

Computer Organization and Architecture Lab

CPU simulation in Tk Gate for the given problem

(Sum of N natural Number)

This ISA design features a 16-bit instruction format with an 8-bit immediate value field. It supports 4 registers with 2-bit register addressing. There are inter-stage buffers (non-accessible) which hold results after every stage. There are two addressing mode bits which enable either register mode or immediate mode or memory address mode or user input mode. The 4-bit opcode supports a range of operations, including arithmetic, memory load/store, branching, and comparison. This ISA is designed for simple control flow and basic data manipulation, suitable for operations like arithmetic, move without complex branching structures.

Possible Commands

- i. ADD
- ii. SUB
- iii. MUL
- iv. DIV
- v. MOV
- vi. LOAD (Not Used)
- vii. CMPZ

- viii. JNE
- ix. STORE (Not Used)
- x. HLT

Modules Used

- Program Counter (PC)
- Instruction Fetch
- Instruction Decode
- Register File
- Arithmetic Logic Unit (ALU)
- Control Unit (CU)
- 16 Bit Comparator
- Jump Module
- Inter Stage Buffers (Registers)

DESCRIPTION OF CPU

▪ No of Registers:	4 (R0,R1,R2,R3)
▪ Architecture:	16 bit
▪ Data Path:	16 bit
▪ Instruction length:	16 bit
▪ PC Counter Size:	8 bit
▪ Range of Data:	0 to 65535
▪ Max Number of Instructions Possible:	128

BUFFER REGISTERS (ISB, Inter Stage Buffers)

- Stage 1-2 : 1 Registers
- Stage 2-3 : 7 Registers
- Stage 3-4 : 3 Registers
- Stage 4-1 : 1 Registers

Stage Descriptions

- Stage 1: **Fetch + Write Back**

- Stage 2: **Decode**
- Stage 3: **(Decode 2) Value Fetch**
- Stage 4: **Execute**

Clock Frequency: 200Hz

DESIGN STRUCTURE

Register File Description

- Number of Registers = 4

Register:	Encoded:
R0	00
R1	01
R2	10
R3	11

- Immediate Values:- **From 0 to 255(8 Bits)**

where 0 = 0000 0000

and 255 = 1111 1111

Operation Description

INSTRUCTION	OP CODE	OPERATION
Add	0000	[R1] <- R1 + R2 Adds R1 and R2
MUL	0001	[R1] <- R1 * R2 Multiplies R1 with R2, stores in R1
DIV	0010	[R1] <- R1 / R2 Divides R1 with R2, stores in R1

SUB	0011	$[R1] \leftarrow R1 - R2$ Subtracts R1 with R2, stores in R1
MOV	0100	$[R1] \leftarrow R2$ Moves R1 to R2, stores in R1
OUT	0101	$[R1]$ Displays value of R1
CMPZ	0110	$[R1] == 0$ Compares R1 to 0, sets flag register
JMP	0111	$[PC] \leftarrow [PC] + \text{offset}$ Updates PC to a new Address
STORE	1000	$[R2] \leftarrow R1$ Stores the value at R1 in memory (Unused)
LOAD	1001	$[R1] \leftarrow [R2]$ Loads the value at R1 in memory (Unused)

Instruction Description

OP CODE 4 Bits	OPERAND 1 2 Bits	ADDRESSING MODE 2 Bits	IMMEDIATE VALUE 8 Bits
WWWW	XX	YY	ZZZZZZZZ

- **www:- (4 bits)** This represents the OP Code
- **XX:- (2bits)** This represents the first operand.
(Addr of Register-1)
- **YY:- (2bits)** This represents the addressing mode for the second operand
(00 - Register, 01 - Immediate, 10 - Memory Addr, 11 - User Input)
- **ZZZZZZZZ:- (8bits)** In this segment if the mode of addressing is register then the first 2 bits are assigned to the register-2 address and the rest 6 Bits are ignored, in case of addressing mode or immediate mode, whole 8bits they are used to represent that immediate value, in case of user input mode, all 8bits are ignored.

Code: Sum of N Natural Numbers

1.1 Using Branching

Address	Address	Instruction	Binary Instructions	Hex
Decimal:	Hex Value:			

0	0	MOV R0, N	0100000100000000	4100
4	4	MOV R1, # 0	0100010100000000	4500
8	8	MOV R2, #1	0100100100000001	4901
12	C	L1: ADD R1, R0	0000010000000000	400
16	10	SUB R0, R2	0011000010000000	3080
20	14	CMPZ R0	0110000000000000	6000
24	18	JNE L1	0111000100001100	710C
28	1C	OUT R1	0101010000000000	5400
32	20	HLT	1111111111111111	FFFF

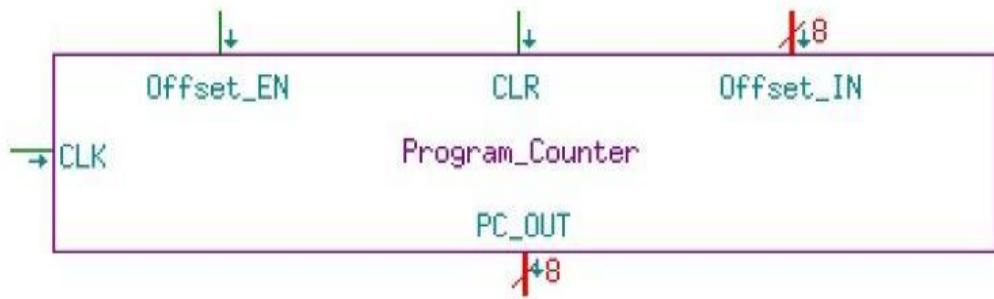
1.2 Without Using Branch:

Address (Decimal)	Address(Hex)	Instruction	Binary Instruction	Hex

0	0	MOV R0, N	0100001100000000	4300
4	4	MOV R2 ,2	0100100100000010	4902
8	8	MOV R1, R0	0100100100000001	4901
12	C	MUL R0, R0	0001000000000000	0400
16	10	ADD R0, R1	0000000010000000	3080
20	14	DIV R0, R2	0010000010000000	2080
24	18	OUT R0	0101000000000000	5800
28	1C	HLT	1111111111111111	FFFF

Module Description

1. Program Counter



DESCRIPTION

Input Ports :- 4

Output Ports :- 1

Offset EN :- When this is set high offset in value will be updated in pc

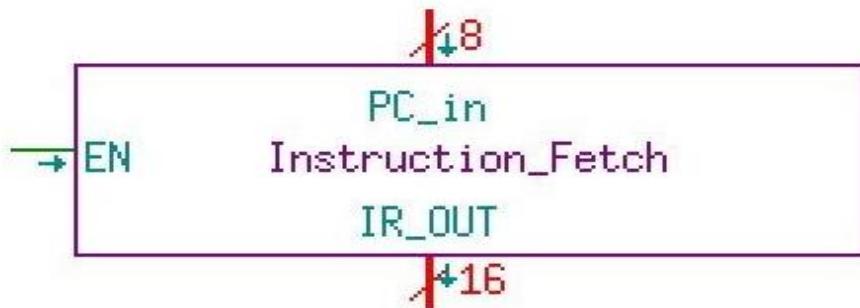
Offset IN (8 bit) :- This is the instruction index calculated by jump module

CLR :- It makes the value 0

CLK: - Connects the CPU clock

PC out:- It connects the PC to the main bus

2. Instruction Fetch



DESCRIPTION

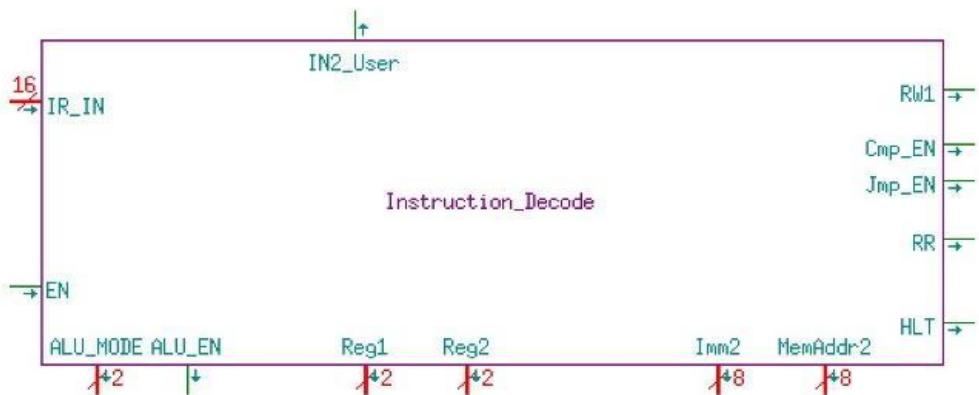
Input Posts :- 2

Output Ports :- 1

PC IN (8 bit) :- This takes the instruction index from the program counter

IR OUT(16 bit):- This is the binary instruction from the memory (Read Only Memory)

3. INSTRUCTION DECODER



DESCRIPTION

Input Ports :- 2

Output Ports:-12

IN2 user :- This signal sets to high when user has to give input.

IR IN (16 bit) :- This input takes the binary instruction.

ALU MODE (2 bit) :- This will decide the type of ALU operation.
 (00 for add, 01 for multiply, 10 for divide, 11 for subtraction)

ALU EN :- This is the output signal for the alu which turns on the alu

Imm2 (8 bit) :- This is the immediate value decoded form input instruction.

MemAddr2 (8 bit) :- This is the memory address decoded form the input instruction for load store operations (not used)

RW1 (1 bit) :-This decides the Read Write mode of the Register Address mode1

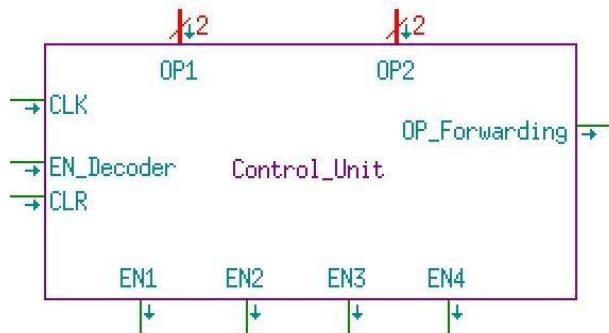
CMP_EN :- This turns on the comparator

JMP_EN :- This turns on jump operator

RR :- It identifies the register-register input.

HLT:- This is the halt signal which stops the CPU.

Reg 1, Reg2 (2 bit) :- This is the address of the registers decoded from the input instruction.



4. CONTROL UNIT

DESCRIPTION

Input Ports:- 5

Output Ports:- 5

OP1 (2 bit):- Not used, for pipelining.

OP2 (2 bit):- Not used, for pipelining.

OP Forwarding:- Not used, for pipelining.

EN Decoder (1 bit):- This is the input which turns on the decoder.

CLK:- It connects the CPU clock.

CLR:- It resets to value 0.

5. Comparator:



DESCRIPTION

Input Ports:- 3

Output Ports:- 1

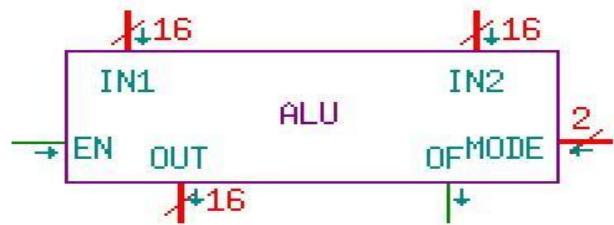
REG 1 (16 bit) :- This is data input 1 from register file.

REG 2(16 bit) :- This is data input 2 from register file.

EN :- This enables the comparator

OUT :- Comparison operation result.

6. ALU



DESCRIPTION

Input Ports (16bit):- 4

Output Ports (16 bit):- 2

OUT :- Arithmetic operation result.

OF :- The signal will be high when overflow happens

(In case of negative number if a valid negative number is on the out port then also OF will be high)

MODE (2bit) :- This will decide the type of operation.

7. JUMP MODULE



DESCRIPTION

Input Ports:- 2

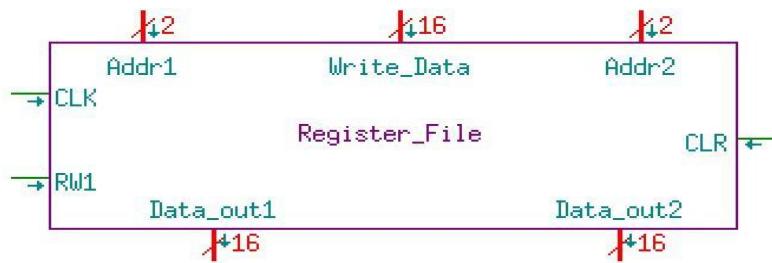
Output Ports:- 1

JNE Enable:- Enable is used to start or stop the jump module.

Status Flag:- Checks the signal from the status register

Offset Enable:- It connects the output to the program counter.

8. Register file



DESCRIPTION

Input ports :- 6

Output ports :- 2

Adder 1 (2 bit) :- The address of register 1

Adder 2 (2 bit) :- The address of register 2

Data out 1 (16 bit) :- Data of register 1

Data out 2 (16 bit) :- Data of register 2

CLK :- It connects the CPU clock

RW1 (1 bit) :- This decide the mode of the register Addr1 when set to 0 the register will be in Read mode and when to 1, Write mode

(Note: Register at Addr2 stays always in the Read mode)

CLR:- It resets the value to 0

THANK YOU FOR READING 😊