	0000	0100	shllv rs, rt	rs ← (rs) left-shifted by (rt)
	Shift right logical variable		0101	shrlv rs, rt	rs ← (rs) right-shifted by (rt)
	Shift right arithmetic variable		0110	shrav rs, rt	rs ← (rs) right-shifted by (rt)
Memory	Load Word	0010	- -	lw rt,imm(rs)	rt ← mem[(rs) + imm]
	Store Word	0011	- -	sw rt,imm,(rs)	mem[(rs) + imm] ← (rt)
Branch	Unconditional branch	0100	000	b L	goto L
	Branch Register	0101	- -	br rs	goto (rs)
	Branch on less than 0	0110	000	bltz rs,L	if(rs) < 0 then goto L
	Branch on flag zero		001	bz rs,L	if (rs) = 0 then goto L
	Branch on flag not zero		010	bnz rs,L	if(rs) != 0 then goto L
	Branch and link	0100	001	bl L	goto L; 31 ← (PC)+4
	Branch on Carry		010	bcy L	goto L if Carry = 1
	Branch on No Carry		011	bncy L	goto L if Carry = 0

Opcode	Class	Functions
0000	R type	add, comp, and, xor, shllv, shrlv, shrav
0001	Immediate	addi, compi, shll, shrl, shra
0010	Load	lw
0011	Store	sw
0100	Branch 1	b, bl, bcy, bncy
0101	Branch 2	br
0110	Branch 3	bltz, bz, bnz

R type : 0000

Instruction format :

Opcode	rs	rt	function	extension
4 bits	5 bits	5 bits	4 bits	14 bits

Functions:

Function	Function code
add	0000
comp	0001
and	0010
xor	0011
shllv	0100
shrlv	0101
shrav	0110

Immediate type : 0001

Instruction format :

Op code	rs	immediate	function	extension
4 bits	5 bits	16 bits	4 bits	3 bits

Functions:

Function	Function code
addi	0000
compi	0001
shll	0010
shrl	0011
shra	0100

Load type : 0010

Instruction format :

Opcode	**rs**	**rt**	**immediate**	**extension**
4 bits	5 bits	5 bits	16 bits	2 bits

Functions:

Function	**Function code**
lw	**---**

Store type : 0011

Instruction format :

Opcode	**rt**	**rs**	**immediate**	**extension**
4 bits	5 bits	5 bits	16 bits	2 bits

Branch type 1 : 0100

Opcode	immediate	function	extension
4 bits	16 bits	4 bits	8 bits

Function	Function code
b	0000
bl	0001
bcy	0010
bncy	0011

Branch type 2 : 0101

Instruction format :

Opcode	rs	extension
4 bits	5 bits	23 bits

Functions:

Function	Function code
br	-

Branch type 3 : 0110

Instruction format :

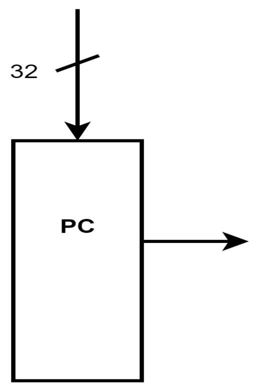
Opcode	rs	immediate	Function code	extension
4 bits	5 bits	16 bits	4 bits	3 bits

Functions:

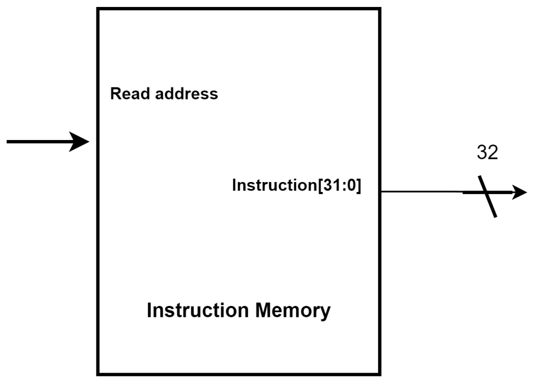
Function	Function code
bltz	0000
bz	0001
bnz	0010

Datapath elements:

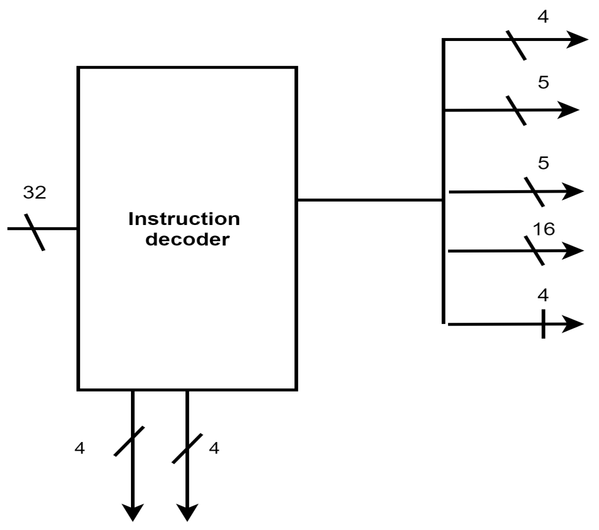
Program counter:



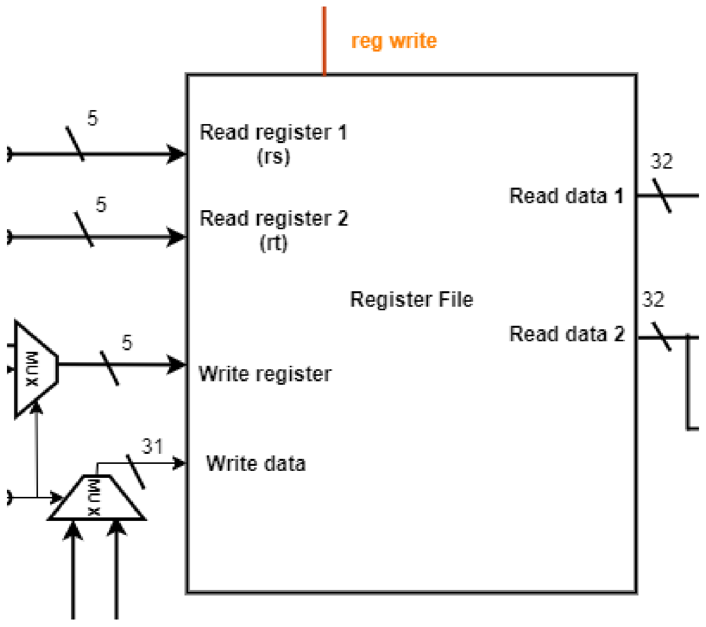
Instruction memory:



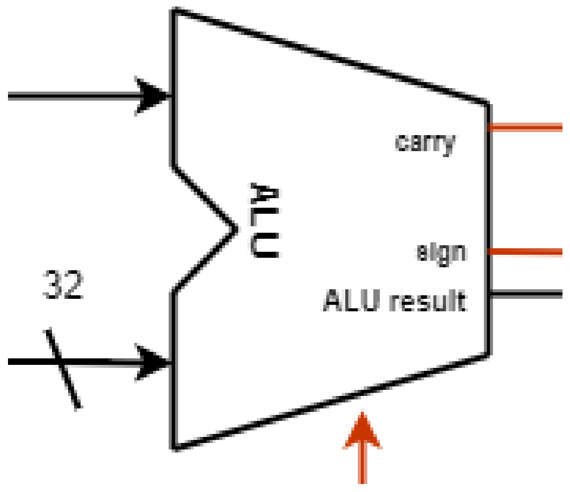
Instruction decoder:



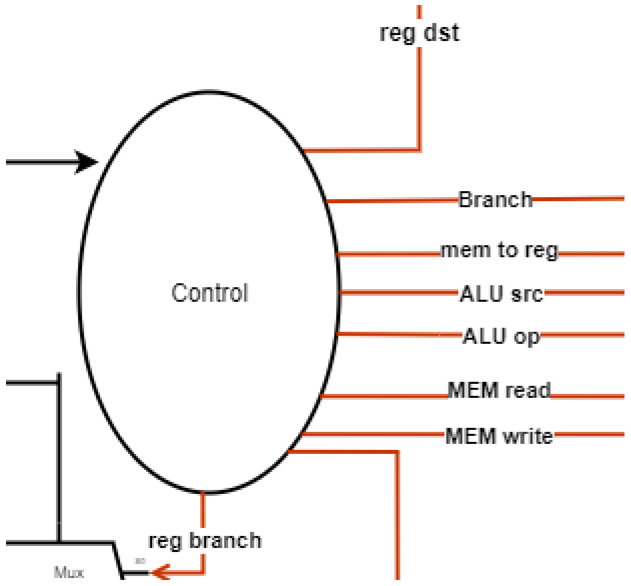
regfile:



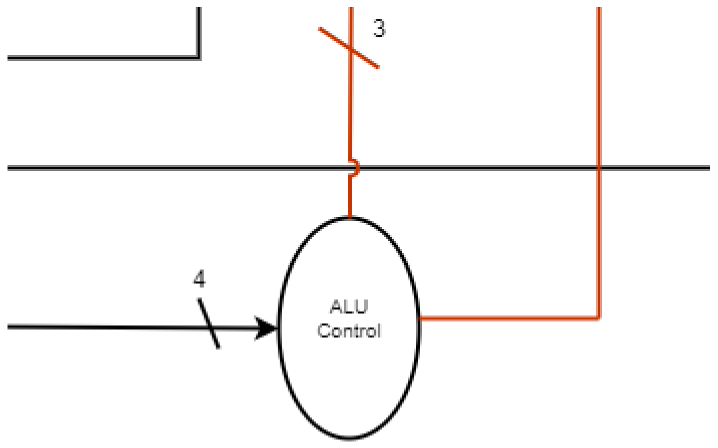
ALU:



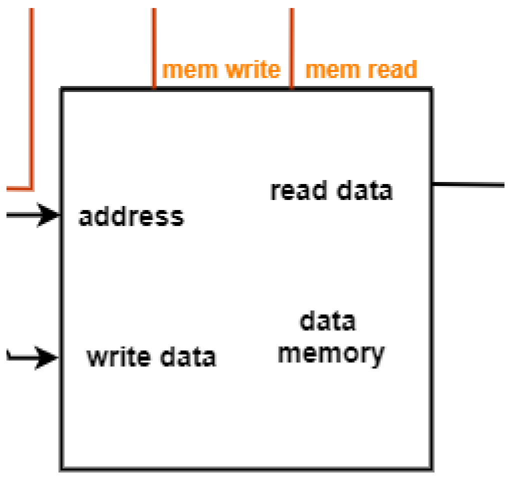
Controller:



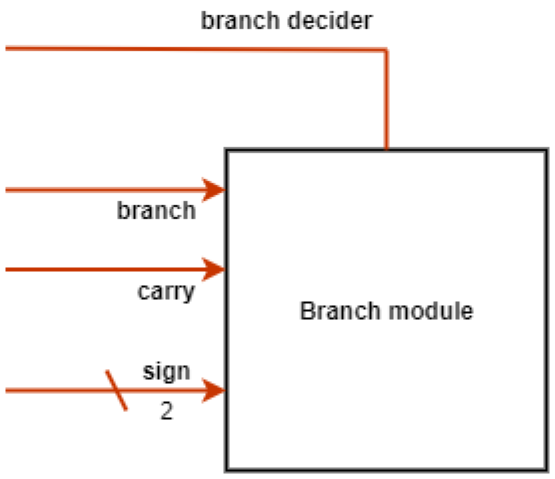
ALU control:



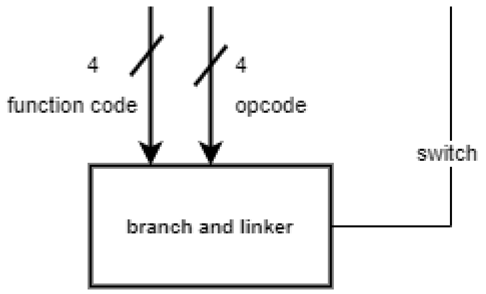
datamemory:



branchmodule :



Linker :



Truth table for control:

Opcode	ALUOp	ALUsrc	Branch	Regdst	Memtoreg	Regwrite	Memread	Memwrite	Reg Branch
R type (0000)	001	0	0	0	0	1	0	0	0
Immediate (0001)	011	1	0	0	1	0	0	0	0
Load (0010)	000	1	0	1	1	1	1	0	0
Store (0011)	000	1	0	0	_	0	0	1	0
Branch1 (0100)	100	_	1	0	0	0	0	0	0
Branch2 (0101)	000	_	1	0	0	0	0	0	1
Branch3 (0110)	101	_	1	0	0	0	0	0	0

Truth table for ALU control:

Opcode	Function code	Result (ALU)	Flag (Branch)
R type (0000)	0000	000	000
	0001	001	000
	0010	010	000
	0011	011	000
	0100	101	000
	0101	111	000
	0110	110	000
Immediate (0001)	0000	000	000
	0001	001	000
	0010	101	000
	0011	111	000
	0100	110	000
Load (0010)	---	000	000
Store (0011)	---	000	000
Branch 1 (0100)	0000	000	110
	0001	000	110
	001-0	000	001
	0011	000	010
Branch 2 (0101)	---	000	110
Branch 3 (0110)	0000	000	011
	0001	000	100
	0010	000	101

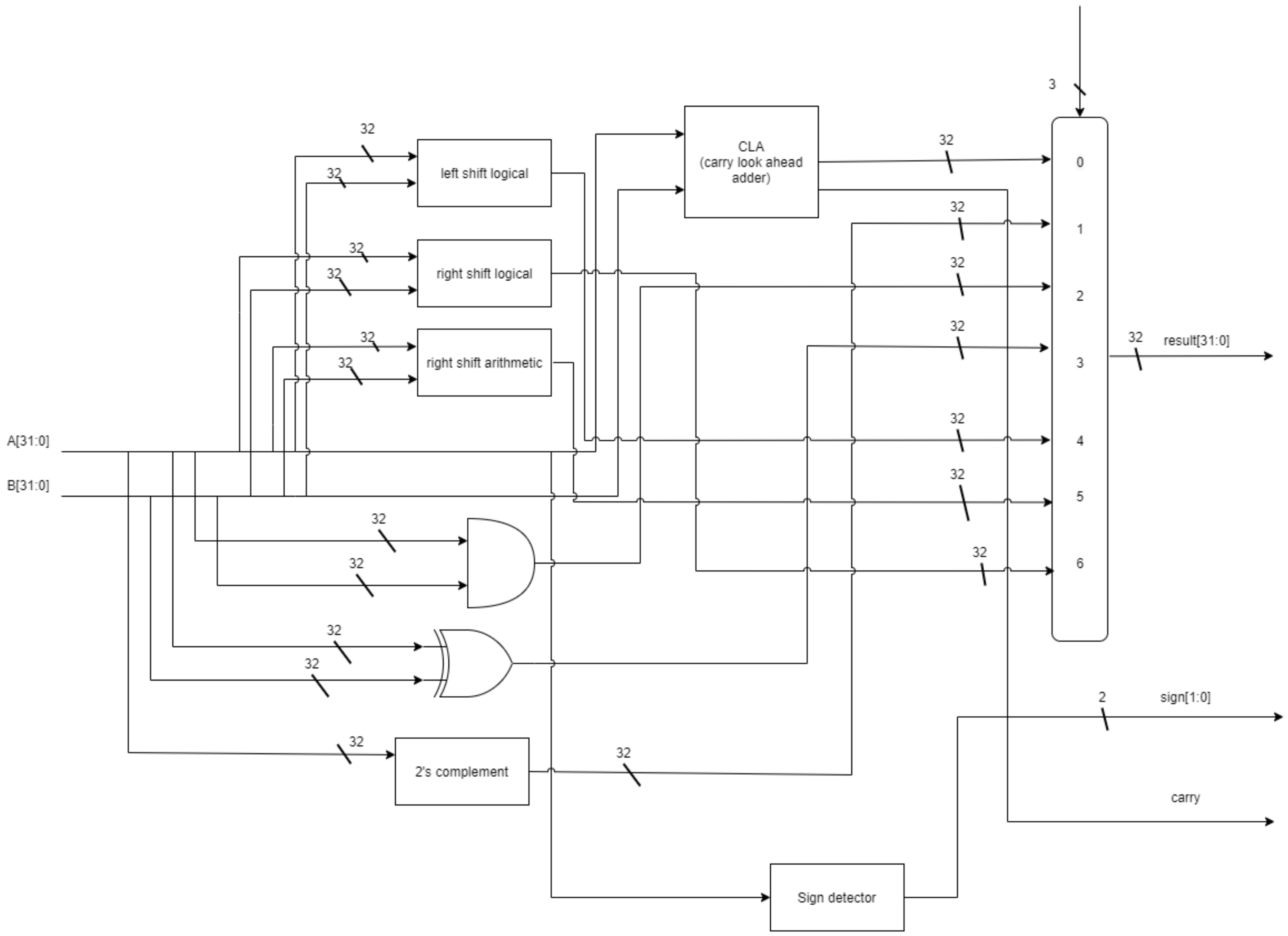
Result (ALU):

000: add,
001: 2’s complement,
010: and, 011: xor,
101: logical left shift,
110: arithmetic right shift,
111: logical right shift

Flag (Branch):

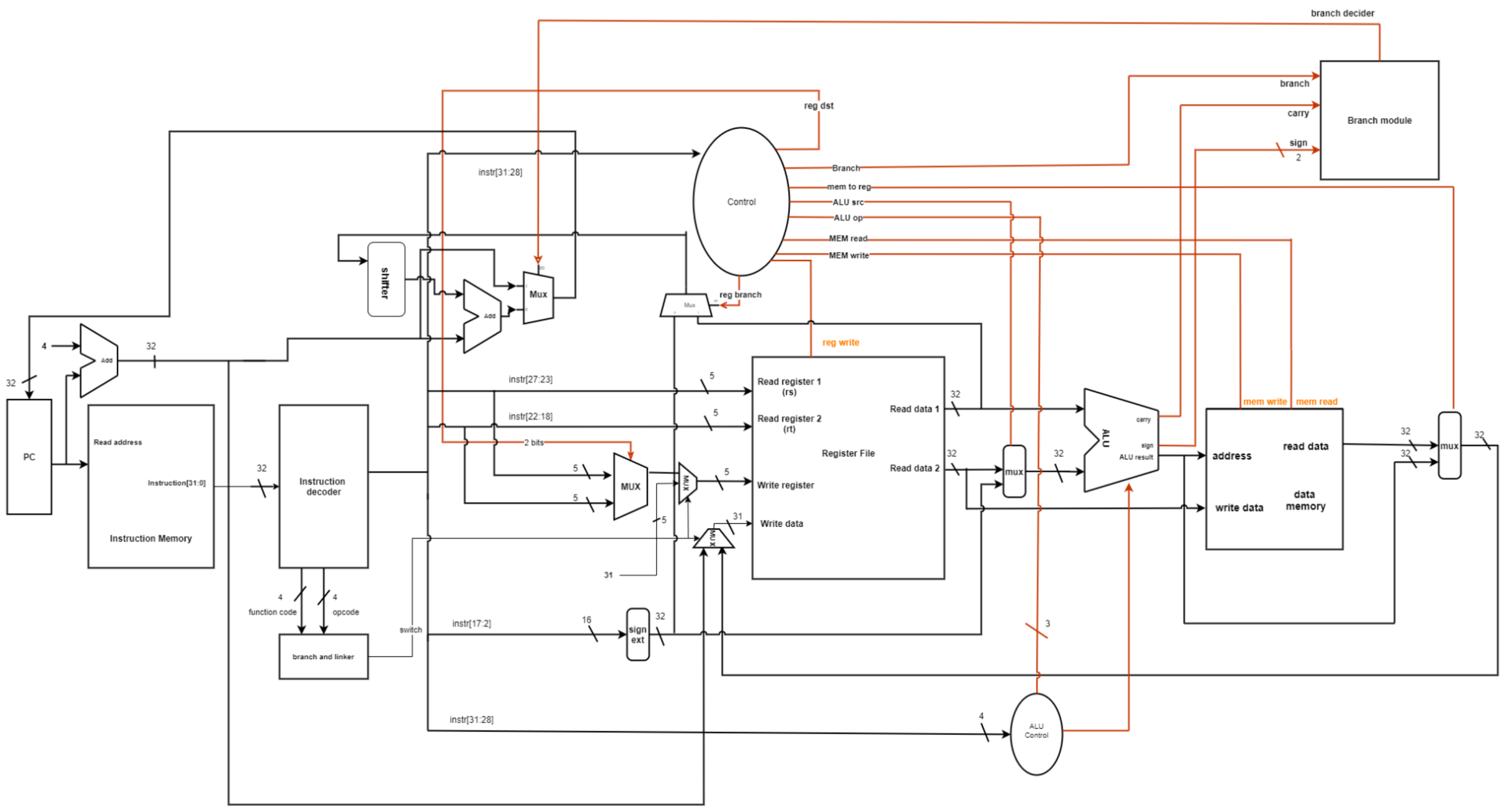
000: no branching,
001: branch if carry,
010: branch if not carry,
011: branch if less than,
100: branch if zero,
101: branch if not zero

ALU:



3
32
left shift logical
32
CLA
(carry look ahead adder)
32
0
32
right shift logical
32
1
32
2
32
right shift arithmetic
32
3
32
result[31:0]
A[31:0]
4
B[31:0]
5
32
6
32
32
32
sign[1:0]
2
32
2's complement
32
carry
Sign detector

Complete datapath with control signals for a single execution cycle :



Extension scope :

We allocated 4 bits for the opcode which implies there is scope for **16** different types of operations (opcodes), we already used 7 opcodes hence there is room for **9** more opcodes. For each opcode if we want to accommodate more operations of the particular opcode there is scope as 4 bits are allocated for function code (16 possible function codes) :

Opcode : 0000	
Number of function codes used :	7
Remaining for extension :	9
Opcode : 0001	
Number of function codes used :	5
Remaining for extension :	11
Opcode : 0100	
Number of function codes used :	4
Remaining for extension :	12
Opcode : 0110	
Number of function codes used :	3
Remaining for extension :	13

For some opcodes (load, store, reg branch) there is no function code. But for extension purposes there are some bits left at the end.