## ***Tập lệnh RISC-V***

**1. Phần mở rộng số học dấu chấm động (Floating Point Extensions - F và D)**

[GitHub - openhwgroup/cv32e40p: CV32E40P is an in-order 4-stage RISC-V RV32IMFCXpulp CPU based on RI5CY from PULP-Platform](https://github.com/openhwgroup/cv32e40p)

* **Rocket Chip (SiFive)**: Rocket Chip hỗ trợ các phần mở rộng số học dấu chấm động đơn (F) và kép (D). Rocket Chip có thể cấu hình để bao gồm hoặc loại bỏ phần mở rộng này.
  + GitHub: [Rocket Chip](https://github.com/chipsalliance/rocket-chip)
* **BOOM (Berkeley Out-of-Order Machine)**: BOOM hỗ trợ phần mở rộng F và D trong các lõi out-of-order, thích hợp cho các ứng dụng hiệu suất cao.
  + GitHub: [BOOM](https://github.com/riscv-boom/riscv-boom)

**2. Phần mở rộng Vector (V - Vector Instructions)**

[GitHub - pulp-platform/ara: The PULP Ara is a 64-bit Vector Unit, compatible with the RISC-V Vector Extension Version 1.0, working as a coprocessor to CORE-V's CVA6 core](https://github.com/pulp-platform/ara)

* **RISC-V Vector Extension (riscv-v-spec)**: Đây là dự án mã nguồn mở của đặc tả phần mở rộng vector cho RISC-V, được phát triển và duy trì bởi nhóm RISC-V Vector.
  + GitHub: [riscv-v-spec](https://github.com/riscv/riscv-v-spec)
* **Ara (ETH Zurich)**: Ara là một dự án mã nguồn mở của ETH Zurich, phát triển một lõi xử lý vector 64-bit RISC-V có thể kết hợp với các bộ xử lý khác. Ara hỗ trợ đầy đủ phần mở rộng V và được tích hợp trong nền tảng PULP.
  + GitHub: [Ara](https://github.com/pulp-platform/ara)

**3. Phần mở rộng Bit Manipulation (B - Bit-Manipulation Instructions)**

[GitHub - openhwgroup/cv32e40p: CV32E40P is an in-order 4-stage RISC-V RV32IMFCXpulp CPU based on RI5CY from PULP-Platform](https://github.com/openhwgroup/cv32e40p)

* **Bit Manipulation Extension (riscv-bitmanip)**: Đây là dự án mã nguồn mở của đặc tả phần mở rộng Bit Manipulation, bao gồm các lệnh như dịch và xoay bit, thao tác bitwise.
  + GitHub: [riscv-bitmanip](https://github.com/riscv/riscv-bitmanip)
* **PicoRV32**: PicoRV32 là lõi RISC-V đơn giản, hiệu quả tài nguyên, hỗ trợ các tập lệnh bit manipulation. Dự án này đặc biệt hữu ích cho các ứng dụng nhúng cần tiết kiệm diện tích và năng lượng.
  + GitHub: [PicoRV32](https://github.com/cliffordwolf/picorv32)

**4. Phần mở rộng Atomic (A - Atomic Instructions)**

* **Rocket Chip**: Rocket Chip hỗ trợ các lệnh nguyên tử (A), cho phép xử lý đa luồng và đa lõi. Phần mở rộng A rất quan trọng trong việc đồng bộ hóa dữ liệu.
  + GitHub: [Rocket Chip](https://github.com/chipsalliance/rocket-chip)
* **BOOM**: BOOM hỗ trợ phần mở rộng A, giúp thực hiện các phép toán nguyên tử trong môi trường đa luồng.
  + GitHub: [BOOM](https://github.com/riscv-boom/riscv-boom)

**5. Phần mở rộng Cryptography (K - Cryptography Instructions)**

* **RISC-V Cryptography Extensions (riscv-crypto)**: Đây là dự án mã nguồn mở của đặc tả phần mở rộng mã hóa cho RISC-V, bao gồm các lệnh AES và SHA.
  + GitHub: [riscv-crypto](https://github.com/riscv/riscv-crypto)
* **SHA3 Hardware Accelerator**: Dự án này là một bộ gia tốc SHA3 mã nguồn mở được phát triển trên RISC-V. Bộ gia tốc này phù hợp cho các ứng dụng bảo mật yêu cầu xử lý hash tốc độ cao.
  + GitHub: [SHA3 Accelerator](https://github.com/nesl/riscv-sha3)

**6. Phần mở rộng Transactional Memory (T - Transactional Memory Instructions)**

* **Transactional Memory Model for RISC-V (research project)**: Một số dự án nghiên cứu mã nguồn mở cung cấp mô hình bộ nhớ giao dịch cho RISC-V, bao gồm các lệnh TXSTART, TXCOMMIT, TXABORT. Dự án này đang trong giai đoạn thử nghiệm và phát triển.
  + GitHub: [Transactional Memory RISC-V](https://github.com/cambridge-transactional-memory)

**7. Phần mở rộng Compressed (C - Compressed Instructions)**

* **Rocket Chip**: Rocket Chip hỗ trợ phần mở rộng C, giúp giảm kích thước mã và tối ưu hóa cho các ứng dụng nhúng.
  + GitHub: [Rocket Chip](https://github.com/chipsalliance/rocket-chip)
* **PicoRV32**: PicoRV32 hỗ trợ các lệnh nén C, rất phù hợp cho các ứng dụng nhỏ, yêu cầu tiết kiệm tài nguyên.
  + GitHub: [PicoRV32](https://github.com/cliffordwolf/picorv32)

**8. Các nền tảng FPGA hỗ trợ RISC-V**

* **LiteX**: LiteX là một nền tảng SoC mã nguồn mở cho FPGA, hỗ trợ nhiều lõi RISC-V với các phần mở rộng như A, F, D, C, và một số phần mở rộng khác. LiteX cho phép người dùng xây dựng các hệ thống tùy chỉnh và kiểm thử trên FPGA.
  + GitHub: [LiteX](https://github.com/enjoy-digital/litex)
* **VexRiscv**: VexRiscv là một lõi CPU RISC-V mã nguồn mở được thiết kế để sử dụng với FPGA. Lõi này có thể cấu hình để hỗ trợ nhiều phần mở rộng, bao gồm C, M, A, F.
  + GitHub: [VexRiscv](https://github.com/SpinalHDL/VexRiscv)

## ***Mã nguồn mở RISC-V hỗ trợ AI, DeepLearning***

Dưới đây là một số dự án mã nguồn mở RISC-V hỗ trợ xử lý **AI** và **Deep Learning**. Các dự án này bao gồm các nền tảng SoC, các bộ xử lý vector, các gia tốc phần cứng và các thư viện hỗ trợ thực hiện các tác vụ AI trên RISC-V.

**1. PULP Platform (Parallel Ultra Low Power)**

* **Mô tả**: PULP là nền tảng RISC-V mã nguồn mở phát triển bởi ETH Zurich và University of Bologna, tối ưu cho hiệu năng và tiêu thụ điện năng thấp, đồng thời hỗ trợ khả năng mở rộng đa lõi cho các tác vụ AI nhẹ như xử lý tín hiệu, nhận dạng mẫu, và mạng nơ-ron tích chập (CNN) với mức năng lượng thấp.
* **Tính năng**: Bao gồm các lõi như **Hero** (lõi xử lý song song hỗ trợ AI cơ bản) và **GAP8** (lõi tiết kiệm năng lượng cho AI nhúng).
* **Ứng dụng**: Nhúng, IoT, thiết bị di động.
* **GitHub**: [PULP Platform](https://github.com/pulp-platform)

**2. GAP8 by GreenWaves Technologies**

* **Mô tả**: GAP8 là vi điều khiển RISC-V hỗ trợ các tác vụ AI nhúng, tối ưu hóa cho xử lý đa phương tiện và các tác vụ học sâu (Deep Learning) tiêu thụ ít năng lượng. GAP8 dựa trên kiến trúc PULP và được tối ưu để xử lý các mạng nơ-ron tích chập.
* **Tính năng**: Hỗ trợ các mô hình CNN cơ bản, mạng hồi quy (RNN), và các tác vụ nhận dạng âm thanh và hình ảnh.
* **Ứng dụng**: Camera AI, thiết bị IoT, nhận dạng mẫu thời gian thực.
* **GitHub**: [GAP SDK](https://github.com/GreenWaves-Technologies/gap_sdk)

**3. SHL (Stream-based Hierarchical Learner)**

* **Mô tả**: SHL là một bộ gia tốc phần cứng cho RISC-V hỗ trợ các ứng dụng học sâu, tập trung vào các tác vụ yêu cầu luồng dữ liệu lớn như CNNs và RNNs.
* **Tính năng**: Hỗ trợ các tác vụ học sâu như nhận diện hình ảnh, xử lý âm thanh, và phân loại dữ liệu với kiến trúc đa lõi tối ưu cho hiệu suất.
* **Ứng dụng**: Nhúng, camera thông minh, cảm biến thông minh.
* **GitHub**: [SHL Accelerator](https://github.com/AI-NanoLab/SHL)

**4. VTA (Versatile Tensor Accelerator)**

* **Mô tả**: VTA là một bộ gia tốc tensor mã nguồn mở tích hợp với TVM và hỗ trợ RISC-V, nhằm cung cấp khả năng gia tốc cho các tác vụ AI như mạng nơ-ron sâu và CNNs.
* **Tính năng**: Được tối ưu hóa để thực hiện các toán tử tensor lớn với hiệu suất cao, hỗ trợ mạng nơ-ron học sâu và các tác vụ tensor khác.
* **Ứng dụng**: Thiết bị AI nhúng, các hệ thống học sâu tích hợp.
* **GitHub**: [VTA](https://github.com/apache/tvm-vta)

**5. Dory (Deployment Oriented Runtimes for Inference)**

* **Mô tả**: Dory là một thư viện mã nguồn mở giúp triển khai các mô hình học sâu trên các lõi RISC-V trong nền tảng PULP. Dory hỗ trợ việc tối ưu hóa và triển khai các mô hình inference, tối ưu cho RISC-V với cấu trúc đơn giản.
* **Tính năng**: Hỗ trợ triển khai các mô hình nén và tối ưu hóa mô hình CNNs, RNNs trên thiết bị tiêu thụ năng lượng thấp.
* **Ứng dụng**: Thiết bị IoT, cảm biến thông minh, AI nhúng.
* **GitHub**: [Dory](https://github.com/pulp-platform/dory)

**6. RVV (RISC-V Vector Extensions)**

* **Mô tả**: RVV là phần mở rộng vector cho RISC-V, cho phép xử lý các phép toán vector và tensor với hiệu suất cao, phù hợp cho các tác vụ AI như Deep Learning.
* **Tính năng**: Hỗ trợ tính toán trên vector, lý tưởng cho các mạng nơ-ron tích chập và các tác vụ yêu cầu tính toán matrix lớn.
* **Ứng dụng**: AI nhúng, các tác vụ học sâu yêu cầu tính toán vector.
* **GitHub**: [riscv-v-spec](https://github.com/riscv/riscv-v-spec)

**7. RISC-V Deep Learning Accelerator (RISCVDLA)**

* **Mô tả**: RISCVDLA là một bộ gia tốc cho các tác vụ học sâu, được thiết kế dành riêng cho kiến trúc RISC-V. Bộ gia tốc này có thể chạy các mô hình Deep Learning và các mô hình mạng nơ-ron với hiệu suất cao.
* **Tính năng**: Hỗ trợ các phép tính tensor, tối ưu cho các mô hình CNNs và DNNs.
* **Ứng dụng**: Ứng dụng AI nhúng và các hệ thống Deep Learning nhúng.
* **GitHub**: [RISCVDLA](https://github.com/riscvdla)

**8. NVDLA (NVIDIA Deep Learning Accelerator) trên RISC-V**

* **Mô tả**: NVIDIA Deep Learning Accelerator là bộ gia tốc mã nguồn mở của NVIDIA dành cho các mô hình học sâu, có thể tích hợp với các lõi RISC-V.
* **Tính năng**: Hỗ trợ các mô hình học sâu tiêu chuẩn, như CNN, RNN và các mô hình AI yêu cầu xử lý tensor.
* **Ứng dụng**: Máy tính nhúng, các hệ thống Deep Learning tiêu thụ ít năng lượng.
* **GitHub**: [NVDLA](https://github.com/nvdla/hw)

**9. TVM (AI Compiler) với Hỗ trợ RISC-V**

* **Mô tả**: TVM là một trình biên dịch AI mã nguồn mở hỗ trợ RISC-V, tối ưu hóa mô hình học sâu cho nhiều nền tảng nhúng. TVM tích hợp với VTA để chạy các tác vụ Deep Learning hiệu quả trên các thiết bị RISC-V.
* **Tính năng**: Tối ưu hóa mô hình học sâu, giúp giảm độ trễ và tiêu thụ điện năng cho các tác vụ inference trên RISC-V.
* **Ứng dụng**: Thiết bị AI nhúng, các nền tảng học sâu tiêu thụ ít năng lượng.
* **GitHub**: [TVM](https://github.com/apache/tvm)