|  |
| --- |
|  |
| 8 Bit MICROPROCESSOR |
| Implementation using Harvard Architecture and RISC ISA |

|  |
| --- |
|  |

Submitted By:

Karvy Mohnot- 2018KUEC2061

Anushka Joshi- 2018KUEC2062

**Project Aim:** To design and implement a simple microprocessor with a custom instruction set.

**Microprocessor Objective:** To implement an eight bit RISC type microprocessor using Harvard architecture that aims to perform five basic arithmetic and logical operations namely ADD, SUBTRACT, XOR, AND, OR along with other data transfer operations.

**Project Elements:** A microprocessor consists of the following basic elements:

1. **Program Counter**

It is an 8 bit register. It will always increment by 1 in every clock cycle to access the next instruction.

1. **Register file**

This is a set of 4 registers, each capable of storing an 8 bit value.

1. **Instruction memory**

It takes the address bus (8 bit value) as input, and gives out a 16-bit value that is the instruction to be decoded. The address is provided by the Program Counter (PC).

1. **Data Memory**

During a clock cycle, you are either reading from or writing into the data memory, with the address given by the address bus.

1. **Arithmetic and Logic Unit (ALU)**

It is the core of the processor. It performs all the arithmetic and logical computations.

1. **Control Unit (CU)**

The unit that actually makes the entire processor work as expected. The instruction word serves as the input for the CU, and the output is a set of control signals that decide the operation to be performed.

S2

S3

S5

1

MUX

2X1

MUX

2X1

DEMUX

**ALU**

**REGISTER FILE**

RsAddr

RdAddr Rout

Wdata

MUX

2X1

**DATA MEMORY**

WAddr

RAddr dataout

datain

MUX

2X1

MUX

2X1

DEMUX

Adder

**Instruction Memory**

**PC**

**Instruction Decoder**

Rreg

Wreg

OpReg

ImmtoRegdata

ImmtoMemdata

MemWAdd

MemRAdd

ALUset

OpData

**OUTPUT**

Opcode

InsType

MemWrite

S0

S1

WR

|  |  |
| --- | --- |
|  | **CU** |
| S0 |
| S1 |
| S2 |
| S3 |
| S4 |
| S5 |
| S6 |

S4

S6

Fig: Microprocessor Elemental Layout Description with Signals

Components:

1. Program Counter
2. Instruction Memory
3. Instruction Decoder
4. Data Memory
5. Register Files
6. ALU
7. MUX & DEMUX blocks
8. Adder Block

**ELEMENTAL FUNCTIONING:**

**PROGRAM COUNTER:** It is a register structure that contains the

PCnext

PCcurrent

address pointer value of the current instruction. Each cycle, the

**PC**

value at the pointer is read into the instruction decoder and

the program counter is updated to point to the next instruction.

**MICROPROCESSOR MEMORY:** Harvard Architecture of microprocessor deals with separate data and instruction memory.

1. **INSTRUCTION MEMORY:**

The instruction memory is read only and cannot be written over

by the programmer. It stores up to 4096 instructions using

8 bit addresses and 16 bit data.

**DATA MEMORY**

WAddr

RAddr dataout

datain

1. **DATA MEMORY:**

The data memory is read and write type. It stores data using

8 bit address and 8 bit data.

Signals:

1. WAddr: Write Address
2. RAddr: Read Address
3. datain: data to be written
4. dataout: data read

**INSTRUCTION DECODER:**

It reads the next instruction in from memory, and sends the component pieces of that instruction to the necessary destinations. It takes 16 bit input from the Instruction Memory.

1. **RReg-**

Address of memory where immediate data needs to be stored

Signal for: Rsaddr

1. **WReg-**

Address of memory where immediate data needs to be stored

Signal for: Rdaddr

1. **OpRreg-**

Data stored in Operand 1

Signal for: Rsaddr

1. **ImmtoRegData-**

|  |
| --- |
| **INSTRUCTION DECODER** |
| RReg |
| WReg |
| OpReg |
| ImmtoRegData |
| ImmtoMemData |
| MemWAdd |
| MemRAdd |
| ALUSel |
| OpData |
| **OUTPUTS** |
| Opcode |
| InsType |

Immediate data to be stored in register

Signal for: Wdata

1. **ImmtoMemData-**

Immediate data to be stored in memory

Signal for: datain

1. **MemWAdd-**

Address of memory where data from the register needs to be stored

Signal for: Waddr

1. **MemRAdd-**

Address of memory from which data to be stored in register

Signal for: Raddr

1. **ALUSel-**

Control signal for ALU

Signal for: To sel in ALU

1. **OpData-**

Operand for operation performed by ALU

Signal for: To MUX for ALU Input

**Outputs from Instruction Decoder:**

1. **Opcode**
2. **InsType**

These are inputs for: Control Unit, these help CU in instruction decoding and execution.

**REGISTER FILE:** A register file is the component that contains all the general purpose registers of the microprocessor. Register file in our microprocessor contains four general purpose registers each capable of storing an eight bit value.

**REGISTER FILE**

RsAddr

RdAddr Rout

Wdata

Signals:

1. RsAddr: Source Address
2. RdAddr: Destination Address
3. Rout: Source data read
4. Wdata: Data to be written

**ARITHMETIC AND LOGIC UNIT (ALU):** ALU is the block that performs all the arithmetic and logical expressions of the microprocessor. The inputs to an ALU are the data to be operated on, called [operands](https://en.wikipedia.org/wiki/Operand), and a code indicating the operation to be performed; the ALU's output is the result of the performed operation. Size of ALU in our microprocessor is 8 bit. It performs 5 operations ADD, SUBTRACT, AND, OR and XOR; the operation performed is defined according to the instruction format explained in later section.

ALUSel

ALUOut

ALUOp

R03

ALU

**CONTROL UNIT:**

|  |  |
| --- | --- |
|  | **CU** |
| S0 |
| S1 |
| S2 |
| S3 |
| S4 |
| S5 |
| S6 |

The control unit is responsible for setting all the control signals so that each instruction is executed properly.

Inputs to the Control Unit:

Opcode

1. Opcode- 2 bits

Instype

1. Instype- 2 bits

The instruction word serves as the input for the CU, and the output is a

set of control signals that decide the operation to be performed.

**MUX & DEMUX:**

**INSTRUCTION SET ARCHITECTURE:**

**Addressing modes:** The instruction set architecture has been devised in a way such that the first operand is bound to be accessed from register in both data transfer and arithmetic and logical instructions.

Available operand addressing modes:

1. Direct Register addressing
2. Indirect Memory addressing
3. Immediate addressing

**Instruction Format:**

|  |  |  |
| --- | --- | --- |
| A15 | A14 | Instruction Type |
| 0 | 0 | Data Transfer Instructions |
| 0 | 1 | Arithmetic & Logic Instructions |
| 1 | 0 | HLT, NOP Instructions (not implemented) |
| 1 | 1 | Don’t Care |

1. **Data Transfer Instructions:**

Instruction Size: 2 Bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| 0 | 0 | Type of Data Transfer | | Specify the Register Number | | X | X | -----Data or Memory Address----- | | | | | | | |

**Type of Data Transfer:**

|  |  |  |
| --- | --- | --- |
| **A13** | **A12** | **Type of Data Transfer** |
| **0** | **0** | Immediate to Register |
| **0** | **1** | Immediate to Memory |
| **1** | **0** | Memory to Register |
| **1** | **1** | Register to Memory |

1. **Arithmetic & Logical Operations based Instructions:**

Instruction Size: 2 Bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| 0 | 1 | Decides if A7-A0  Are data bits or memory | Selects the Operation to be performed by ALU | | | Specifies the source register | | -----Data or Memory Address----- | | | | | | | |

|  |  |
| --- | --- |
| **A13** | **A7-A0** |
| 0 | Immediate Data |
| 1 | Memory |

|  |  |  |  |
| --- | --- | --- | --- |
| **A12** | **A11** | **A10** | **Operation to be performed** |
| 0 | 0 | 0 | Addition |
| 0 | 0 | 1 | Subtraction |
| 0 | 1 | 0 | AND |
| 0 | 1 | 1 | OR |
| 1 | 0 | 1 | XOR |

The results of all the arithmetic and logical instructions will be stored in the source register by modifying the source data.