# I. Giới thiệu chung

- RISC-V là một kiến trúc tập lệnh (ISA) phần cứng mã nguồn mở dựa trên kiến trúc tập lệnh máy tính với tập lệnh đơn giản hóa (RISC). RISC là một kiến trúc tập lệnh có các đặc điểm sau:

* Số lượng lệnh cơ bản ít
* Mỗi lệnh được thực hiện trong một chu kỳ đồng hồ (clock cycle).
* Các lệnh có kích thước cố định

RISC-V kế thừa các đặc điểm này của RISC, nhưng nó cũng có một số cải tiến:

* Khả năng mở rộng: RISC-V có một kiến trúc mở cho phép các nhà phát triển thêm các tính năng và chức năng mới.
* Khả năng tùy biến: RISC-V có thể được tùy chỉnh để đáp ứng các yêu cầu cụ thể của các ứng dụng khác nhau.

RISC-V là một dự án mã nguồn mở, có nghĩa là bất kỳ ai cũng có thể sử dụng, sửa đổi và phân phối nó miễn phí. Điều này có một số lợi ích, bao gồm:

* Tính cạnh tranh: Khả năng truy cập miễn phí vào RISC-V đã thúc đẩy sự cạnh tranh giữa các nhà cung cấp, dẫn đến giá cả thấp hơn và hiệu suất cao hơn.
* Tính đổi mới: Khả năng tùy biến của RISC-V đã khuyến khích sự đổi mới, dẫn đến sự phát triển của các ứng dụng mới và sáng tạo.
* Tính khả dụng: RISC-V có sẵn trên nhiều nền tảng khác nhau, bao gồm máy tính, điện thoại thông minh, thiết bị IoT và máy móc công nghiệp.

- Đề tài: Thiết kế bộ xử lý pipelined theo kiến trúc tập lệnh RISC-V 32 bits.

- Các chức năng:

* Thực thi các lệnh RISC-V cơ bản, bao gồm phép toán logic, so sánh, nhảy, nạp/lưu dữ liệu, v.v.
* Hỗ trợ các thanh ghi tổng cộng, thanh ghi lệnh, thanh ghi dữ liệu, v.v.
* Hỗ trợ bộ nhớ chương trình và bộ nhớ dữ liệu.

- Các bước tiến hành:

1. Nghiên cứu tập lệnh RISC-V, xác định rõ định dạng mã hóa lệnh và các kỹ thuật thiết kế bộ xử lý pipelined.
2. Phân tích các chức năng cần thiết và kiến trúc của bộ xử lý .
3. Thiết kế các đơn vị chức năng (ALU,RegisterFile,...) của bộ xử lý và xác định các tín hiệu điều khiển.
4. Thiết kế khối datapath từ các đơn vị chức năng.
5. Thiết kế khối controller.
6. Mô phỏng hoạt động của bộ xử lý.
7. Test trên kit DE2.

- Công cụ sử dụng: Quartus,ModelSim, ngôn ngữ Verilog.

- Test plan:

* Kiểm tra các lệnh cơ bản
* Kiểm tra các phép toán logic và so sánh
* Kiểm tra các lệnh nhảy
* Kiểm tra các lệnh nạp/lưu dữ liệu
* Kiểm tra hiệu suất/timing

# II. Giới thiệu sơ lược về đề tài

- Sử dụng cơ chế pipeline 5 giai đoạn (Fetch,Decode,Execute,Memory,Writeback).

- Các thành phần phần cứng đặc trưng:

+ PC counter : chứa địa chỉ của các lệnh được thực thi.

+ Instructions Memory: vùng nhớ lưu trữ lệnh

+ Register File: một tập 32 thanh ghi, mỗi thanh ghi có chiều dài 32 bit (4 byte) có thể được đọc hoặc ghi bằng địa chỉ, với mỗi lệnh cho phép đọc nhiều nhất từ hai thanh ghi và cho phép ghi vào nhiều nhất 1 thanh ghi. Có tín hiệu điều khiển cho phép việc đọc/ghi.

+ Sign Extend: khối mở rộng immediate 16 bit thành 32 bit (dành cho các lệnh I-type).

+ Khối ALU: chức năng chính là lấy hai dữ liệu đầu vào (32 bit) và thực hiện tính toán (theo opcode với từng lệnh riêng biệt) và đưa ra kết quả (32 bit), thêm vào đó là 1 bit kết quả nhánh để dùng cho các lệnh rẽ nhánh/lệnh nhảy.

+ Data Memory: vùng nhớ lưu trữ dữ liệu, có tín hiệu cho phép đọc/ghi dữ liệu (đối với các lệnh load – store).

+ 4 pipeline registers (IF-ID/ID-EX/EX-MEM/MEM-WB): kết nối 5 stage của processor. Mỗi pipeline registers sẽ có các ngõ vào và ngõ ra tương ứng với hai stage mà nó kết nối, chi tiết về các registers này sẽ được thể hiện chi tiết trong final report.

# III. Giới thiệu tập lệnh

- Định dạng mã hóa lệnh (encoding format):

A table with numbers and letters

Description automatically generated

- Tập lệnh:

|  |  |  |
| --- | --- | --- |
| Instruction | Type | Description |
| ADD | R-Type | Addition |
| SUB | R-Type | Substitution |
| SLL | R-Type | Shift Left Logic |
| SLT | R-Type | Set Less Than |
| SLTU | R-Type | Set Less Than Unsigned |
| XOR | R-Type | Bitwise XOR |
| SRL | R-Type | Shift Right Logic |
| SRA | R-Type | Shift Right Arithmetic |
| OR | R-Type | Bitwise OR |
| AND | R-Type | Bitwise AND |
| MUL (\*) | R-Type | Multiplication |
| MULH (\*) | R-Type | Multplication signed x signed and return upper half (32 bits) |
| MULHSU (\*) | R-Type | Multplication signed x unsigned and return upper half (32 bits) |
| MULHU (\*) | R-Type | Multplication unsigned x unsigned and return upper half (32 bits) |
| DIV (\*) | R-Type | Integer Divition |
| REM (\*) | R-Type | Remainder of signed integer devition |
| REMU (\*) | R-Type | Remainder of unsigned integer devition |
| LB | I-Type | Load Byte |
| LH | I-Type | Load Half word (2 Bytes) |
| LW | I-Type | Load Word (4 Bytes) |
| LBU | I-Type | Load Byte Unsigned |
| LHU | I-Type | Load Halfword Unsigned |
| ADDI | I-Type | Addition Immediate |
| SLTI | I-Type | Set Less Than By Immediate |
| SLTIU | I-Type | Set Less Than Unsigned By Immediate |
| XORI | I-Type | Bitwise XOR with Immediate |
| ORI | I-Type | Bitwise OR with Immediate |
| ANDI | I-Type | Bitwise AND with Immidiate |
| SLLI | I-Type | Shift Left Logic By Immediate |
| SRLI | I-Type | Shift Right Logic Immediate |
| SRAI | I-Type | Shift Right Arithmetic Immediate |
| JALR (\*) | I-Type | Jump & Link Register |
| SB | S-Type | Store Byte |
| SH | S-Type | Store Halfword |
| SW | S-Type | Store word |
| SBU (\*) | S-Type | Store unsigned Byte |
| SHU (\*) | S-Type | Store unsigned Half word |
| BEQ | B-Type | Branch if equal |
| BNE | B-Type | Branch if not equal |
| BLT | B-Type | Branch Less Than |
| BGE | B-Type | Branch Greater Than or equal |
| BLTU | B-Type | Branch Less Than Unsigned |
| BGEU | B-Type | Branch Greater Than or equal unsigned |
| LUI | U-Type | Load Upper Immidiate |
| AUIPC | U-Type | Add Upper Immidiate to PC |
| JAL | J-Type | Jump and Link (Unconditional Jumps) |
| FENCE (\*) |  | Fence - This to ensure all the operation before FENCE observed before operation after the Fence |
| FENCE.I (\*) |  | Fence Instruction |

(\*): các lệnh mới được xây dựng dựa trên tính chất open source của RISC-V, tùy thuộc vào nhu cầu sử dụng của nhóm. Tuy nhiên có thể sẽ không hoàn thiện được tất cả nếu có khó khăn trong quá trình thực hiện.