KGP_RISC Processor design DOCUMENTATION
Group – 64

Group Members:
1. Jatoth charan sai – 19CS10035
2. Tirupati Suguna Bhaskar – 19CS10063

Instruction Format

- Instruction set is divided into 3 types
    1. R – type
    2. Immediate – type
    3. Branch – type(i,ii)

R-type:

Instruction Format

OP-code	Register-1(rs)	Register-2(rt)	Shift-amount (sh)	N/A XXXXX	Function
[31:26]	[25:21]	[20:16]	[15:11]	[10:6]	[5:0]

OP-codes and function codes

Instruction	Action	OP-code[31:26]	Reg-1[25:21]	Reg-2[20:16]	Shift(sh) [15:11]	Function
$add\ \ rs\ \ rt$	$rs \leftarrow rs + rt$	000000	rs	rt	N/A	000000
$comp\ \ rs\ \ rt$	$rs \leftarrow 2's\ complement(rt)$	000000	rs	rt	N/A	000001
$and\ \ rs\ \ rt$	$rs \leftarrow rs\ \&\ rt$	000000	rs	rt	N/A	000010
$xor\ \ rs\ \ rt$	$rs \leftarrow rs\ \hat{}\ rt$	000000	rs	rt	N/A	000011
$shll\ \ rs\ \ sh$	$rs \leftarrow rs \ll sh$	000001	rs	N/A	sh	000100
$shrl\ \ rs\ \ sh$	$rs \leftarrow rs \gg sh$	000001	rs	N/A	sh	000101
$shllv\ \ rs\ \ rt$	$rs \leftarrow rs \ll rt$	000000	rs	rt	N/A	000100
$shrlv\ \ rs\ \ rt$	$rs \leftarrow rs \gg rt$	000000	rs	rt	N/A	000101
$shra\ \ rs\ \ sh$	$rs \leftarrow rs\ (ARS)\ sh$	000001	rs	N/A	sh	000110
$shrav\ \ rs\ \ rt$	$rs \leftarrow rs(ARS)\ rt$	000000	rs	rt	N/A	000110

NOTE: $(ARS) \leftarrow >>> \leftarrow$ Arithmetic right shift

Immediate – type:

Instruction Format

OP-code	Register-1(rs)	Register-2(rt)	Immediate value
[31:26]	[25:21]	[20:16]	[15:0]

OP-codes and function codes

Instruction	Action	OP-code[31:26]	Reg-1[25:21]	Reg-2[20:16]	Imm[15:0]
$addi\ \ rs\ \ imm$	$rs \leftarrow rs + imm$	100010	rs	N/A	imm
$compi\ \ rs\ \ imm$	$rs \leftarrow 2's\ complement(imm)$	100011	rs	N/A	imm
$lw\ \ rs\ \ imm(rt)$	$rs \leftarrow MEM[imm + rt]$	100100	rs	rt	imm
$sw\ \ rs\ \ imm(rt)$	$MEM[imm + rt] \leftarrow rs$	100101	rs	rt	imm

NOTE: MEM ← memory

Branch – type:

Instruction Format

i.

OP-code	Register-1(rs)	Address
[31:26]	[25:21]	[20:0]

ii.

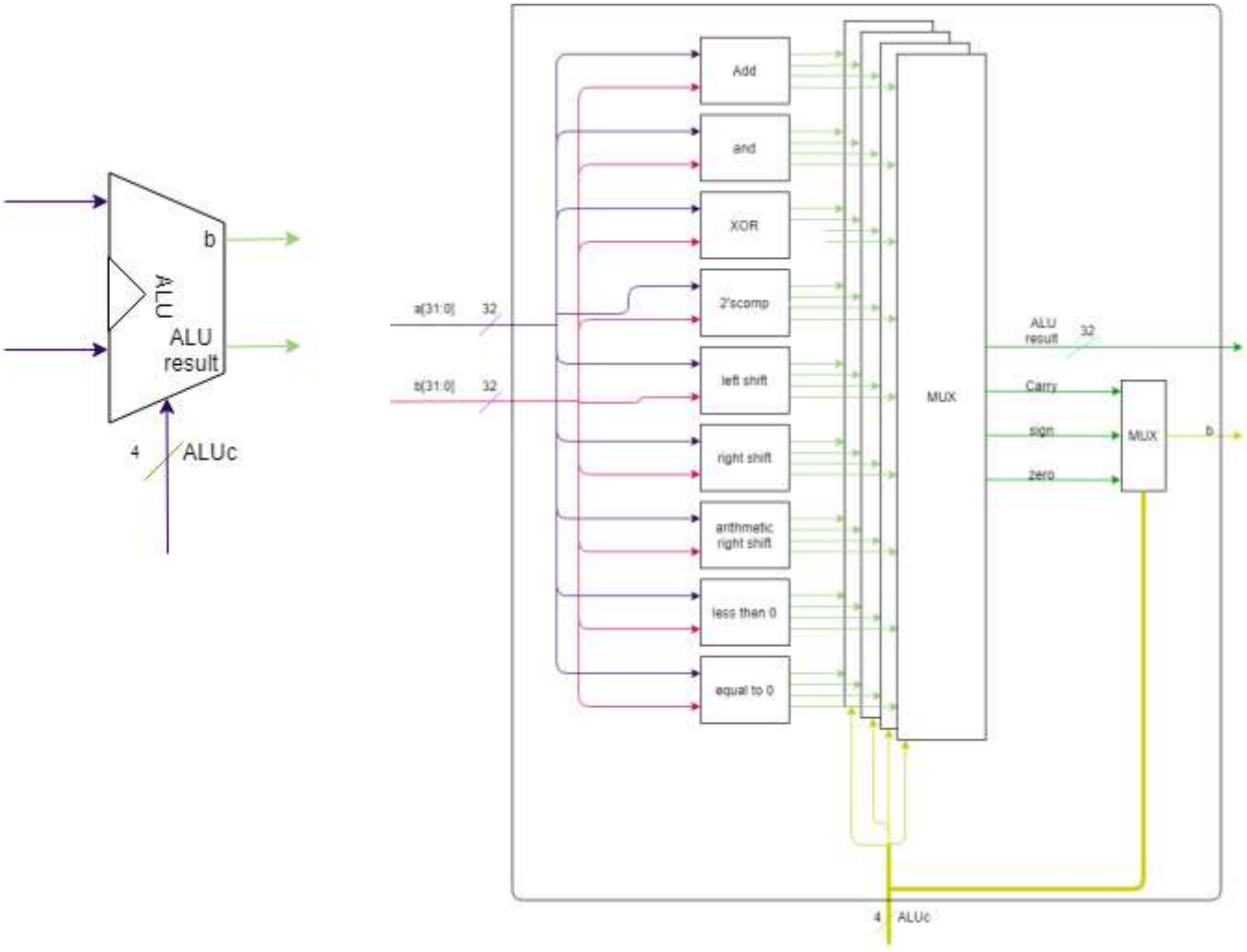
OP-code	Address
[31:26]	[25:0]

OP-codes and function codes

Instruction	Action	OP-code[31:26]	Reg[25:21]	Address[20:0]
i.				
$br\ \ rs$	$goto\ \ (rs)$	010001	rs	N/A
$bltz\ \ rs\ \ L$	$if(rs) < 0\ \ \ goto\ \ \ L$	010010	rs	L
$bz\ \ rs\ \ L$	$if(rs) = 0\ \ \ goto\ \ \ L$	010011	rs	L
$bnz\ \ rs\ \ L$	$if(rs) \neq 0\ \ \ goto\ \ \ L$	010100	rs	L
ii.(jump)			**Address[25:0]**	
$b\ \ L$	$goto\ \ \ L$	010000	L	
$bl\ \ L$	$goto\ \ \ L; 31 \leftarrow (PC) + 4$	010101	L	
$bcy\ \ \ L$	$goto\ \ \ L\ \ if\ Carry = 1$	010110	L	
$bncy\ \ \ L$	$goto\ \ \ L\ \ if\ Caray\ = 0$	010111	L	

NOTE: PC ← program counter

- Architecture for ALU block:



❖ Instruction set Architecture:

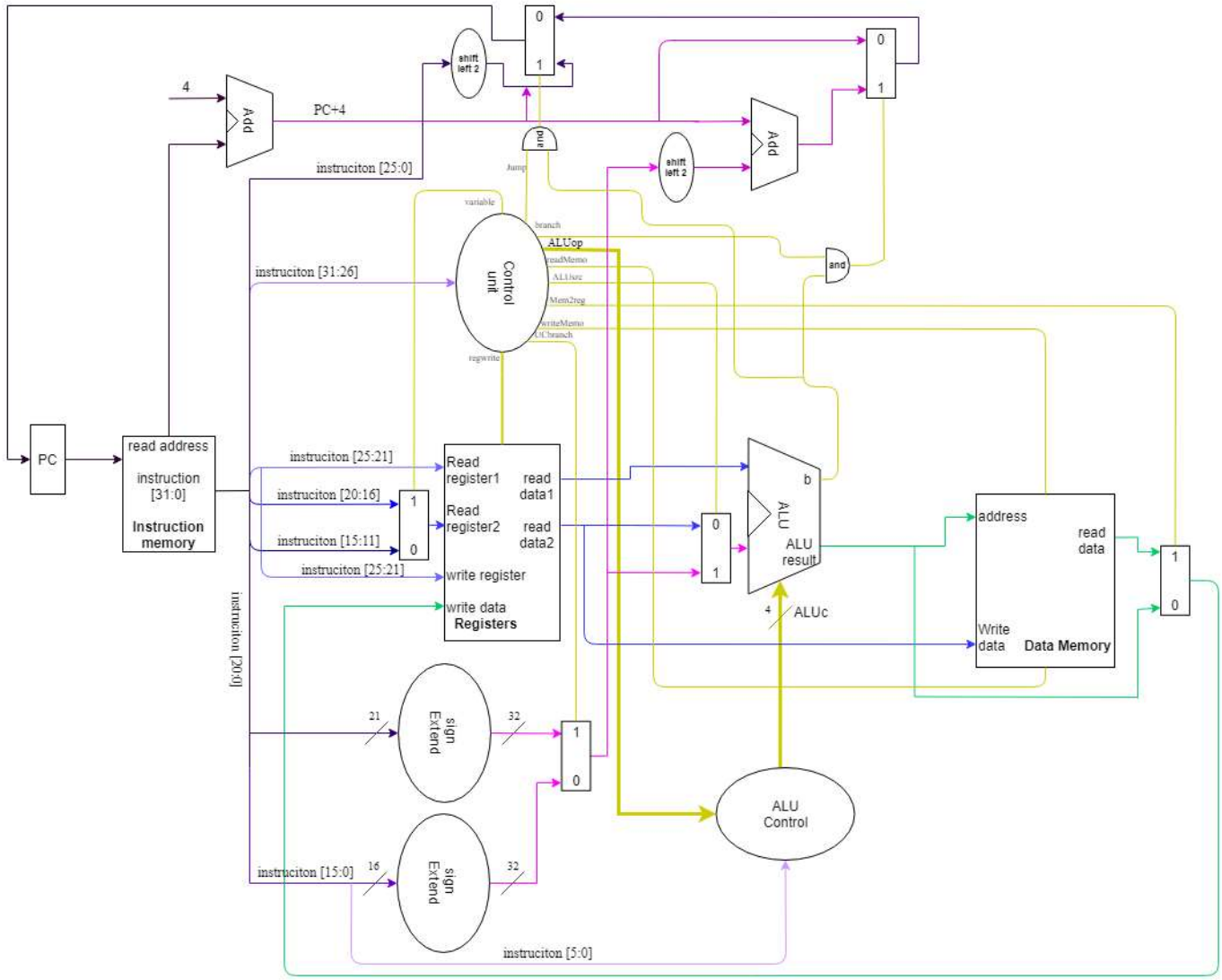


fig: CPU design architecture

Control Signals:

1. **Variable**: signal is to {read value }/{shift amount (for $shll, shrl, shra$)} to register.
2. **Regwrite:** enables if any result is to stored in register
3. **Writememo:** enables if any value is to be written into the memory
4. **Mem2reg:** enables if lw instruction is encountered
5. **ALUsrc:** enables if immediate-type instructions are encountered($addi/compi/lw/sw$)
6. **readMemo:** enables if any data is to be read from memory
7. **ALUop:** 4-bit control signals for ALU control
8. **UCbranch:** for branch instructions which have address of size 21 bits
9. **Branch:** for branch instruction's which has address of size 21 turn it will be enabled
10. **Jump:** for branch instruction's which has address of size 16 turn it will be enabled
11. **b:** this control signal is from ALU which ensures branch condition is satisfied or not.

Control signals and data path

1. **R-type instructions**
   a. OP-codes → 000000

Instruction	function	variable	Regwrite	writememo	Mem2reg	ALUsrc	readMemo	ALUop	UC-branch	branch	Jump
add	000000	1	1	0	0	0	0	0000	0	0	0
$comp$	000001	1	1	0	0	0	0	0001	0	0	0
and	000010	1	1	0	0	0	0	0010	0	0	0
xor	000011	1	1	0	0	0	0	0011	0	0	0

$shllv$	000100	1	1	0	0	0	0	0100	0	0	0
$shrlv$	000101	1	1	0	0	0	0	0101	0	0	0
$shrav$	000110	1	1	0	0	0	0	0110	0	0	0

Data path:

⇨ Read instructions→read register's 1,2 → control output→ input data1,2 in ALU→output of ALU→write ALU result in desired register.

b. Op-codes → 000001

Instruction	function	variable	Regwrite	writememo	Mem2reg	ALUsrc	readMemo	ALUop	$UC-branch$	branch	Jump
$shll$	000100	0	1	0	0	0	0	0100	0	0	0
$shrl$	000101	0	1	0	0	0	0	0101	0	0	0
$shra$	000110	0	1	0	0	0	0	0110	0	0	0

Data path:

⇨ Read instructions→ read register and shamt→ control signals output → input data 1,2 in ALU → output of ALU → write ALU result in desired register.

2. Immediate-type instructions

a.

instrc	opcode	variable	Regwrite	writememo	Mem2reg	ALUsrc	readMemo	ALUop	$UC-branch$	branch	Jump
$addi$	100010	X	1	0	0	1	0	0000	0	0	0
$compi$	100011	X	1	0	0	1	0	0001	0	0	0

Data path:

⇨ Read instructions→ read register 1 → control signals output→ immediate sign extension→ data 1,2 input in ALU unit→ output of ALU → write ALU result in desired register.

b.

instrc	opcode	variable	Regwrite	writememo	Mem2reg	ALUsrc	readMemo	ALUop	$UC-branch$	branch	Jump
sw	100100	1	0	1	0	1	0	0000	0	0	0
lw	100101	1	1	0	0	1	1	0000	0	0	0

Data path:

1. For load word.

⇨ Read instructions→ control signals output → read register 1,2 → immediate sign extension→ data 1,2 input in ALU unit→ output of ALU → memory read(lw) form memory of address ALUresult → write value form memory in desired register.

2. For store word.

⇨ Read instructions→ control signals output → read register 1,2 → immediate sign extension→ data 1,2 input in ALU unit→ output of ALU → write read data 2 in ALUresult address in memory.

3. Branch instructions:

a.

instrc	opcode	variable	Regwrite	writememo	Mem2reg	ALUsrc	readMemo	ALUop	$UC-branch$	branch	Jump
br	010001	X	0	0	0	X	0	0100	1	1	0
$bltz$	010010	X	0	0	0	X	0	0101	1	1	0
bz	010011	X	0	0	0	X	0	0110	1	1	0
bnz	010100	X	0	0	0	X	0	0111	1	1	0

instuc	***b***(signal from ALU)
br	1
$bltz$	$1\ if\ rs\ <\ 0\ else\ 0$

bz	$1\ if\ rs\ =\ 0\ else\ 0$
bnz	$1\ if\ rs \neq\ 0\ else\ 0$

Data path:

- ⇨ Read instructions→ control signals→ read register1 → immediate sign extension→ input data1 in ALU → signal b → PC update.

b.

instrc	opcode	variable	Regwrite	writememo	Mem2reg	ALUsrc	readMemo	ALUop	$UC - branch$	branch	Jump
b	010000	X	0	0	0	X	0	0000	X	0	1
bl	010101	X	0	0	0	X	0	0001	X	0	1
bcy	010110	X	0	0	0	X	0	0010	X	0	1
$bncy$	010111	X	0	0	0	X	0	0011	X	0	1

instuc	$\boldsymbol{b}$(signal from ALU)
b	1
bl	1
bcy	$1\ if\ Carry\ =\ 0\ else\ 0$
$bncy$	$1\ if\ Carry \neq\ 0\ else\ 0$

Data path:

- ⇨ Read instructions→ control signals→ 25 to 26 bit instruction → place remaining bits form PC+4→ input data1,2 in ALU → signal b → PC update.