KGP RISC

5/11/2021

—

Group 2(19CS10071 , 19CS10060)

Contents :

Instruction Format

R_type:

Opcode(6)	rs(5)	rt(5)	shamt(5)	func(6)	dont_care(5)

Instruction	**Command**	**Opcode(6)**	**func(6)**
Add	add rs,rt	000000	000000
Complement	comp rs,rt	000001	000001
AND	and rs,rt	000000	000010
XOR	xor rs,rt	000000	000011
Shift left logical	shll rs, sh	000100	000100
Shift left logical variable	shllv rs, rt	000000	000100
Shift right logical	shrl rs, sh	000100	000101
Shift right logical variable	shrlv rs, rt	000000	000101
Shift right arithmetic	shra rs, sh	000100	000110
Shift right arithmetic variable	shrav rs, rt	000000	000110

J_type:

op(6)	**target_address(26)**

Instruction	**command**	**op(6)**
Unconditional branch	b L	000101

Branch and link	bl L	000110
Branch on Carry	bcy L	000111
Branch on No Carry	bncy L	001000

I Type :

op(6)	**rs(5)**	**rt(5)**	**Imm(16)**

Load/Store :

instruction	**Command**	**op(6)**
Load word	lw rt,imm(rs)	001001
Store word	rw rt,imm(rs)	001010

Add Immediate

Add Immediate	addi rs,imm	000010
Comp Immediate	compi rs,imm	000011

Conditional branches :

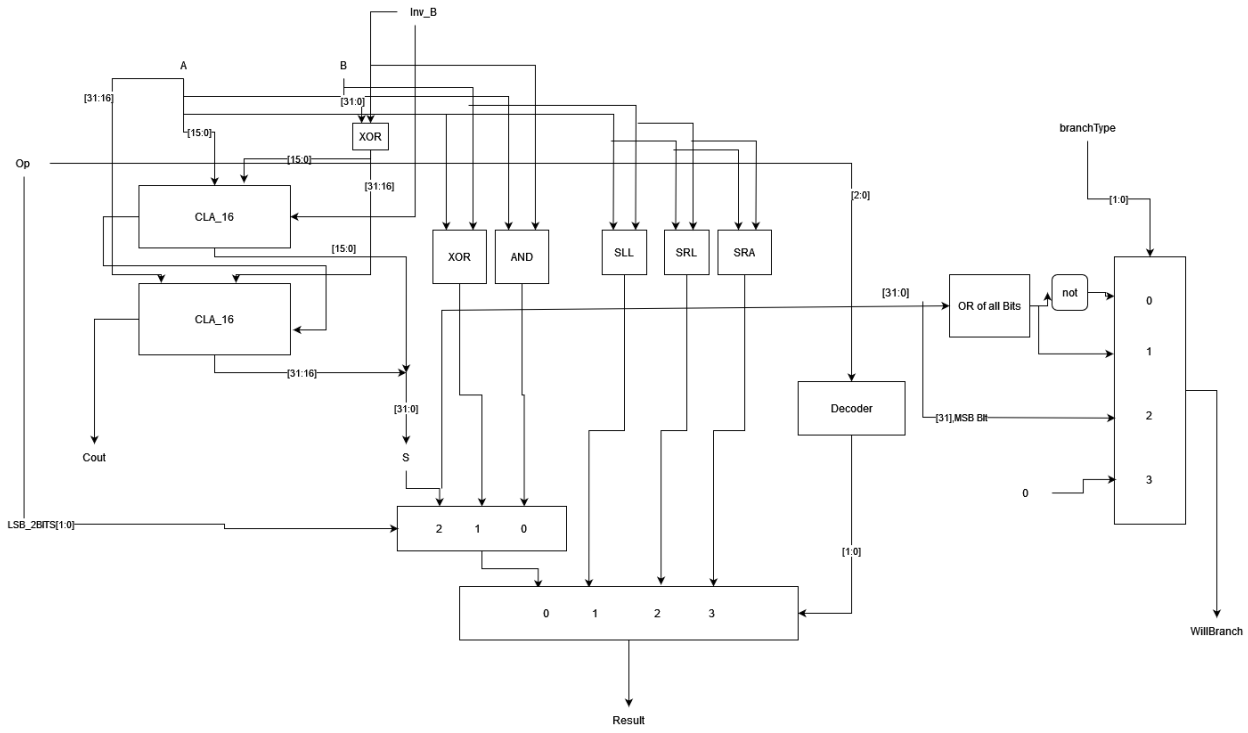
op(6)	**rs(5)**	**dont_care(5)**	**Imm(16)**

Instruction	**Command**	**op(6)**
Branch register	br rs	001011
Branch on less than 0	bltz rs,L	001100
Branch on flag 0	bz rs,L	001101
Branch on flag not 0	bnz rs,L	001110

As per the design, (64-14) = 50 more instructions can be added.

- Opcode is 6 bit so total of 64 instruction
- Out of which 14 are used

ALU DESIGN



ALU_control : <Op><Inv_B><Cin><Branch_Type>

Truth Table for ALU

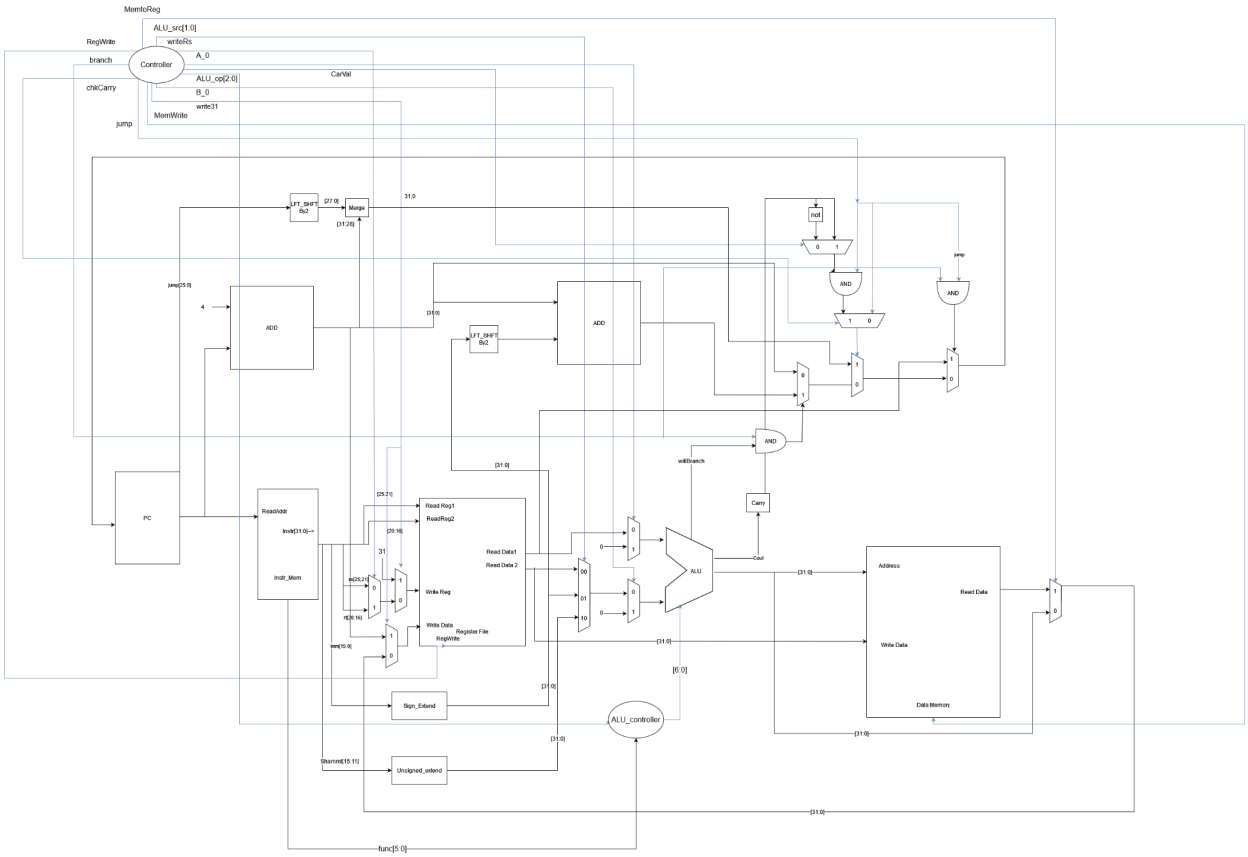
Instruction	ALU_OP	Instruction_operation	Func	ALU_action	ALU_control unit
lw rt,imm(rs)	**001**	load	X	add with a = rs,b = imm	0100011
sw rt,imm(rs)	001	save	X	add with a = rs,b = imm	0100011
bltz rs,L	010	branch on	X	add with b =	0100010

		less than 0		0 ,a = rs	
bz rs,L	011	branch on rs = 0	X	add with b=0	0100000
bnz rs,L	100	branch on not 0	X	add with b = 0	0100001
add rs,rt	000	rs ← (rs) + (rt)	000000	add with a = rs,b = rt	0100011
comp rs,rt	000	rs<=2's rt	000001	(comp)add with a=0,b = rt	0101111
addi rs,imm	101	rs ← (rs) + imm	X	add with a = rs,b = imm	0100011
compi rs,imm	110	rs ← 2's Complement (imm)	X	add with a = 0,b = imm	0101111
and rs,rt	000	rs ← (rs) ∧ (rt)	000010	bitwise and with a = rs,b = rt	0000011
xor rs,rt	000	rs ← (rs) ⊕ (rt)	000011	bitwise xor with a = rs,b = rt	0010011
shll rs, sh	000	rs ← (rs) left-shifted by sh	000100		0110011
shrl rs, sh	000	rs ← (rs) right-shifted by sh	000101		1000011

shllv rs, rt	000	rs ← (rs) left-shifted by (rt)	000100		0110011
shrlv rs, rt	000	rs ← (rs) right-shifted by (rt)	000101		1000011
shra rs, sh	000	rs ← (rs) arithmetic right-shifted by sh	000110		1010011
shrav rs, rt	000	rs ← (rs) right-shifted by (rt)	000110		1010011

CPU DESIGN

Blue Indicates control signal,else data signal



TRUTH TABLE FOR CONTROLLER:

Command	AluSrc (2bit)	ALU_op (3bit)	MemWrite	MemtoReg	Branch	Jump	RegWrite	write_rs	A_0	B_0	write31	chkCarry	carryVal
add rs,rt	00	000	0	0	0	0	1	0	0	0	0	x	x
comp rs,rt	00	000	0	0	0	0	1	0	1	0	0	x	x
addi rs,imm	01	101	0	0	0	0	1	0	0	0	0	x	x
compi rs,imm	01	110	0	0	0	0	1	0	1	0	0	x	x
and rs,rt	00	000	0	0	0	0	1	0	0	0	0	x	x
xor rs,rt	00	000	0	0	0	0	1	0	0	0	0	x	x
shll rs, sh	10	000	0	0	0	0	1	0	0	0	0	x	x
shrl rs, sh	10	000	0	0	0	0	1	0	0	0	0	x	x
shllv rs, rt	00	000	0	0	0	0	1	0	0	0	0	x	x

shrlv rs, rt	00	000	0	0	0	0	1	0	0	0	0	x	x
shra rs, sh	10	000	0	0	0	0	1	0	0	0	0	x	x
shrav rs, rt	00	000	0	0	0	0	1	0	0	0	0	x	x
lw rt,imm(rs)	01	001	0	1	0	0	1	1	0	0	0	x	x
sw rt,imm(rs)	01	001	1	0	0	0	0	x	0	0	0	x	x
b L	x	x	0	0	0	1	0	x	0	0	0	0	x
br rs	x	x	0	0	1	1	0	x	0	0	0	x	x
bltz rs,L	X	010	0	0	1	0	0	x	0	1	0	x	x
bz rs,L	X	011	0	0	1	0	0	x	0	1	0	x	x
bnz rs,L	X	100	0	0	1	0	0	x	0	1	0	x	x
bl L	X	x	0	0	0	1	1	x	0	0	1	x	x
bcy L	X	x	0	0	0	1	0	x	0	0	0	1	1
bncy L	X	x	0	0	0	1	0	x	0	0	0	1	0

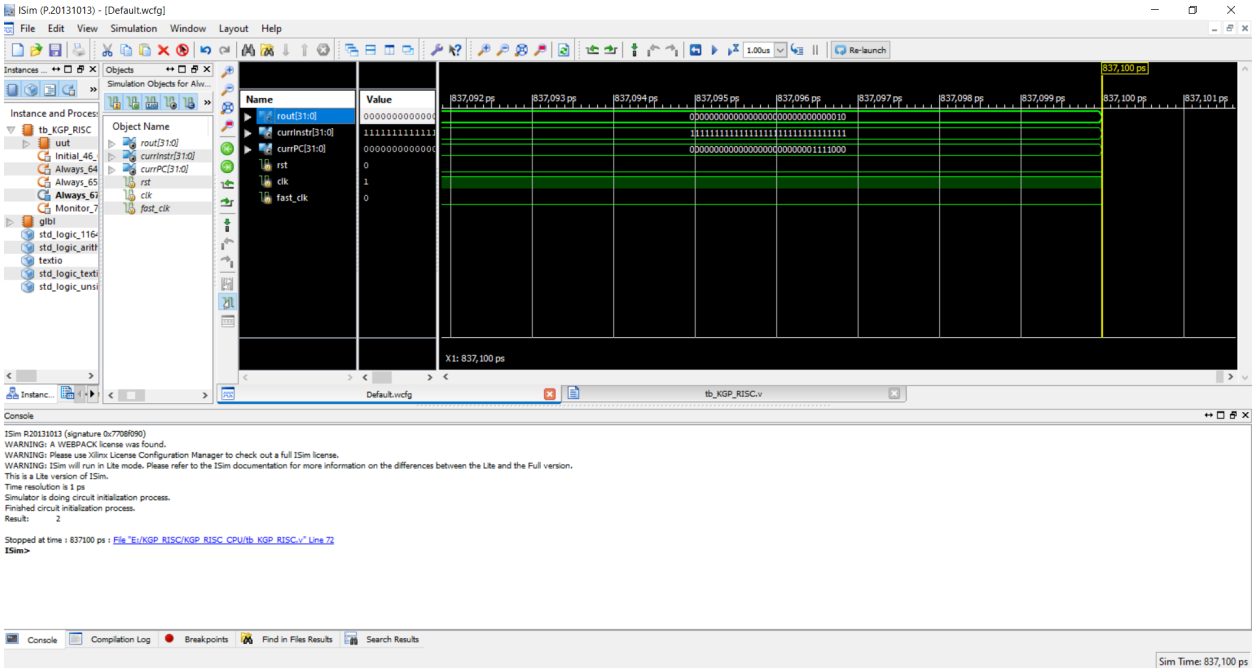
README NOTES:

- The final result is printed in the console log as “Result : <result> “
- The final result is stored in r30 named as rout at top level.
- To load the coe files you have to regenerate the InstrMem with the given coe file.
- main test bench : tb_KGP_RISC.v
- main top module : KGP_RISC.v
- Specs of Instr Mem(These are also mentioned in the readme file)
    - Single Port ROM
    - Enable 32 bit Address
    - Read width as 32 and read depth as 256(just keeping big max prog lines is 30-40,still keeping big to extend to bigger programs if so)
    - Use ENA Pin enabled
    - Load the coe file
- Specs of DataMemory;
    - Single Port Ram
    - Enable 32 bit Address
    - Write Width : 32 , Write Depth : 256

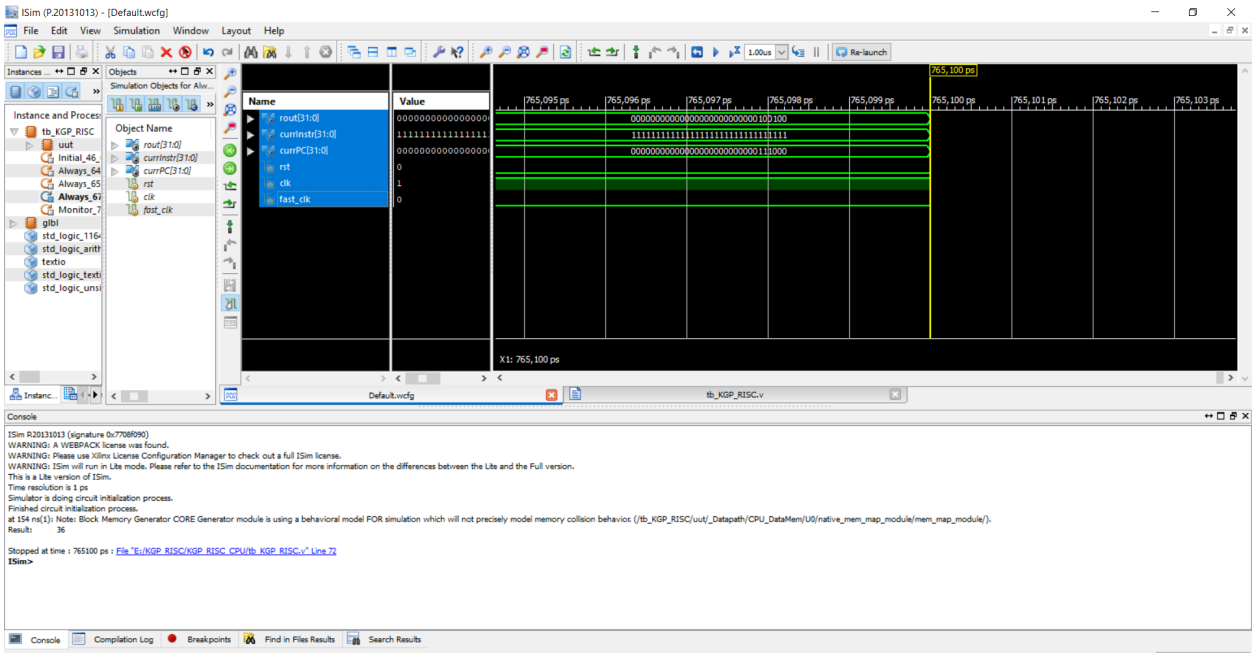
- Write First , ALways enabled
- Load the data coe file

- Details of coe files;
  - gcd.coe : for computing gcd of 14 and 8(expected output : 2)
  - gcddata.coe : for gcd data input (set as 14 and 8)
  - nsum.coe : for computing the sum of first 8 natural nos(expected output : 36)
  - nsumdata.coe : for nsum data input(set as 8)

Simulation Screenshots:



For gcd gcd(8,14) = 2



For nsum ,input = 8 , Result = 36 (1+2+..8 = 8*9/2 = 36)