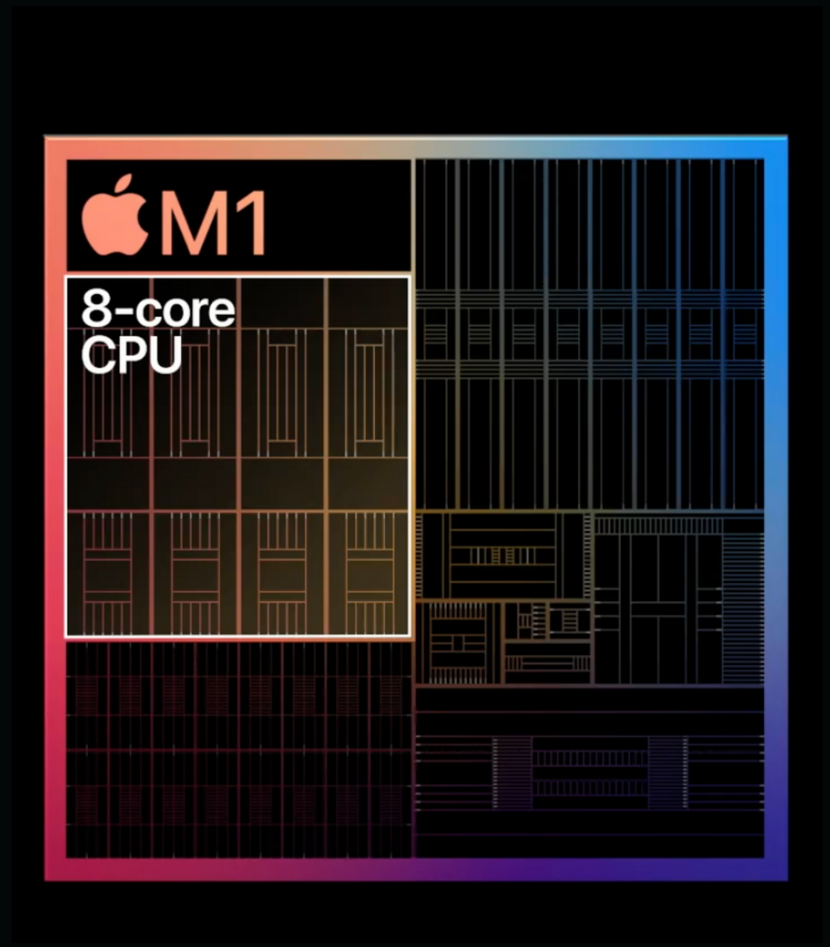
AMOGH ASV : PES1UG20EC002
ANIRUDHHAN R : PES1UG20EC029

RISC-V Architecture

Design of a RISC-V Processor in verilog

About RISC-V

- The Reduced Instruction Set Computer or RISC is a microprocessor design principle that favours a smaller and simpler set of instructions that all take the same amount of time to execute.
- Since each instruction type that a computer must perform requires additional transistors and circuitry, a larger list or set of computer instructions tends to make the microprocessor more complicated and slower in operation.
- The processor also incorporates a flag register which indicates carry, zero and parity status of the result.



RISC Vs CISC

RISC ARCHITECTURE

- Simpler instruction, hence simple instruction decoding.
- Instruction comes undersize of one word.
- Instruction takes a single clock cycle to get executed.
- More general-purpose registers.
- Simple Addressing modes.
- Fewer data types.
- A pipeline can be achieved.

CISC ARCHITECTURE

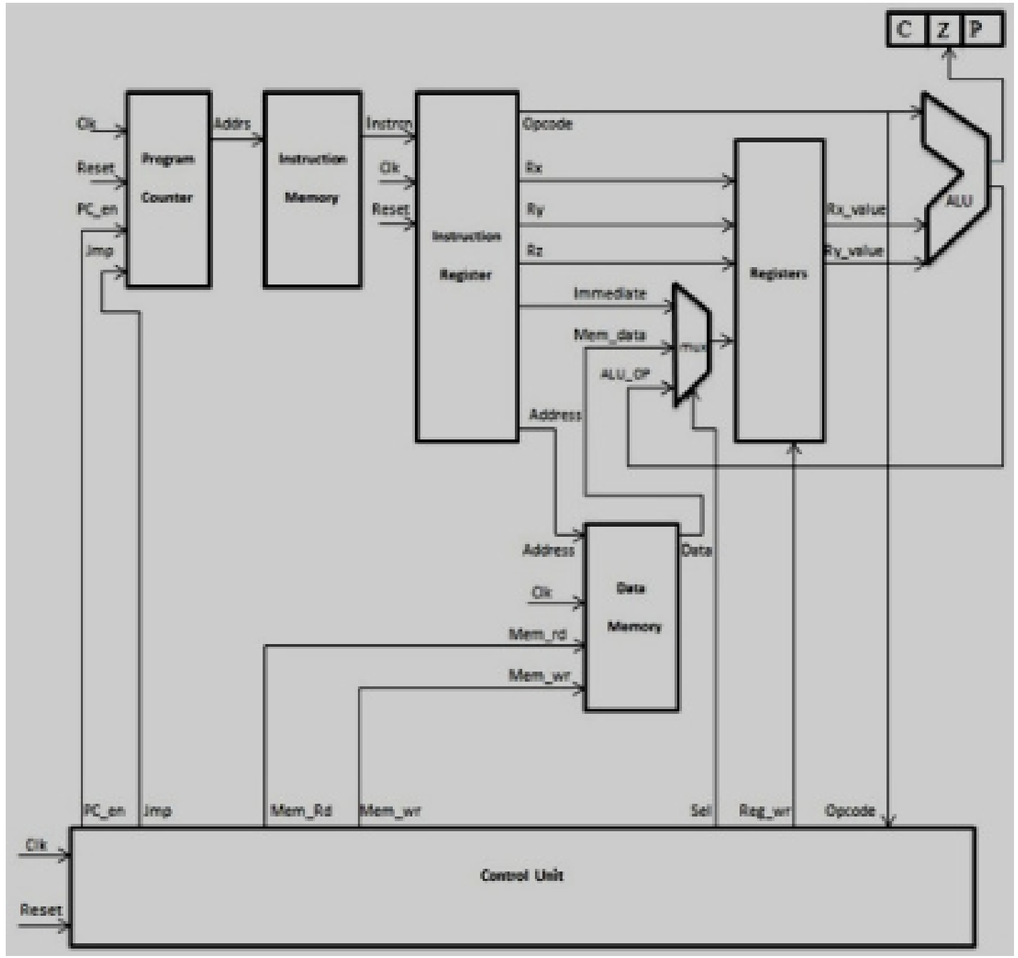
- Complex instruction, hence complex instruction decoding.
- Instructions are larger than one-word size.
- Instruction may take more than a single clock cycle to get executed.
- Less number of general-purpose registers as operations get performed in memory itself.
- Complex Addressing Modes.
- More Data types.

02

ARCHITECTURE OF THE PROJECT

16-BIT RISC PROCESSOR

- The objective of the project is to design a 16-bit RISC processor.
- The processor incorporates the following:
  - 16-bit ALU capable of performing 11 arithmetical and logical operations
  - 16-bit program counter
  - 24-bit Instruction register
  - Sixteen 16-bit general purpose registers
  - 3-bit flag register to indicate carry, zero and parity
- The processor has four states: idle, fetch, decode and execute
- The control unit provides necessary signal interaction to perform expected function in all the states.



Architecture of RISC-V

- There are sixteen 16-bit general purpose registers names R0 through R15.
- The register array has two read and one write ports. When ‘reg_wr’ signal is enabled, the data is written into a register indicated by the write address.
- Otherwise two registers indicated by the read addresses are read.
- The ALU performs 11 arithmetical and logical operations.
- Some of the operations require two operands, while others need only one.
- All the operands for ALU operations are provided by registers and the result of operation is written back into the specified destination register through the multiplexer.

02

INSTRUCTION SET AND OPERATIONS

OPCODE	FUNCTION	HEXADECIMAL
0	STOP	0
1	ADD	1
2	SUB	2
3	MUL	3
4	AND	4
5	OR	5
6	XOR	6
7	NOT	7
8	SLLI	8
9	SLRI	9
10	INC	A
11	DEC	B
12	MVI	C
13	LW	D
14	SW	E
15	JUMP	F

Instruction OPCODES

- STOP : Halts the processor
- ADD : Adds the numbers in the registers
- SUB : Subracts the numbers in the registers
- MUL : Multiplies the numbers in the registers
- AND : Bitwise AND operation performed for the numbers in the register
- OR : Bitwise OR operation performed for the numbers in the register
- XOR : Bitwise XOR operation performed for the numbers in the register
- NOT : Inverse of the number in the register is computed
- SLLI : Register data shifted left
- SLRI : Register data shifted right
- INC : Register data is incremented by 1
- MVI : Move immediate operation
- LW : Load Word operation
- SW : Store Word Operation

OPERATIONS

The instructions can be classified into the following five functional categories: data transfer operations, arithmetic operations, logical operations, branching operations and control operations.

- **Data Transfer Operations :** This group of instructions copies data from a location called source to another location called destination without modifying the contents of the source. MVI, LOAD and STORE instructions come under this category.

- **Arithmetic Operations** : These instructions perform arithmetic operations such as addition (ADD), subtraction (SUB), multiplication (MUL), increment (INC) and decrement (DEC).

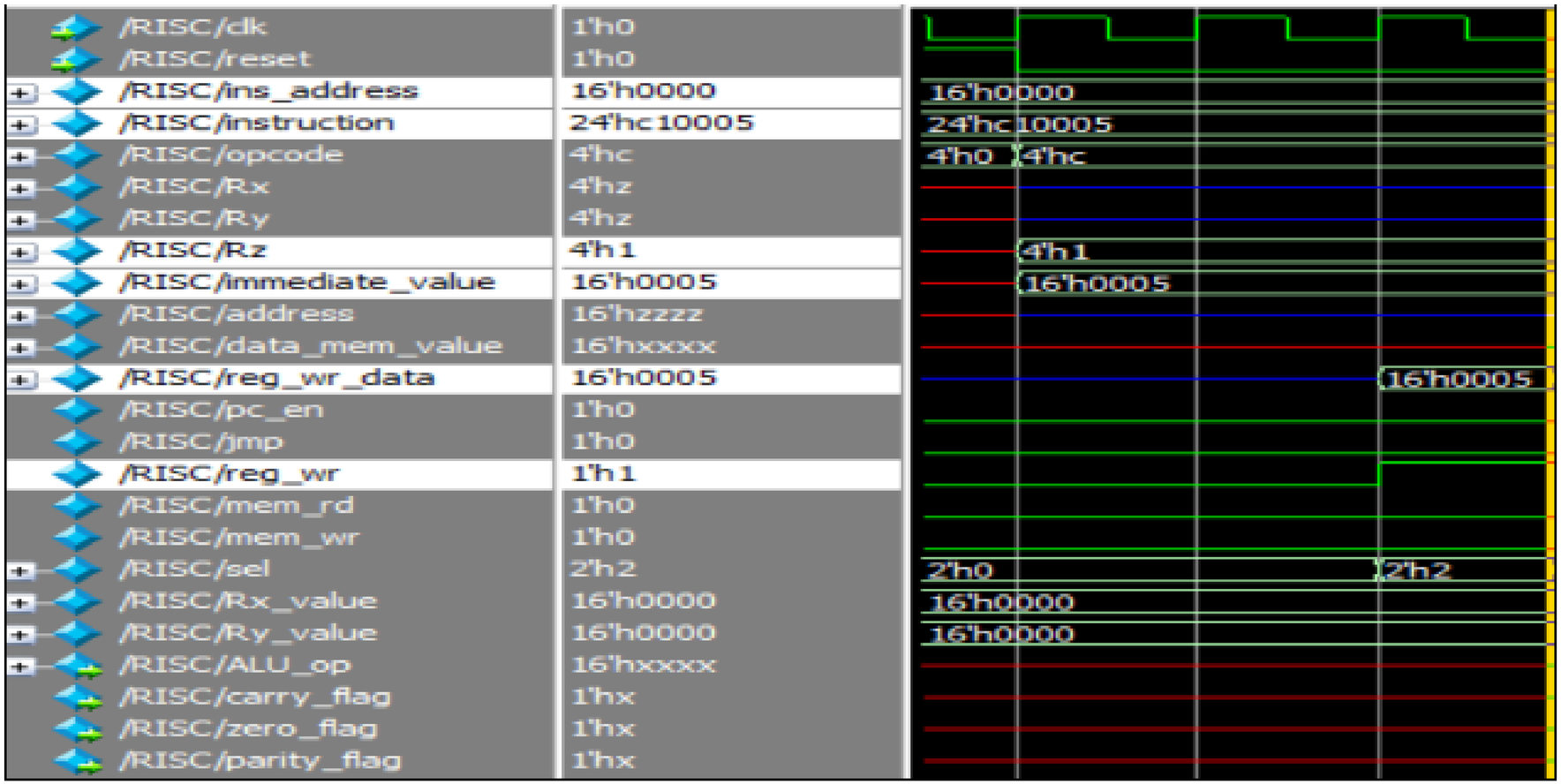
- **Logical Operations :** These instructions perform various logical operations such as AND, OR, XOR, NOT, SHL, SHR

- **Control Operations :** The instruction HLT does not produce any result but stops the program execution.

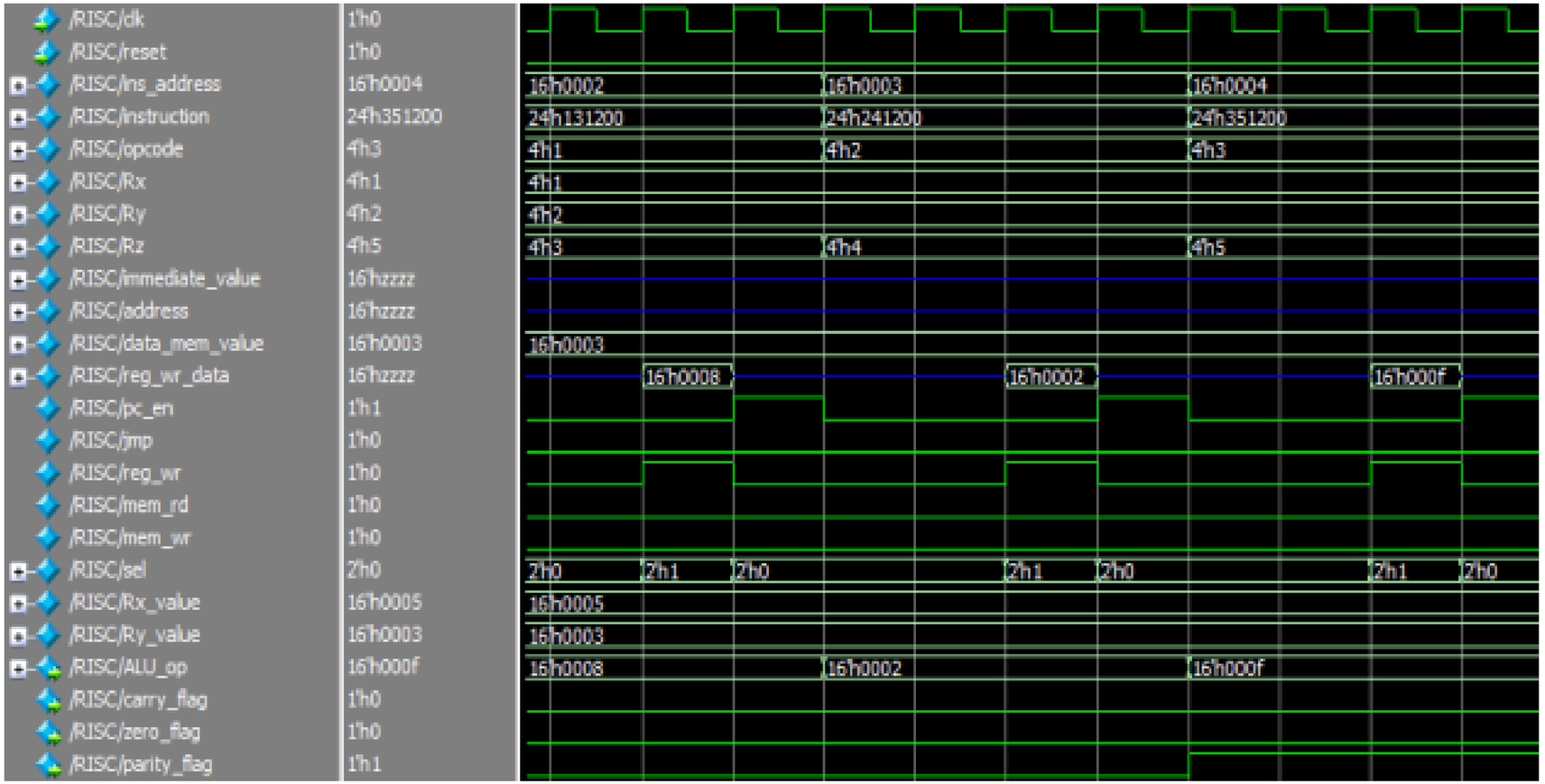
02

SIMULATION RESULTS

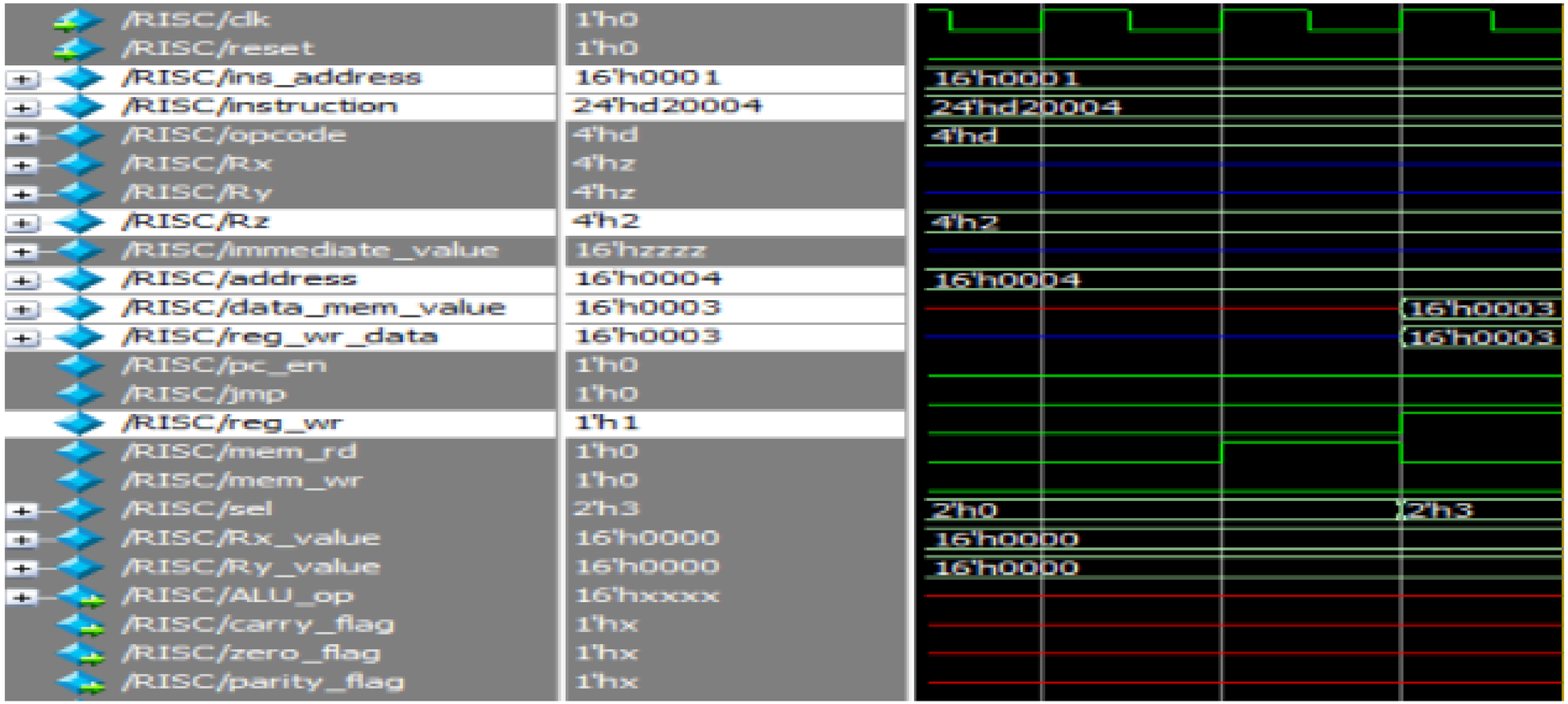
MOVE INSTRUCTION



ARITHMETIC INSTRUCTIONS



LOAD INSTRUCTION



THANK YOU