16-bit Processor in VHDL

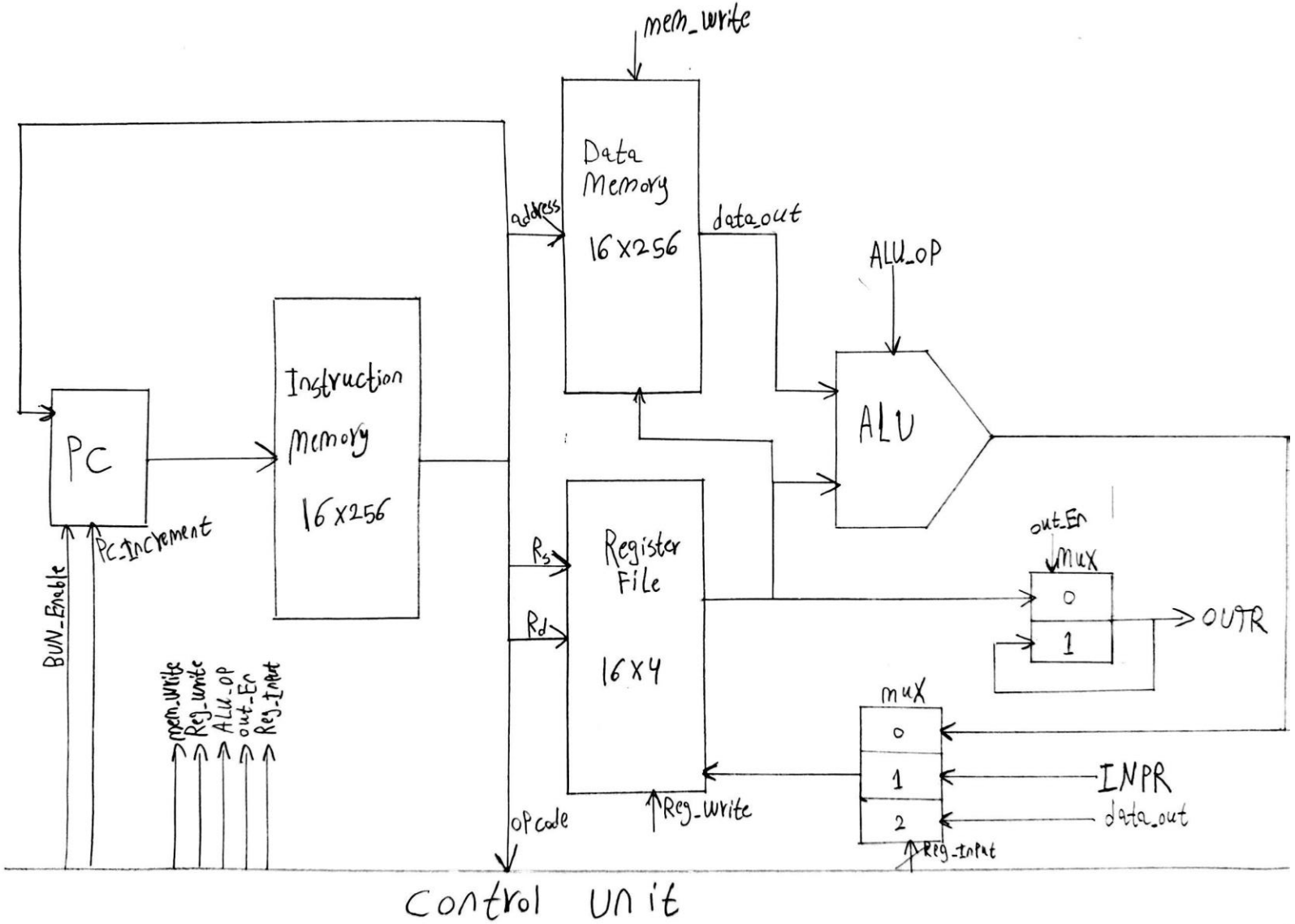
Made By:

Hazem Mohamed Ahmed, 221006745

Abdelrhman Yasser Ali, 221004349

Eyad Hany Aboulnasr, 221005710

Youssif Hany Sayed, 221005893


mem_write
Data Memory
16x256
address
data_out
ALU_OP
ALU
Instruction Memory
16x256
PC
PC_Increment
BUN_Enable
Register File
16x4
Rs
Rd
out_En
mux
0
1
OUTR
mux
0
1
2
INPR
data_out
Reg_Input
mem_write
Reg_write
ALU_OP
out_En
Reg_Input
opcode
Reg_write
Control Unit

1. Introduction

This project implements a 16-bit processor in VHDL. The design includes:

- A Program Counter (PC) that holds the address of the current instruction.
- An Instruction Memory (ROM) storing 16-bit instructions.
- A Control Unit that decodes instructions and generates control signals.
- A Register File with multiple 16-bit registers.
- A Data Memory (RAM) for loading and storing data.
- An Arithmetic Logic Unit (ALU) handling logical/arithmetic operations.
- A set of muxes for routing data (e.g., register input, output).
- An INPR input register and an OUTR output register (or signal).

The CPU executes instructions in a basic Fetch–Decode–Execute cycle:

1. Fetch: The PC outputs its address to the Instruction Memory; the instruction is read.
2. Decode: The high bits of the instruction (opcode) go to the Control Unit, which sets the control signals for the datapath.
3. Execute: The processor performs the operation (e.g., ALU operation, memory load, register write), possibly updates the PC, and completes the cycle.

2. Methodology & Design Flow

Modular Approach:
Each hardware block (PC, Instruction Memory, Data Memory, Register File, ALU, etc.) was coded as a separate VHDL module (entity-architecture). This improves readability and allows easy testing of each component.

Register-Transfer Level (RTL) Organization:
- The PC increments or loads a branch address.
- Instruction Memory outputs instructions based on PC.
- The Control Unit decodes the opcode and sets signals.
- Register File reads Rs and writes to Rd if enabled.
- Data Memory is accessed by the lower 8 bits of the instruction.

ALU operations include:
- Logical (AND, OR, XOR, NOT)
- Shifts (SHL, SHR)
- Arithmetic (ADD, SUB)

3. Fetch–Decode–Execute Implementation

The control signals are sequenced by a small state counter (SC) in the Control Unit:

- State 0 (Fetch): Read instruction from memory to internal register.
- State 1 (Decode): Extract opcode and registers from the instruction.
- State 2 (Execute): Perform ALU or memory operations.
- State 3 (Finish): Update PC and reset to fetch the next instruction.

4. Instruction Format and Opcodes

Each instruction is 16 bits wide. The format is shown below:

Instruction Format (16 bits)	
[15..12] Opcode	[11..10] Rd (Destination Register)
[9..8] Rs (Source Register)	[7..0] Address / Immediate / Second Operand Address

Example Breakdown:

Instruction: 1011_00_01_00000101

- Opcode (1011) = ADD
- Rd (00) = Register 0
- Rs (01) = Register 1
- Address (00000101) = Memory address 5

Sample Opcodes

Opcode	Mnemonic	ALU Op	Action
0001	LDA	n/a	Load from Data_Memory([7..0]) → Rd
0010	INP	n/a	Load INPR → Rd
0011	OUT	n/a	Drive Rs → OUTR
0100	XOR	000	R(Rd) = R(Rs) xor DataMem([7..0])
0101	BUN	n/a	PC = [7..0]
0110	AND	001	R(Rd) = R(Rs) and DataMem([7..0])
0111	OR	010	R(Rd) = R(Rs) or DataMem([7..0])
1000	NOT	011	R(Rd) = NOT(Rs)
1001	SHL	100	Shift-left R(Rs) by 1 bit
1010	SHR	101	Shift-right R(Rs) by 1 bit
1011	ADD	110	R(Rd) = R(Rs) + DataMem([7..0])
1100	SUB	111	R(Rd) = R(Rs) - DataMem([7..0])

5. Conclusion

This project demonstrates a CPU architecture in VHDL. Key points:

- Modular design for easier debugging and development.
- A 16-bit instruction format supporting multiple operations.
- Control Unit manages execution flow using a state machine.
- ALU supports logical, shift, and arithmetic operations.