REPORT

- K. SAI SURYA TEJA 16CS30015
- S. SASI BHUSHAN 16CS30032
- GROUP - 46

- Instruction encoding:

opcode[5:4]➡ **opcode[3:0]⬇**	00	01	10	11
0000	add	addi	lw	b
0001	comp	compi	sw	bz
0010	and	shll		bnz
0011	xor	shrl		bcy
0100	shllv	shra		bncy
0101	shrlv	br		bs
0110	shrav	ret		bns
0111				bv
1000				bnv
1001				call

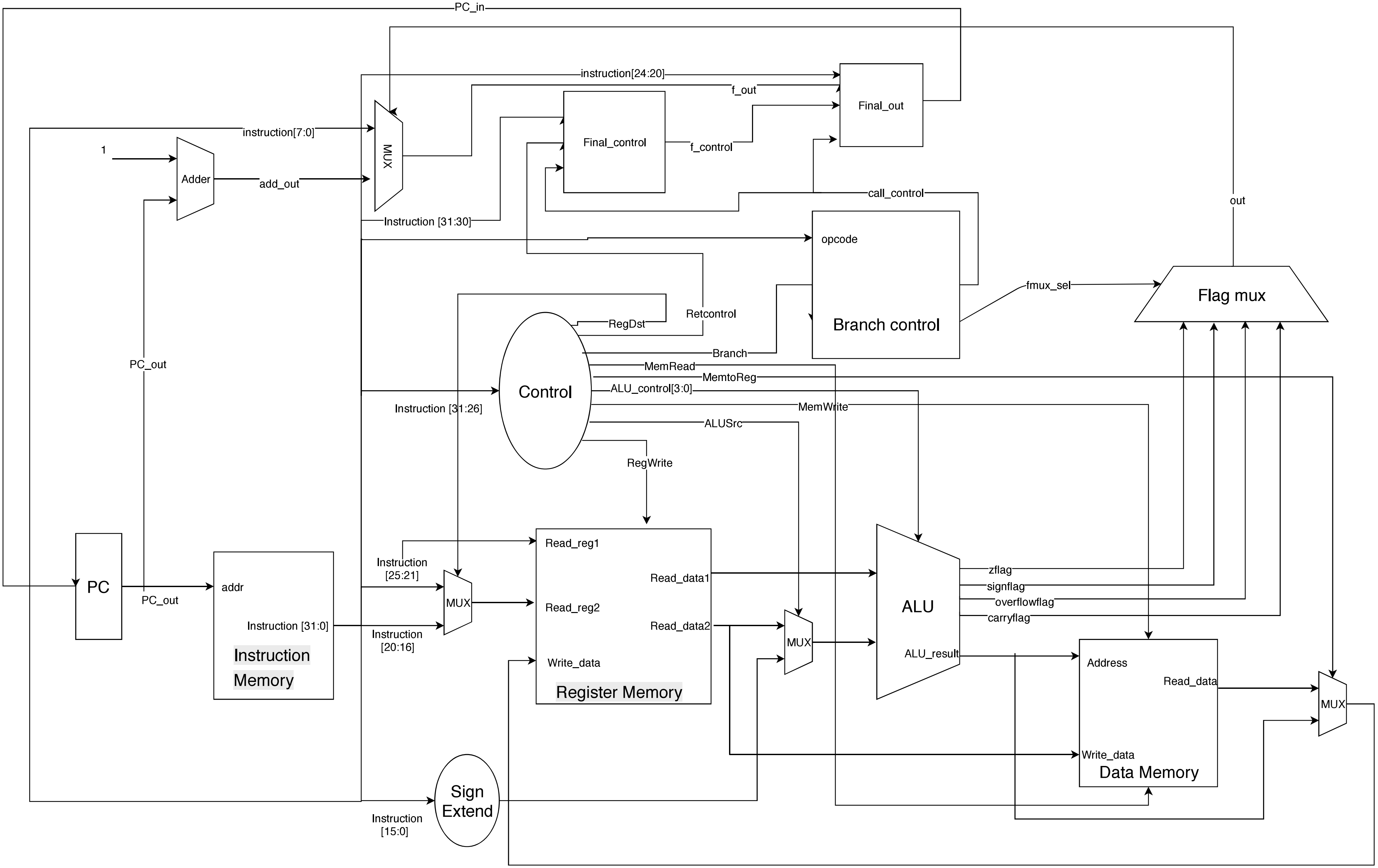
- Registers: The 0th register 00000 is used as $0 and its value is fixed at 32’b0. The 31st register 11111 is $ra. It is used for returning after a function call.

- Instruction format:

Opcode(6bit)	rs(5 bit)	rt(5 bit)	Offset/Address(16 bit)

If any of the 4 divisions and not valid for an instruction, it is set to 0.

- **The Output of the Bubble Sort is stored in the Data Memory (in Datamemory module instantiated as dm) in locations 1, 2, 3, 4 (in Dmemory of dm instant) in sorted order .**



KGP-RISC DATAPATH

KGP-RISC Instruction Set

Arithmetic Instructions

Instruction	Example	Meaning	Comments
Add	add rs,rt	rs ← (rs) + (rt)	
Add immediate	addi rs,imm	rs ← (rs) + imm	"Immediate" means a constant number
Comp	comp rs,rt	rs ← 2's Complement (rt)	The value in rs is complemented (2'cs complement).
Complement Immediate	compi rs,imm	rs ← 2's Complement (imm)	Immediate is a constant. Its 2's complement is stored in rs

Logical

Instruction	Example	Meaning	Comments
AND	and rs,rt	rs ← (rs) ∧ (rt)	Bitwise AND
XOR	xor rs,rt	rs ← (rs) ⊕ (rt)	Bitwise OR

Memory

Instruction	Example	Meaning	Comments
Load Word	lw rt,imm(rs)	rt ← mem[(rs) + imm]	Copy from memory to register
Store Word	sw rt,imm,(rs)	mem[(rs) + imm] ← (rt)	Copy from register to memory

Branch

Instruction	Example	Meaning	Comments
Unconditional branch	b L	goto L	Always goto L
Branch Register	br rs	goto (rs)	Goto the value in register rs
Branch on zero	bz L	if (zflag == 1) then goto L	Branch if ALU_result is 0
Branch on not zero	bnz L	if(zflag == 0) then goto L	Branch if ALU_result is 1
Branch on Carry	bcy L	if (carryflag == 1) then goto L	Branch if carry is 1
Branch on No Carry	bncy L	if (carryflag == 0) then goto L	Branch if carry is 0
Branch on Sign	Bs L	if (signflag == 1) then goto L	Branch if sign of ALU_result is negative

Branch on Not Sign	bns L	if (signflag == 0) then goto L	Branch if sign of ALU_result is positive
Branch on Overflow	bv L	if (overflowflag == 1) then goto L	Branch if overflow is 1
Branch on No Overflow	bnv L	if (overflowflag == 0) then goto L	Branch if overflow is 1
Call	Call L	ra 🡐 (PC)+4 ; goto L	Store the next PC in ra and jump to L
Return	Ret	goto (ra)	Jump to the address in ra

Shift

Instruction	Example	Meaning	Comments
Shift left logical	shll rs, sh	rs 🡐 (rs) left-shifted by sh	Logically left shift value in (rs) by sh and store in rs
Shift right logical	shrl rs, sh	rs 🡐 (rs) right-shifted by sh	Logically right shift value in rs by sh and store in rs
Shift left logical variable	shllv rs, rt	rs 🡐 (rs) left-shifted by (rt)	Left shift (rs) by (rt) and store in rs
Shift right logical variable	shrlv rs, rt	rs 🡐 (rs) right-shifted by (rt)	Right shift (rs) by (rt) and store in rs

Shift right arithmetic	shra rs, sh	rs ← (rs) arithmetic right-shifted by sh	Right arithmetic shift (rs) by (rt) and store in rs
Shift right arithmetic variable	shrav rs, rt	rs ← (rs) right-shifted by (rt)	Right arithmetic shift value in rs by value in rt and store in rs

Registers

Register Number	Register Name	Description
0	**$zero**	The value 0
2-3	**$v0 - $v1**	(**v**alues) from expression evaluation and function results
4-7	**$a0 - $a3**	(**a**rguments) First four parameters for subroutine
8-15, 24-25	**$t0 - $t9**	**T**emporary variables
16-23	**$s0 - $s7**	**S**aved values representing final computed results
31	**$ra**	**R**eturn **a**ddress