Encoding Format for all instructions

R-type instructions

Add	add rs, rt	000000|-rs--|-rt--|xxxxxxxxxx|000001
Comp	comp rs, rt	000000|-rs--|-rt--|xxxxxxxxxx|000010
AND	and rs, rt	000000|-rs--|-rt--|xxxxxxxxxx|000011
XOR	xor rs, rt	000000|-rs--|-rt--|xxxxxxxxxx|000100
Shift left logical variable	shllv rs, rt	000000|-rs--|-rt--|xxxxxxxxxx|000101
Shift right logical variable	shrlv rs, rt	000000|-rs--|-rt--|xxxxxxxxxx|000110
Shift right arithmetic variable	shrav rs, rt	000000|-rs--|-rt--|xxxxxxxxxx|000111

Immediate Type instructions

Add immediate	addi rs, imm	100001|-rs--|------imm------------
Complement Immediate	compi rs, imm	100010|-rs--|------imm------------
Shift left logical	shll rs, imm	100101|-rs--|------imm------------
Shift right logical	shrl rs, imm	100110|-rs--|------imm------------
Shift right arithmetic	shra rs, imm	100111|-rs--|------imm------------

Load/store instructions

Load Word	lw rt, imm(rs)	010000|-rs--|-rt--|------imm-------
Store Word	sw rt, imm(rs)	010001|-rs--|-rt--|------imm-------

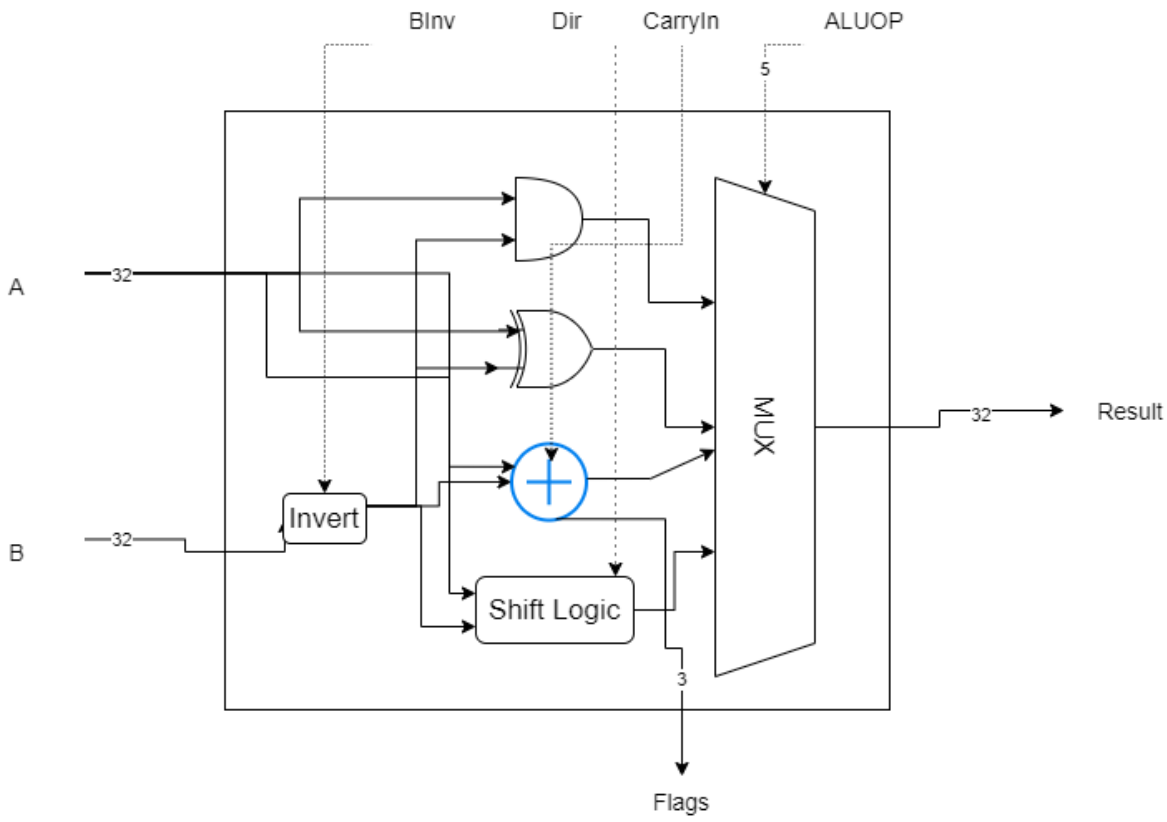
Branch instructions

Unconditional branch	b L	110000|------L-------------------
Branch and link	bl L	110001|------L-------------------
Branch on Carry	bcy L	110010|------L-------------------
Branch on No Carry	bncy L	110011|------L-------------------
Branch Register	br rs	110100|-rs--|xxxxxxxxxxxxxxxxxxxx
Branch on less than 0	bltz rs, L	110101|-rs--|------L--------------
Branch on flag zero	bz rs, L	110110|-rs--|------L--------------
Branch on flag not zero	bnz rs, L	110111|-rs--|------L--------------

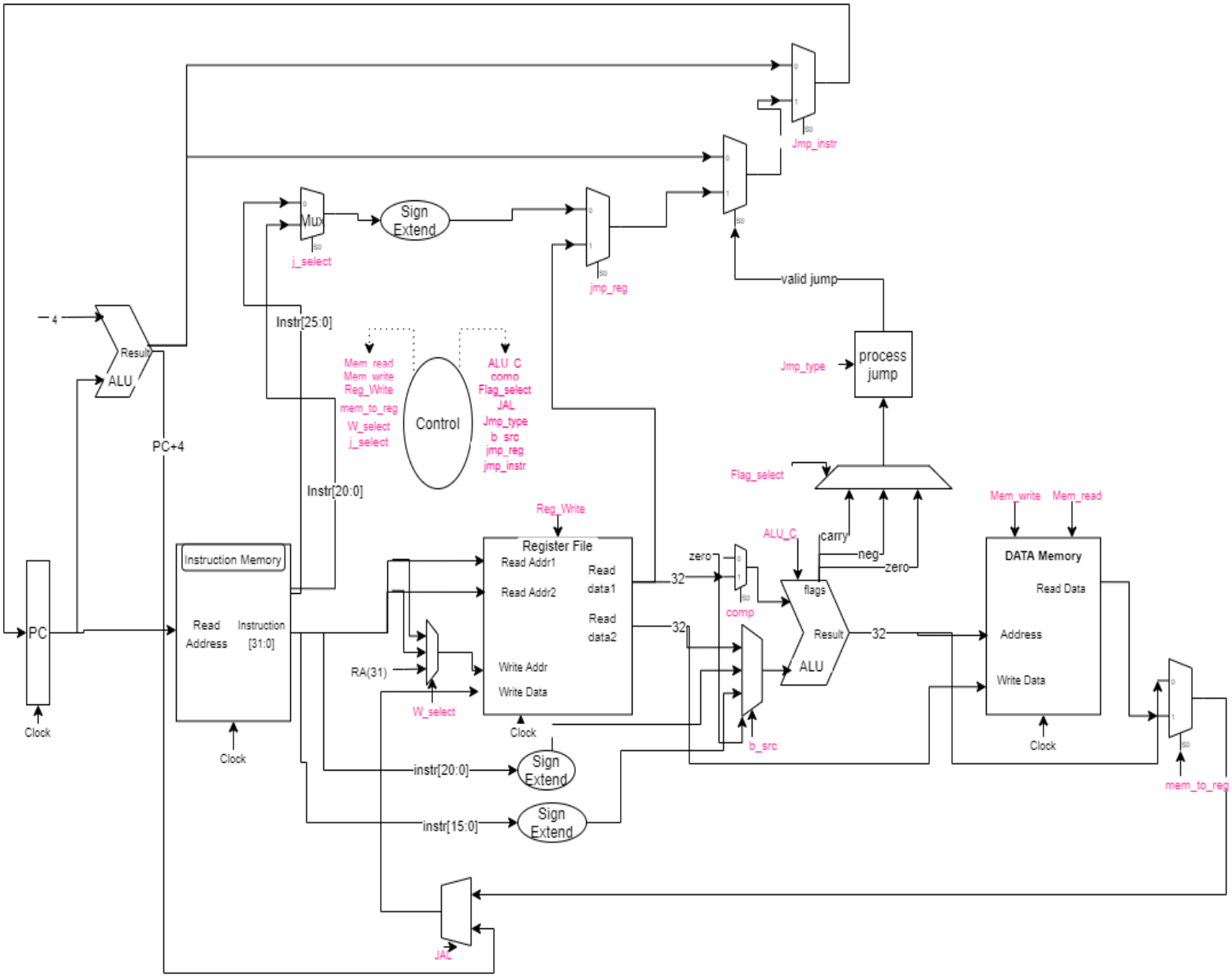
The datapath elements needed in the design are ALU, Control, memories like register file, instruction and data cache.

In the following diagrams the dotted lines denote control signals and solid lines denote data paths.

ALU Design



OverAll Design of RISC cpu



Description of Control Signals

Multibit Control signals

ALU has a 5 bit control signal which is represented as

ALU_C [5 bit]
xxxxx

x	x	x	xx
B_inv	Dir	Carry_in	ALUOP

ALUOP

00 - ADD
01 - AND
10 - XOR
11 - Shift

W_select is a 2 bit control signal which decides the write register

00 - Read_addr1
01 - Read_addr2
10 - $ra

b_src is a 2 bit control signal which decides the second operand of ALU

00 - Read_Data2
01 - Immediate value
10 - Immediate value for memory instructions
11 - zero

flag_select is a 2 bit control signal which decides the flag for branch instructions

00 - zero
01 - neg
10 - carry

Control Signals Truth Table

add, comp, and, xor, shllv, shrlv, shrav in order

Opcode	Function code	Mem_Read	Mem_Write	Reg_Write	Mem_to_reg	W_select	comp	flag_select	JAL	jmp_type	b_src	jmp_reg	jmp_instr	ALU_C
000000	000001	0	0	1	0	00	0	XX	0	XXX	00	X	0	0X000
000000	000010	0	0	1	0	00	1	XX	0	XXX	00	X	0	1X100
000000	000011	0	0	1	0	00	0	XX	0	XXX	00	X	0	0X001
000000	000100	0	0	1	0	00	0	XX	0	XXX	00	X	0	0X010
000000	000101	0	0	1	0	00	0	XX	0	XXX	00	X	0	00X11
000000	000110	0	0	1	0	00	0	XX	0	XXX	00	X	0	01X11
000000	000111	0	0	1	0	00	0	XX	0	XXX	00	X	0	01111

addi, compi, shll, shrl, shra

Opcode	Mem_Read	Mem_Write	Reg_Write	Mem_to_reg	W_select	comp	flag_select	JAL	jmp_type	b_src	jmp_reg	jmp_instr	ALU_C
100001	0	0	1	0	00	0	XX	0	XXX	01	X	0	0X000
100010	0	0	1	0	00	1	XX	0	XXX	01	X	0	1X100
100101	0	0	1	0	00	0	XX	0	XXX	01	X	0	00X11
100110	0	0	1	0	00	0	XX	0	XXX	01	X	0	01X11
100111	0	0	1	0	00	0	XX	0	XXX	01	X	0	01111

lw, sw in order

Opcode	Mem_Read	Mem_Write	Reg_Write	Mem_to_reg	W_select	comp	flag_select	JAL	jmp_type	b_src	jmp_reg	jmp_instr	ALU_C
010000	1	0	1	1	01	0	XX	0	XXX	10	X	0	0X000
010001	0	1	0	0	XX	0	XX	0	XXX	10	X	0	0X000

b, bl, bcy, bncy, br, bltz, bz, bnz in order

Opcode	Mem_Read	Mem_Write	Reg_Write	Mem_to_reg	W_select	comp	flag_select	JAL	jmp_type	b_src	jmp_reg	jmp_instr	j_select	ALU_C
110000	0	0	0	0	00	0	XX	0	000	00	0	1	0	XXXXX
110001	0	0	1	0	10	0	XX	1	001	00	0	1	0	XXXXX
110010	0	0	0	0	00	0	10	0	010	00	0	1	0	XXXXX
110011	0	0	0	0	00	0	10	0	011	00	0	1	0	XXXXX
110100	0	0	0	0	00	0	XX	0	100	00	1	1	0	XXXXX
110101	0	0	0	0	00	0	01	0	101	11	0	1	1	00000
110110	0	0	0	0	00	0	00	0	110	11	0	1	1	00000
110111	0	0	0	0	00	0	00	0	111	11	0	1	1	00000

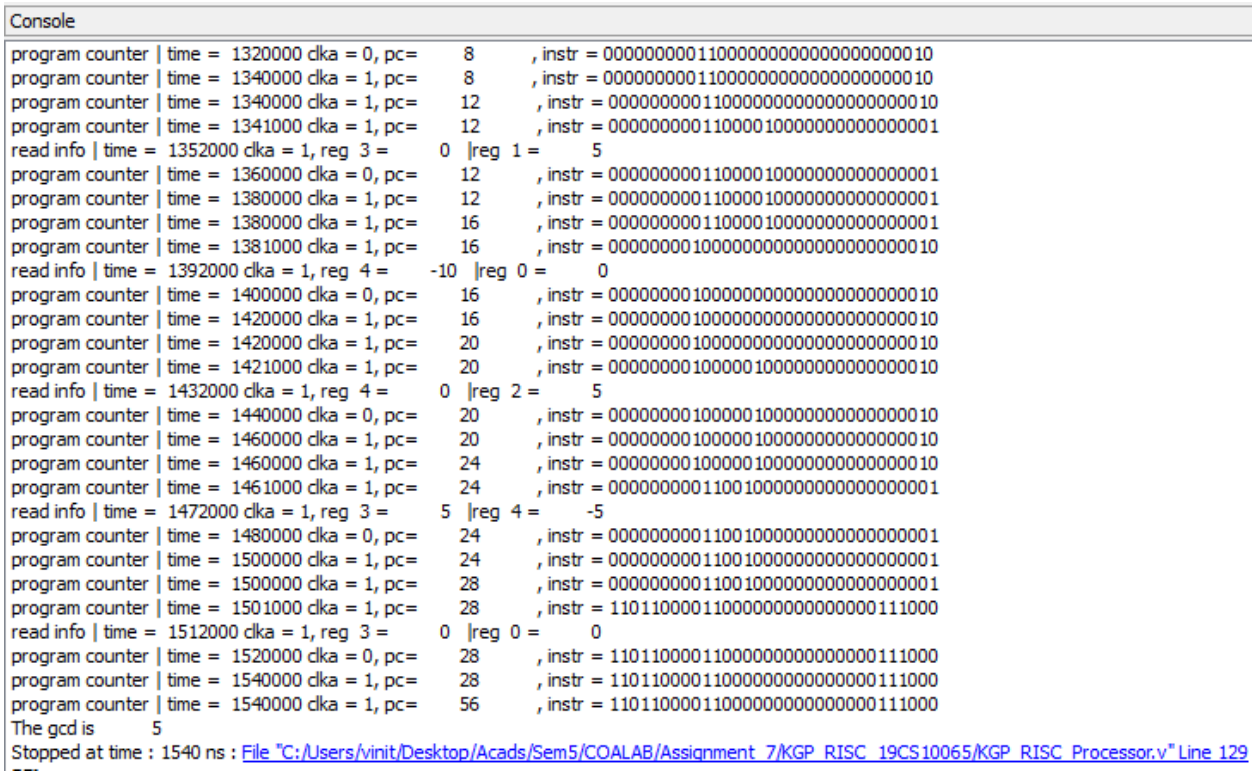
Finally we run a program to find the gcd of two numbers
The bars in between are given for readability

0 010000|00000|00001|0000000000000000 lw $1, 0($0)
4 010000|00000|00010|0000000000000100 lw $2, 4($0)
8 000000|00011|00000|0000000000|000010 comp $3, $0
12 000000|00011|00001|0000000000|000001 add $3, $1
16 000000|00100|00000|0000000000|000010 comp $4, $0
20 000000|00100|00010|0000000000|000010 comp $4, $2
24 000000|00011|00100|0000000000|000001 add $3, $4
28 110110|00011|00000000000000111000 bz $3, 56
32 110101|00011|00000000000000110000 bltz $3, 48
36 000000|00001|00000|0000000000|000010 comp $1, $0
40 000000|00001|00011|0000000000|000001 add $1, $3
44 110000|00000000000000000000001000 b 8

48 000000|00010|00011|0000000000|000010 comp $2, $3
52 110000|00000000000000000000001000 b 8
56 00000000000000000000000000000000 end

The data memory has two integers 10 and 25 which are loaded and then the program calculates the gcd of the two numbers.

As it can be seen the program finds the gcd to be 5 in the final line.



```
Console
program counter | time =  1320000 clka = 0, pc=       8        , instr = 00000000011000000000000000000010
program counter | time =  1340000 clka = 1, pc=       8        , instr = 00000000011000000000000000000010
program counter | time =  1340000 clka = 1, pc=      12        , instr = 00000000011000000000000000000010
program counter | time =  1341000 clka = 1, pc=      12        , instr = 00000000011000010000000000000001
read info | time =  1352000 clka = 1, reg  3 =       0  |reg  1 =       5
program counter | time =  1360000 clka = 0, pc=      12        , instr = 00000000011000010000000000000001
program counter | time =  1380000 clka = 1, pc=      12        , instr = 00000000011000010000000000000001
program counter | time =  1380000 clka = 1, pc=      16        , instr = 00000000011000010000000000000001
program counter | time =  1381000 clka = 1, pc=      16        , instr = 00000000100000000000000000000010
read info | time =  1392000 clka = 1, reg  4 =     -10  |reg  0 =       0
program counter | time =  1400000 clka = 0, pc=      16        , instr = 00000000100000000000000000000010
program counter | time =  1420000 clka = 1, pc=      16        , instr = 00000000100000000000000000000010
program counter | time =  1420000 clka = 1, pc=      20        , instr = 00000000100000000000000000000010
program counter | time =  1421000 clka = 1, pc=      20        , instr = 00000000100000100000000000000010
read info | time =  1432000 clka = 1, reg  4 =       0  |reg  2 =       5
program counter | time =  1440000 clka = 0, pc=      20        , instr = 00000000100000100000000000000010
program counter | time =  1460000 clka = 1, pc=      20        , instr = 00000000100000100000000000000010
program counter | time =  1460000 clka = 1, pc=      24        , instr = 00000000100000100000000000000010
program counter | time =  1461000 clka = 1, pc=      24        , instr = 00000000011001000000000000000001
read info | time =  1472000 clka = 1, reg  3 =       5  |reg  4 =      -5
program counter | time =  1480000 clka = 0, pc=      24        , instr = 00000000011001000000000000000001
program counter | time =  1500000 clka = 1, pc=      24        , instr = 00000000011001000000000000000001
program counter | time =  1500000 clka = 1, pc=      28        , instr = 00000000011001000000000000000001
program counter | time =  1501000 clka = 1, pc=      28        , instr = 11011000011000000000000000111000
read info | time =  1512000 clka = 1, reg  3 =       0  |reg  0 =       0
program counter | time =  1520000 clka = 0, pc=      28        , instr = 11011000011000000000000000111000
program counter | time =  1540000 clka = 1, pc=      28        , instr = 11011000011000000000000000111000
program counter | time =  1540000 clka = 1, pc=      56        , instr = 11011000011000000000000000111000
The gcd is         5
Stopped at time : 1540 ns : File "C:/Users/vinit/Desktop/Acads/Sem5/COALAB/Assignment_7/KGP_RISC_19CS10065/KGP_RISC_Processor.v" Line 129
```