**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()** | **Register-2()** | **Shift-amount ()** | **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** |
|  |  |  |  |  |  | 000000 |
|  |  |  |  |  |  | 000001 |
|  |  |  |  |  |  | 000010 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | 000100 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

NOTE: Arithmetic right shift

## **Immediate – type:**

Instruction Format

|  |  |  |  |
| --- | --- | --- | --- |
| **OP-code** | **Register-1()** | **Register-2()** | **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]** |
|  |  |  |  |  |  |
|  |  |  |  | N/A |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

NOTE: MEM memory

## **Branch – type:**

Instruction Format

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

|  |  |
| --- | --- |
| **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. |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  | 010100 |  |  |
| ii.(jump) |  |  | **Address[25:0]** | |
|  |  |  |  | |
|  |  | 010101 |  | |
|  |  | 010110 |  | |
|  |  | 010111 |  | |

NOTE: PC program counter

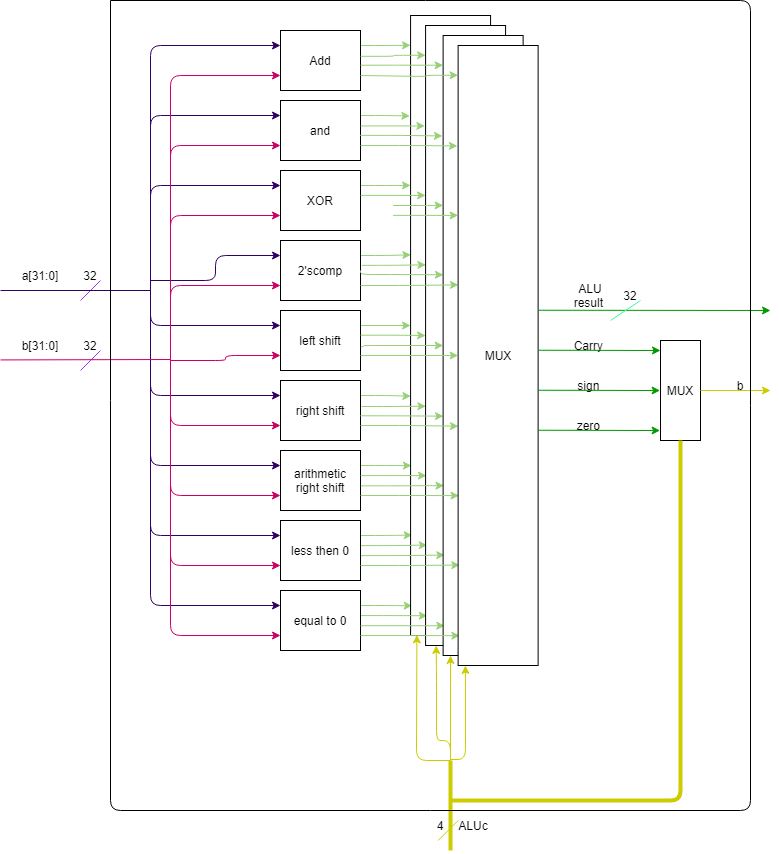
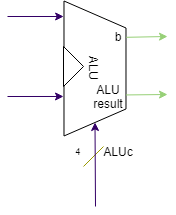
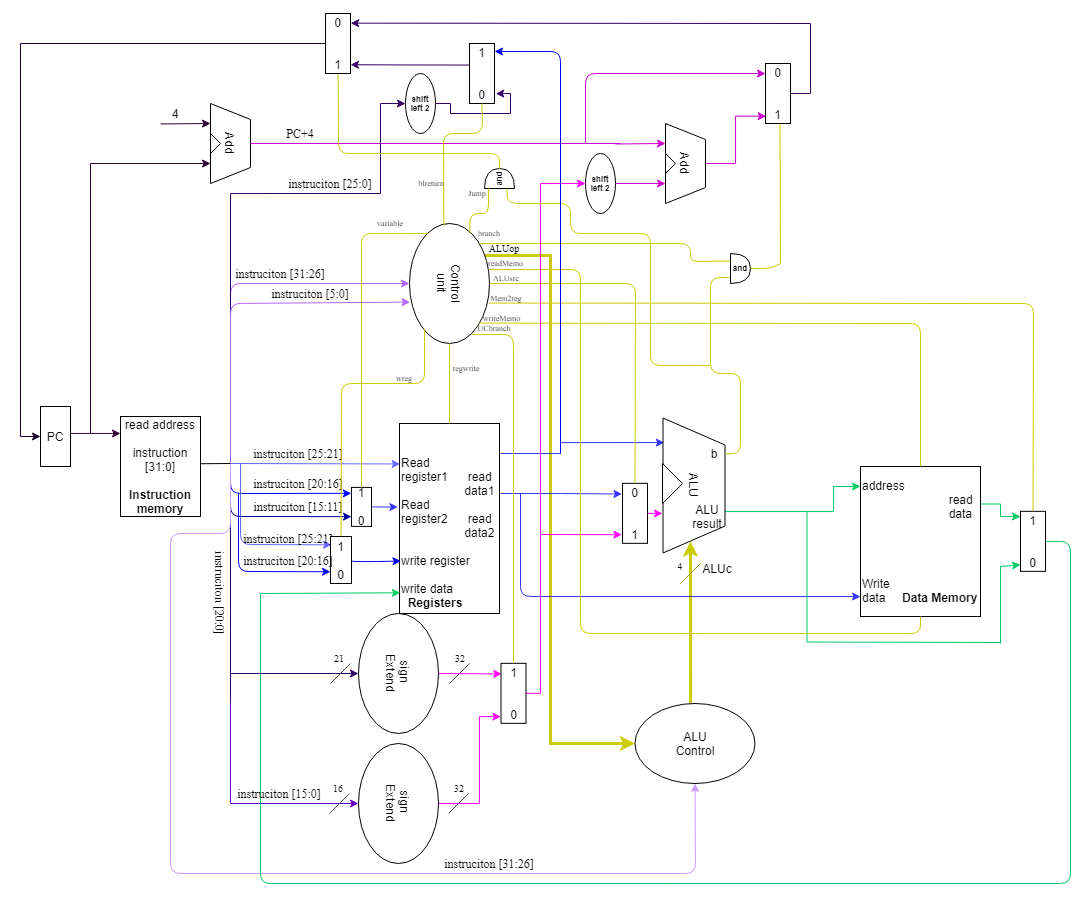
* Architecture for ALU block:
* Instruction set Architecture:

fig: CPU design architecture

F

Control Signals:

1. : signal is to {read value }/{shift amount (for )} to register.
2. **:** enables if any result is to be stored in register
3. **:** enables if any value is to be written into the memory
4. **:** enables if instruction is encountered
5. **:** enables if instruction type is immediate-type ()
6. **:** enables if any data is to be read from memory
7. **:** 4-bit control signals for ALU control
8. **:** for branch instructions which have address of size 21 bits
9. **:** for branch instruction’s which has address of size 21 turn it will be enabled
10. **:** for branch instruction’s which has address of size 16 turn it will be enabled
11. ***:*** this control signal is from ALU which ensures branch condition is satisfied or not.
12. **:** it will enable if load word instruction is given so that value will be loaded into the rs register.
13. **:** it enables when there is a instruction is given.

## Control signals and data path

1. **R-type instructions**
   1. OP-codes 🡪 000000

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

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.
  1. Op-codes 🡪 000001

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Data path:

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

1. **Immediate-type instructions**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

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.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 0 |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |

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() form memory of address ALUresult 🡪 write value form memory in desired register.

1. 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.

1. **Branch instructions:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 010001 |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 010010 |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 010011 |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 010100 |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
|  | (signal from ALU) |
|  |  |
|  |  |
|  |  |
|  |  |

Data path:

* Read instructions🡪 control signals🡪 read register1 🡪 immediate sign extension🡪 input data1 in ALU 🡪 signal b 🡪 PC update.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 010000 |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 010101 |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 010110 |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 010111 |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
|  | (signal from ALU) |
|  |  |
|  |  |
|  |  |
|  |  |

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.
* Registers Address:

# registers

regs = {

    "$z0":"00000",

    "$at" : "00001",

    "$v0" : "00010",

    "$v1" : "00011",

    "$a0" : "00100",

    "$a1" : "00101",

    "$a2" : "00110",

    "$a3" : "00111",

    "$t0" : "01000",

    "$t1" : "01001",

    "$t2" : "01010",

    "$t3" : "01011",

    "$t4" : "01100",

    "$t5" : "01101",

    "$t6" : "01110",

    "$t7" : "01111",

    "$s0" : "10000",

    "$s1" : "10001",

    "$s2" : "10010",

    "$s3" : "10011",

    "$s4" : "10100",

    "$s5" : "10101",

    "$s6" : "10110",

    "$s7" : "10111",

    "$t8" : "11000",

    "$t9" : "11001",

    "$k0" : "11010",

    "$k1" : "11011",

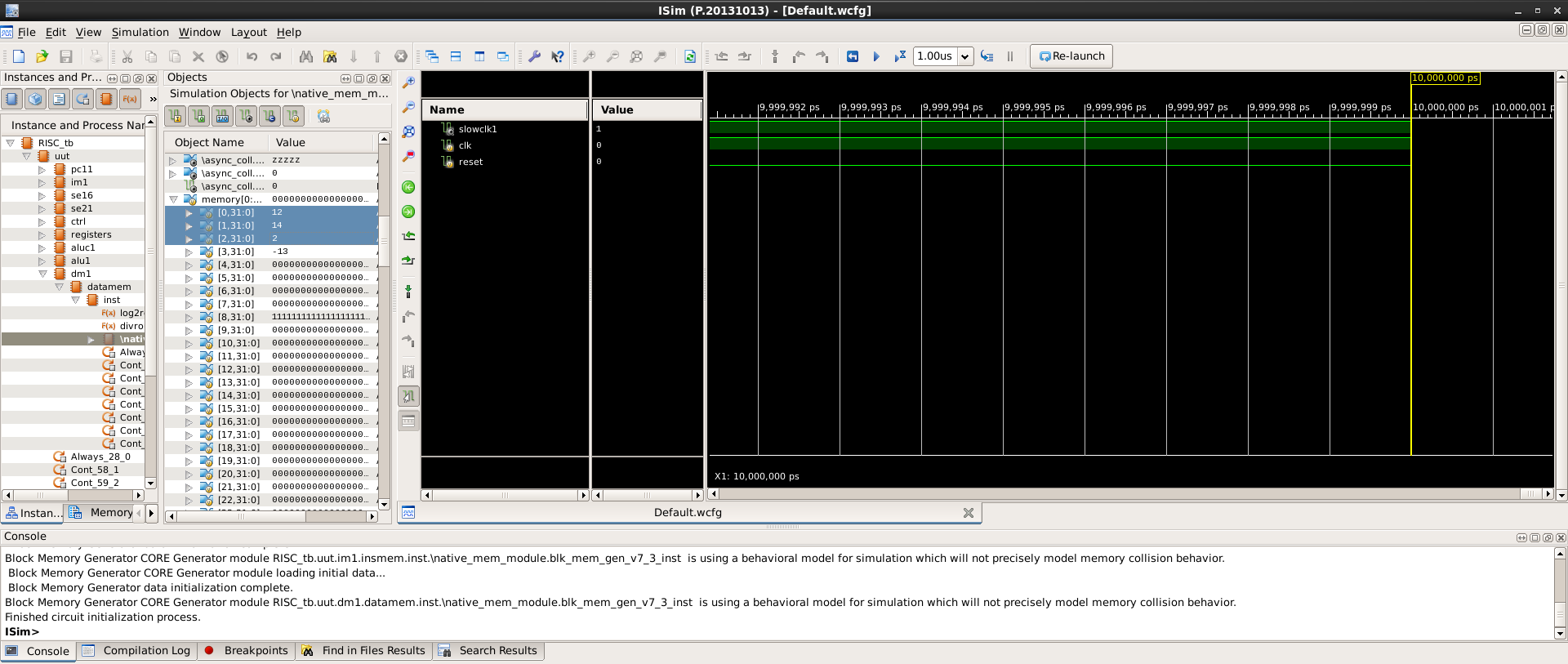
    "$gp" : "11100",

    "$fp" : "11101",

    "$sp" : "11110",

    "$ra"  :"11111"}

* Output for Given instructions:

1. GCD instructions were given
   1. 1st 2 inputs from data are taken and calculated gcd is stored in 3rd memory block.
   2. Eg: GCD of 14,12 is 2 which is stored in mem[2] whose initial value is 12.

**Assumptions and Other Points to Be Noted**:

1. Block RAMs have a peculiar issue that they have significant delay in fetching data (around #1 clock time) from the RAM. Hence, we have divided the input clock into two parts, a slower one and a faster one. The faster one is eight times faster and is used to fetch data from Block RAM whereas the slower one is used for other modules.

2. Block RAMs have addresses as 0, 1, 2, …. Hence, we have used PC+1 instead of PC+4.

**Note** : 1. Register file : Don’t use high level code, Use 32 \* 32 one bit input mux’s . Don’t compare entire 32 bit register addresses.