EGRE 426 – Lab 4 Week 1 Report

Due: November 12, 2025, 11:59 PM

Team Members:

Macarios Atia

Ansh

**1. Objective**

The goal of Week 1 is to design the instruction set and define the components, and make sure that the control unit table for a 16-bit single-cycle RISC processor actually works .This design forms the basics for our cpu where the cpu will be simulated and integrated within our code.

**2. Instruction Set Architecture (ISA) Design**

**Instruction Formats**

|  |  |  |
| --- | --- | --- |
| Type | Format | Description |
| R-Type | Opcode (4 bits)] Rs (4 bits) Rt (4 bits)] [Rd (4 bits) | arithmetic/logic |
| I-Type | Opcode (4 bits)] [Rs (4 bits)] [Rt (4 bits) Immediate (4 bits) | immediate or memory access |
| J-Type | Opcode (4 bits) Address (12 bits) | for jumps |

|  |
| --- |
| Word Size: 16 bits |
| Memory Size: 512 bytes |
| Register File: 8 general-purpose registers (R0–R7) |

**3. Instruction and Control Signal Table**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction | Opcode | reg\_dst | jump | branch | mem\_read | mem\_to\_reg | ALU\_OP | mem\_write | alu\_src | reg\_write |
| Add Signed | 0000 | 1 | 0 | 0 | X | 0 | 10 | X | 0 | 1 |
| Subtract Signed | 0000 | 1 | 0 | 0 | X | 0 | 10 | X | 0 | 1 |
| AND | 0000 | 1 | 0 | 0 | X | 0 | 10 | X | 0 | 1 |
| OR | 0000 | 1 | 0 | 0 | X | 0 | 10 | X | 0 | 1 |
| Shift Left Logical | 0000 | 1 | 0 | 0 | X | 0 | 10 | X | 0 | 1 |
| Shift Right Logical | 0000 | 1 | 0 | 0 | X | 0 | 10 | X | 0 | 1 |
| Shift Right Arithmetic | 0000 | 1 | 0 | 0 | X | 0 | 10 | X | 0 | 1 |
| Exclusive OR | 0000 | 1 | 0 | 0 | X | 0 | 10 | X | 0 | 1 |
| Load Word | 0001 | 0 | 0 | 0 | 1 | 1 | 00 | 0 | 1 | 1 |
| Store Word | 0010 | 0 | 0 | 0 | 0 | 1 | 00 | 1 | 1 | 1 |
| Add Immediate | 0011 | 0 | 0 | 0 | X | X | 00 | X | 1 | 1 |
| Branch | 0100 | X | 0 | 1 | X | X | 01 | X | 1 | X |
| Branch Greater Than | 0101 | X | 0 | 1 | X | X | 01 | X | 1 | X |
| Branch On Equal | 0110 | X | 0 | 1 | X | X | 01 | X | 1 | X |
| Branch Greater Than or Equal | 0111 | X | 0 | 1 | X | X | 01 | X | 1 | X |
| Jump | 1000 | X | 1 | 0 | X | X | 11 | X | X | X |

**4. Datapath Components and Description**

|  |  |  |
| --- | --- | --- |
| Component | Function | Notes |
| Program Counter (PC) | Holds address of next instruction | 16-bit register, increments by 2 |
| Instruction Memory | Stores machine instructions | 512 bytes total |
| Register File | Holds operands and results | 8 registers, each 16 bits |
| ALU | Performs arithmetic/logic operations | Supports add, sub, and, or, xor, slt |
| Data Memory | Stores data for lw/sw instructions | Addressed by ALU output |
| Multiplexers | Select data/control sources | Used for reg\_dst, alu\_src, mem\_to\_reg |
| Sign Extender | Extends immediate fields | 4 → 16 bits |
| Control Unit | make control signals based on opcode | Controls datapath flow |

**5. Register Definitions**

|  |  |  |  |
| --- | --- | --- | --- |
| Register | Symbol | Description | Size |
| R0 | $v0 | General-purpose (result register) | 16 bits |
| R1 | $v1 | General-purpose | 16 bits |
| R2 | $v2 | General-purpose | 16 bits |
| R3 | $v3 | General-purpose | 16 bits |
| R4 | $t0 | Temporary data register | 16 bits |
| R5 | $a0 | Address/data register | 16 bits |
| R6 | $a1 | Loop counter | 16 bits |
| R7 | R | Reserved register | 16 bits |

**6. Variable Initialization (from Pseudocode)**

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Register | Initial Value (hex) | Description |
| $v0 | R0 | 0040 | Used for division and OR operations |
| $v1 | R1 | 1010 | Used in OR logic |
| $v2 | R2 | 000F | Multiplied by 4 during loop |
| $v3 | R3 | 00F0 | XORed with $v2 |
| $t0 | R4 | 0000 | Temporary memory holder |
| $a0 | R5 | 0010 | Memory pointer (increments by 2) |
| $a1 | R6 | 0005 | Loop counter (decrements each iteration) |

**7. Memory Initialization**

|  |  |  |
| --- | --- | --- |
| Address | Data (hex) | Description |
| $a0 | 0101 | Data element 1 |
| $a0 + 2 | 0110 | Data element 2 |
| $a0 + 4 | 0011 | Data element 3 |
| $a0 + 6 | 00F0 | Data element 4 |
| $a0 + 8 | 00FF | Data element 5 |

**8. Control Unit Inputs and Outputs**

|  |  |
| --- | --- |
| Input | Description |
| Opcode [15–12] | instruction type |
| Output | Description |
| reg\_dst | Select destination register |
| jump | Jump control |
| branch | Branch control |
| mem\_read | Enables data memory read |
| mem\_to\_reg | Selects memory data for write-back |
| ALU\_OP | Finds ALU function |
| mem\_write | Enables data memory write |
| alu\_src | Selects ALU input (register or immediate) |
| reg\_write | Enables register file write |

**9. Summary**

The instruction formats, datapath components, and control signals were fully defined.All registers, memory initializations, and variable mappings were clearly shown above in the tables .The setup we have right now makes sure that simulator and control logic can be implemented and tested in Week which should give us enough time to work on the pipelining of our cpu with the buffers added.