|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  logo_128  **MÔN HỌC: KIẾN TRÚC MÁY TÍNH**  **Giảng Viên Hướng dẫn:** TS. Tạ Thị Kim Huệ  **Đề Tài:**  **Mô phỏng đường dữ liệu bộ xử lý RISC-V cho các chương trình viết bằng ngôn ngữ C**  **Sinh viên thực hiện:**   |  |  | | --- | --- | | Họ và tên : | Nguyễn Thanh Lâm | | MSSV : | 20172644 | | Lớp : | ĐTVT.06 – K62 | | Email : | Lam.nt172644@sis.hust.edu.vn | |  |  |   Hà Nội, 1/2022 |

LỜI MỞ ĐẦU

Trải qua một kỳ học, chúng em đã được TS. Tạ Thị Kim Huệ giảng dạy bộ môn Kiến trúc máy tính. Được hiểu thêm về cấu trúc của bộ xử lý RISC-V, hay biết thêm hiệu năng của cache, … Chúng em cảm ơn cô rất nhiều đã giúp đỡ cho chúng em thời gian qua.

Trong các đề tài bảo vệ môn học, chúng em chọn đề tài “Mô phỏng đường dữ liệu bộ xử lý RISC-V cho chương trình C bằng Ripes”. Để tìm hiểu sâu sắc hơn quá trình biên dịch một chương trình C và nắm rõ cấu trúc xử lý RISC-V.

Cùng với việc thực hiện đề tài này, em xin gửi lời cảm ơn sâu sắc đến giảng viên hướng dẫn TS. Tạ Thị Kim Huệ đã nhiệt tình chỉ dẫn các bước, hướng nghiên cứu, thực hiện cũng như yêu cầu cần có của đề tài trong suốt thời gian thực hiện đề tài. Trong quá trình thực hiện, dù đã rất cố gắng nhưng không tránh khỏi những sai sót và hạn chế nhất định. Vì vậy chúng em rất mong nhận được sự góp ý, bổ sung thêm của cô để đề tài được hoàn thiện hơn

MỤC LỤC

[CHƯƠNG 1: TỔNG QUAN VỀ KIẾN TRÚC MÁY TÍNH 4](#_Toc92190360)

[1. Khái niệm chung về kiến trúc máy tính. 4](#_Toc92190361)

[2. Quá trình biên dịch 5](#_Toc92190362)

[CHƯƠNG 2: TỔNG QUAN VỀ PHẦN MỀM RIPES 7](#_Toc92190363)

[1. Giới thiệu khái quát về Ripes 7](#_Toc92190364)

[2. Các tác vụ chính trong Repis 7](#_Toc92190365)

[2.1. The Editor Tab 7](#_Toc92190366)

[2.2. The Processor Tab 8](#_Toc92190367)

[2.3. Tab Memory 8](#_Toc92190368)

[2.4. Tab Cache 9](#_Toc92190369)

[2.5. Tab I/O 10](#_Toc92190370)

[CHƯƠNG 3: MÔ PHỎNG VÀ THỰC THI CHƯƠNG TRÌNH TRÊN RIPES 12](#_Toc92190371)

[1. Toolchain. 12](#_Toc92190372)

[2. Toolchain Registration 12](#_Toc92190373)

[3. Compiling and Executing a C program 13](#_Toc92190374)

[CHƯƠNG 4: KẾT LUẬN 17](#_Toc92190375)

[TÀI LIỆU THAM KHẢO 17](#_Toc92190376)

DANH MỤC HÌNH ẢNH

[Hình 1: Thiết kế một đường ống của kiến trúc MIPS 6](#_Toc92191333)

[Hình 2: Hệ thống phân cấp dịch cho ngôn ngữ C 7](#_Toc92191334)

[Hình 3: The Editor Tab 9](#_Toc92191335)

[Hình 4: The Processor Tab 10](#_Toc92191336)

[Hình 5: Tab Memory 11](#_Toc92191337)

[Hình 6: Tab Cache 12](#_Toc92191338)

[Hình 7: Tab I/O 12](#_Toc92191339)

[Hình 8: Toolchain Registration 15](#_Toc92191340)

[Hình 9: Chương trình C 15](#_Toc92191341)

[Hình 10: Mô phỏng, thực thi chương trình 16](#_Toc92191342)

[Hình 11: Mô phỏng đường dữ liệu 16](#_Toc92191343)

[Hình 12: Mô phỏng bộ dữ liệu đệm 17](#_Toc92191344)

[Hình 13: Memory 17](#_Toc92191345)

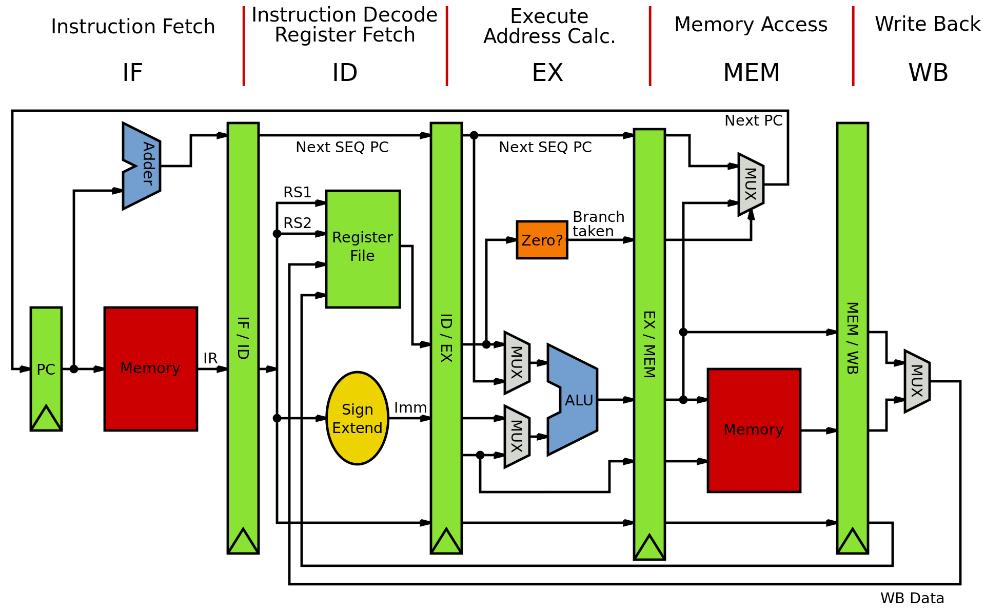
[Hình 14: I/O 18](#_Toc92191346)

CHƯƠNG 1: TỔNG QUAN VỀ KIẾN TRÚC MÁY TÍNH

1. Khái niệm chung về kiến trúc máy tính.

Kiến trúc máy tính là thiết kế khái niệm và cấu trúc hoạt động căn bản của một hệ thống máy tính. Nó là một bản thiết kế (blueprint) mô tả có tính chất chức năng về các yêu cầu (đặc biệt là tốc độ và các kết nối tương hỗ) và những sự thi hành thiết kế cho những bộ phận khác nhau của một máy tính - tập trung chủ yếu vào việc CPU hoạt động nội tại như thế nào và truy cập các địa chỉ trong bộ nhớ bằng cách nào.

Nó cũng có thể được định nghĩa như là khoa học và nghệ thuật lựa chọn và kết nối các thành phần phần cứng để tạo thành các máy tính đáp ứng được các mục đích về tính năng, hiệu suất và giá cả.

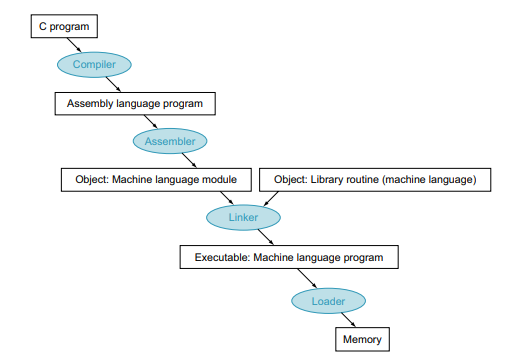


Hình 1: Thiết kế một đường ống của kiến trúc MIPS

Kiến trúc máy tính bao gồm ít nhất ba phạm trù con chính

* Kiến trúc tập lệnh :là hình ảnh trừu tượng của một hệ thống tính toán được nhìn từ góc độ của một lập trình viên sử dụng  ngôn ngữ máy, bao gồm tập lệnh, cách đánh địa chỉ ô nhớ (*memory address modes*), các thanh ghi, và các định dạng địa chỉ và dữ liệu.
* Vi kiến trúc (*Microarchitecture*), là một mô tả bậc thấp, cụ thể hơn về hệ thống. Mô tả này nói về các bộ phận cấu thành của hệ thống được kết nối với nhau như thế nào và chúng hoạt động tương hỗ như thể nào để thực hiện kiến trúc tập lệnh[]](https://vi.wikipedia.org/wiki/Ki%E1%BA%BFn_tr%C3%BAc_m%C3%A1y_t%C3%ADnh#cite_note-2). Ví dụ, kích thước bộ đệm cache của một máy tính là một đặc điểm về tổ chức máy tính mà thường không liên quan đến kiến trúc tập lệnh.
* Thiết kế hệ thống (*System Design*) bao gồm tất cả các thành phần phần cứng khác bên trong một hệ thống tính toán chẳng hạn:
* Các đường kết nối hệ thống như bus và switch
* Các bộ điều kiển bộ nhớ (*memory controller*) và các cây phả hệ bộ nhớ
* Các cơ chế CPU off-load như Direct memory access (*truy nhập bộ nhớ trực tiếp)*
* Các vấn đề như đa xử lý (*multi-processing*).

## 2. Quá trình biên dịch



Hình 2: Hệ thống phân cấp dịch cho ngôn ngữ C

* Giai đoạn tiền xử lý – Proprocessor

+ Nhận mã nguồn

+ Xóa bỏ các chú thích, comment của chương trinh

+ Chỉ thị tiền xử lí

* Dịch ngôn nhữ C sang Assembly

+ Phân tích cú pháp của mã nguồn

+ Chuyển chúng sang mã Assembly

* Dịch Assembly

+Dịch chương trinh => Sang mã máy 0 và 1

+ Một tệp mã máy (.obj) sinh ra trong hệ thống sau đó

* Linker

+ Mã máy của 1 chương trinh từ nhiều nguồn (file .c,...) được liên kết lại với nhau để thành một trương trinh duy nhất

+ Mã máy của các hàm thư viện gọi trong chương trinh cũng được đưa vào chương trinh cuối

# CHƯƠNG 2: TỔNG QUAN VỀ PHẦN MỀM RIPES

1. Giới thiệu khái quát về Ripes

Ripes là một trình mô phỏng bộ xử lý đồ họa và trình soạn thảo mã lắp ráp được xây dựng cho kiến ​​trúc tập lệnh RISC-V, phù hợp để dạy cách mã cấp độ lắp ráp được thực thi trên các vi kiến ​​trúc khác nhau.

1. Các tác vụ chính trong Repis
   1. The Editor Tab

Ảnh có chứa bàn

Mô tả được tạo tự động

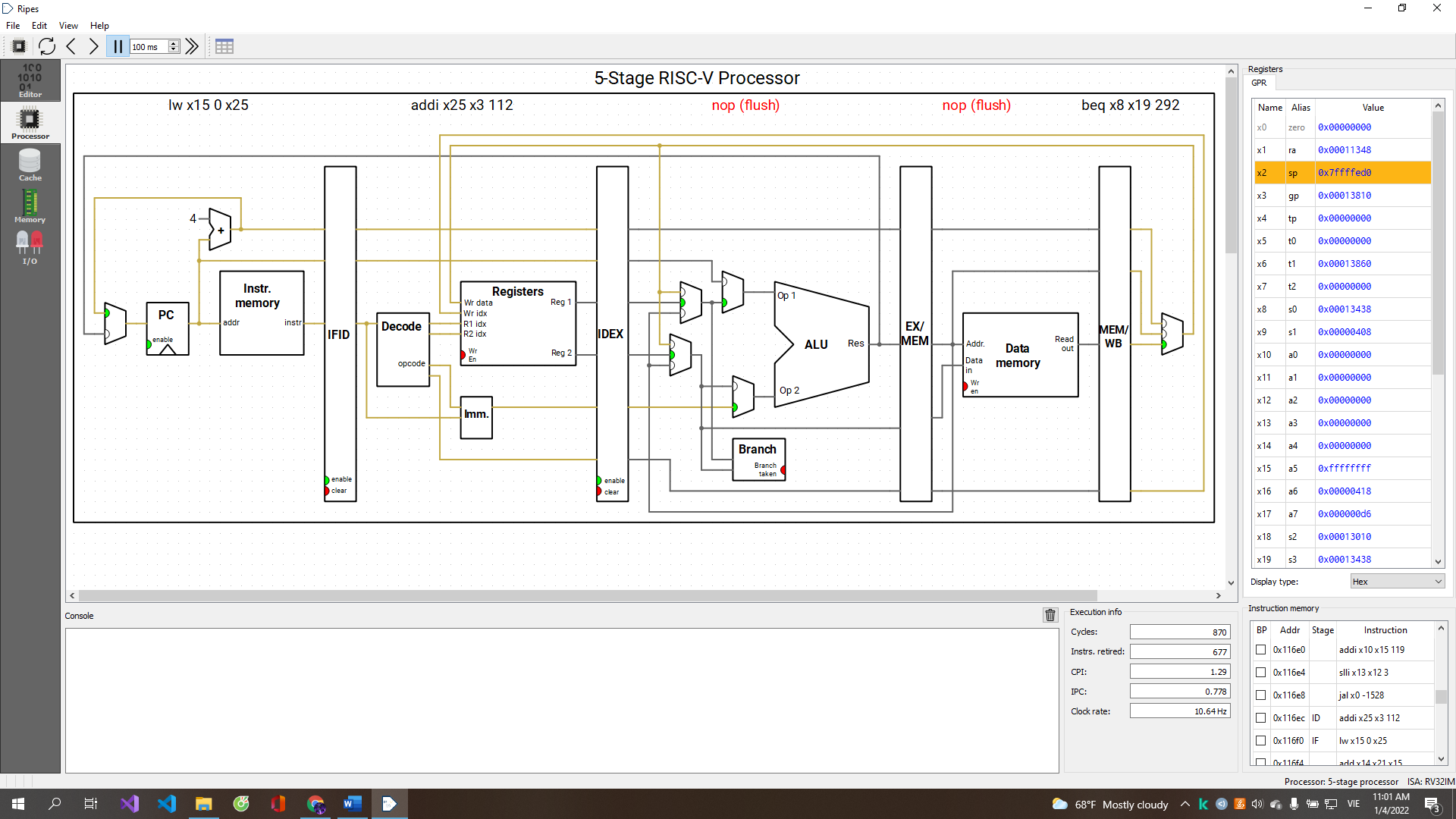
Hình 3: The Editor Tab

Tab trình chỉnh sửa hiển thị hai đoạn mã. Ở phía bên trái, có thể viết một chương trình hợp ngữ được viết bằng các tập lệnh RISC-V RV32 (I/M/C). Bất cứ khi nào thực hiện bất kỳ chỉnh sửa nào trong chương trình hợp ngữ này - và không tìm thấy lỗi cú pháp - mã hợp ngữ sẽ tự động được lắp ráp và chèn vào trình mô phỏng. Nếu một trình biên dịch C đã được đăng ký, thì kiểu đầu vào có thể được đặt thành C. Sau đó, có thể viết, biên dịch và thực thi các chương trình ngôn ngữ C trong Ripes.

Tiếp theo, ở phía bên phải, một chế độ xem mã thứ hai được hiển thị. Đây là chế độ xem không tương tác của chương trình hiện tại ở trạng thái đã lắp ráp của nó, được ký hiệu là trình xem chương trình. Chúng ta có thể xem chương trình đã lắp ráp dưới dạng hướng dẫn RISC-V đã được tháo rời hoặc dưới dạng mã nhị phân thô. Có thể nhấp vào thanh bên màu xanh lam của chế độ xem bên phải để đặt điểm ngắt tại địa chỉ mong muốn. Nhấn biểu tượng sẽ hiển thị danh sách tất cả các biểu tượng trong chương trình hiện tại. Thông qua đó, có thể điều hướng trình xem chương trình đến bất kỳ biểu tượng nào trong số này.

Ripes được đóng gói với nhiều ví dụ khác nhau về chương trình hợp ngữ RISC-V, có thể tìm thấy trong menu File-> Load Examples.

* 1. The Processor Tab



Hình 4: The Processor Tab

Tab Processor là nơi Ripes hiển thị chế độ xem của bộ xử lý hiện được chọn, cũng như bất kỳ thông tin bổ sung nào liên quan đến việc thực thi. Ngoài chế độ xem bộ xử lý, tab bộ xử lý chứa các chế độ xem sau:

* Thanh ghi: Một danh sách tất cả các thanh ghi của bộ xử lý. Giá trị thanh ghi có thể được chỉnh sửa thông qua việc nhấp vào giá trị của thanh ghi đã cho. Việc chỉnh sửa giá trị thanh ghi ngay lập tức được phản ánh trong mạch bộ xử lý. Sổ đăng ký được sửa đổi gần đây nhất được đánh dấu bằng nền màu vàng.
* Bộ nhớ lệnh: Chế độ xem chương trình hiện tại được tải trong trình mô phỏng.
* BP: Các điểm ngắt, nhấp để chuyển đổi. Bất kỳ điểm ngắt nào được đặt trong tab trình chỉnh sửa sẽ được phản ánh ở đây.
* Addr: Địa chỉ của lệnh đã cho
* Stage: Liệt kê (các) giai đoạn hiện đang thực hiện lệnh đã cho
* Instruction: Hướng dẫn tháo rời
* Execution info: Các số liệu thống kê khác nhau dựa trên số lượng chu kỳ và số lượng lệnh ngừng hoạt động hiện tại.
* Solution: Bất kỳ đầu ra nào thông qua chức năng in ecall sẽ được hiển thị ở đây.
  1. Tab Memory

Ảnh có chứa văn bản, máy tính, máy tính xách tay, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 5: Tab Memory

Tab bộ nhớ cung cấp một cái nhìn vào toàn bộ không gian địa chỉ có thể xác định của bộ xử lý. Điều hướng bộ nhớ có thể được thực hiện như sau

* + Cuộn chế độ xem bộ nhớ
  + Tới đăng ký sẽ cuộn chế độ xem bộ nhớ đến giá trị hiện có trong thanh ghi đã chọn
  + Đi tới phần sẽ cuộn chế độ xem bộ nhớ đến địa chỉ của giá trị phần đã cho trong bộ nhớ (tức là. textphân đoạn bộ nhớ lệnh, dataphân đoạn dữ liệu tĩnh, v.v.). Hơn nữa, một địa chỉ tùy chỉnh có thể được chỉ định thông qua tùy chọn"Địa chỉ.".

Bất cứ khi nào bộ xử lý được đặt lại, tất cả bộ nhớ được ghi trong quá trình thực thi chương trình sẽ được đặt lại về trạng thái ban đầu.

* 1. Tab Cache

Ảnh có chứa bàn

Mô tả được tạo tự động

Hình 6: Tab Cache

Kể từ phiên bản 2.1.0, Ripes bao gồm mô phỏng bộ nhớ cache. Trình mô phỏng bộ nhớ đệm mô phỏng bộ đệm L1D (dữ liệu) và L1I (lệnh), trong đó có thể định cấu hình bố trí và hành vi của từng loại bộ đệm. Do đó, chúng tôi có thể phân tích hiệu suất bộ đệm của các chương trình của chúng tôi để xem các thiết kế bộ đệm khác nhau tương tác như thế nào với các mẫu truy cập bộ nhớ mà chương trình của chúng tôi thể hiện.

* 1. Tab I/O

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 7: Tab I/O

Kể từ phiên bản 2.2, Ripes bao gồm các thiết bị I / O được ánh xạ bộ nhớ khác nhau. Thông qua những điều này, có thể nhanh chóng nhận ra một hệ thống nhúng nhỏ. Trang sau đây cung cấp thông tin tổng quan về việc sử dụng các thiết bị cũng như cách tạo thiết bị của riêng bạn.

# CHƯƠNG 3: MÔ PHỎNG VÀ THỰC THI CHƯƠNG TRÌNH TRÊN RIPES

Bất kỳ mô hình bộ xử lý nào trong Ripes không phụ thuộc vào mã do người dùng lập lịch đều có thể thực thi bất kỳ tệp thực thi nào được biên dịch cho tập lệnh mà mô hình bộ xử lý triển khai. Các phần sau sẽ trình bày về cách một chương trình .c có thể được biên dịch và sau đó được thực thi trong trình mô phỏng.

Ví dụ như chương trình ## (được biên dịch trước), hãy tham khảo ranpi.c. Lưu ý rằng một phiên bản biên dịch trước của chương trình này được đóng gói cùng với Ripes như một trong những ví dụ (trong Ripes, chọn File-> Load Example ...-> RanPI).

Đáng chú ý, chương trình này chứa các phép toán dấu phẩy động trong khi bộ xử lý mô phỏng không chứa đơn vị dấu chấm động. Do đó (suy ra bằng cách đặt -march = rv32im, như được thấy bên dưới), trình biên dịch sẽ bao gồm các thói quen soft-float thực hiện các phép toán dấu phẩy động thông qua số học số nguyên.

1. Toolchain.

Ban đầu, hãy lưu ý rằng chuỗi công cụ RISC-V có sẵn thông qua các kho lưu trữ gói tiêu chuẩn như gcc-7-riscv64-linux-gnu sẽ không thể tạo tệp thực thi với phiên bản Ripes hiện tại, ngay cả khi chuyển đổi -march = rv32im được sử dụng

Ripes yêu cầu chuỗi công cụ RISC-V bằng kim loại trần, cụ thể là bộ tuple riscv64- unknown- elf-. Điều này có thể được tìm nạp từ:

* + Tải xuống chuỗi công cụ dựng sẵn cho nền tảng của bạn từ sifive/ freedomtools. Phiên bản này được cho là hoạt động tốt với Ripes.
  + Hoặc, nếu bạn muốn xây dựng chuỗi công cụ của riêng mình, hãy làm theo hướng dẫn https//github.com/riscv/riscv-gnu-toolchain để xây dựng chuỗi công cụ newlib.

Lưu ý: chuỗi công cụ phải được xây dựng-- enable-multilibđể cho phép nhắm mục tiêu các hệ thống RISC-V 32- bit.

1. Toolchain Registration

Với một chuỗi công cụ hợp lệ có sẵn, hãy điều hướng đến (Edit->Settings->Editor). Ở đây, người ta có thể cung cấp một vùng tắm cho một trình biên dịch tương thích có thể thực thi, tức là. tệp thực thi (riscv64-hidden-elf-gcc). Ripes sẽ tự động cố gắng tìm kiếm tệp thực thi trong current PATH. Nếu không tìm thấy trình biên dịch tự động, người ta có thể duyệt và điều hướng đến trình biên dịch thực thi.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 8: Toolchain Registration

1. Compiling and Executing a C program

Điều hướng đến tab trình soạn thảo và chọn loại đầu vào cho C (Lưu ý: sẽ xảy ra lỗi nếu chưa thiết lập trình biên dịch C hợp lệ). Một chương trình ví dụ có thể là:

Ảnh có chứa văn bản

Mô tả được tạo tự động

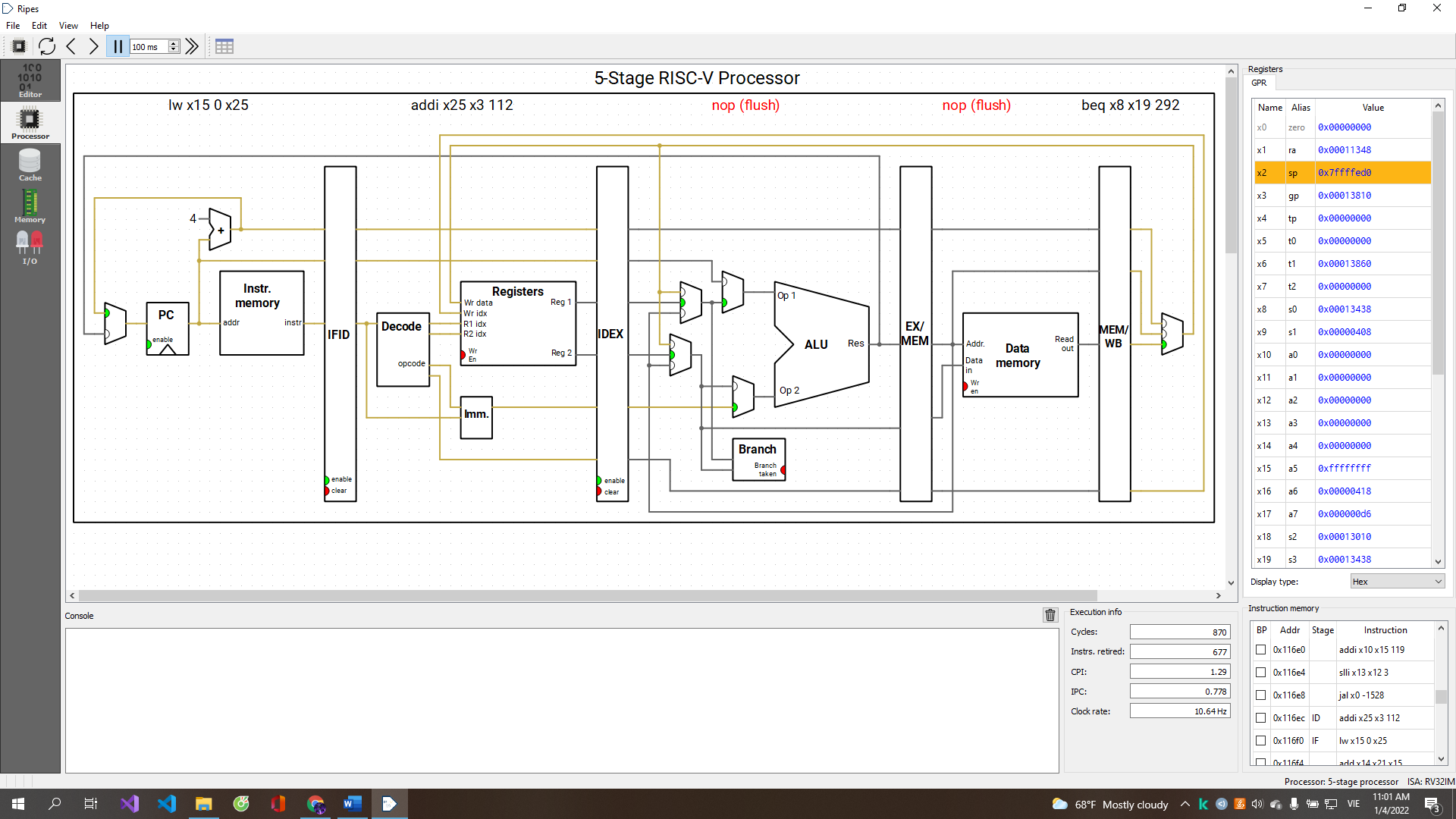
Hình 9: Chương trình C

Nếu không tìm thấy lỗi cú pháp nào và chương trình được tạo thành công, tệp thực thi được tạo sẽ tự động được tải vào trình mô phỏng và hiển thị trong dạng xem đã tháo rời ở bên phải trong màn hình trình chỉnh sửa. Tiếp theo, chúng tôi có thể mô phỏng chương trình như bình thường, bằng cách chạy hoặc bước qua chương trình.

Ảnh có chứa bàn

Mô tả được tạo tự động

Hình 10: Mô phỏng, thực thi chương trình



Hình 11: Mô phỏng đường dữ liệu

Hình 10: Mô phỏng đường dữ liệu

Ảnh có chứa bàn

Mô tả được tạo tự động

Hình 12: Mô phỏng bộ dữ liệu đệm

Ảnh có chứa văn bản, máy tính, máy tính xách tay, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 13: Memory

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 14: I/O

CHƯƠNG 4: KẾT LUẬN

Sau khi tìm hiểu và được sự hướng dẫn tận tình của cô, em đã hoàn thành bài tập lớn “Mô phỏng đường dữ liệ bộ xử lý RISC-V cho chương trình C bằng Ripes”. Sau khi hoàn thành đề tài này em đã hiều hơn về bộ xử lý RISC-V, đường dữ liệu,…

Tuy nhiên do thời gian cho phép cũng kiến thức còn hạn chế, bài tập này còn nhiều thiếu sót. Một lần nữa chúng em xin chân thành cảm ơn cô Tạ Thị Kim Huệ đã nhiệt tình hướng dẫn, truyền đạt kiến thức trong suốt quá trình học tập và thực hiện đề tài này.

# TÀI LIỆU THAM KHẢO

[1]. David A. Patterson & John L. Hennessy. Computer Organization and Design: The Hardware/ Software Interface, Sixth Edition.

[2]. <https://github.com/mortbopet/Ripes/wiki/Building-and-Executing-C-programs-with-Ripes>.

[3]. <https://github.com/mortbopet/Ripes/wiki/Ripes-Introduction>.

Link source code: <https://github.com/ThanhLam-57/KTMT>