**RISC-V**

RISC-V es un ISA (Instruction Set Architecture) disponible para el libre desarrollo del mismo (open source), que nació en 2010 en la universidad de California en Berkeley con un propósito esencialmente educativo, y ha estado creciendo en popularidad gracias entre otras cosas a la flexibilidad que aporta por la gran variedad de extensiones que se pueden montar sobre la misma.

**Implementaciones Actuales**

Como esta arquitectura se construye a partir de las múltiples extensiones de la misma, se vuelve bastante flexible en sus posibles aplicaciones, desde sistemas empotrados como microcontroladores hasta supercomputadores y servidores. Para dar un ejemplo de las posibles implementaciones dejo una lista con las más recientes y significativas, tratando de dar la mejor visualización de su utilidad actual.

**ESP32-C61:** Este procesador le da conexión inalámbrica (conexión wifi 6), bluetooth a dispositivos empotrados donde tiene un único core RISC-V microcontrolador. Este dispositivo salió al mercado el 8 enero del 2024 marcando un claro ejemplo de la implementación de RISC-V para microcontroladores.

Referencia: https://www.espressif.com/en/products/socs/esp32-c61

**SiFive Performance P870-D:** Este procesador es de la marca SiFive, compatible con RVA23, perfil de implementación de RISC-V, este es un procesador planteado para centros de datos donde se necesita una alta cantidad de paralelización enfocándose en la eficiencia y tamaño, y no necesariamente en la potencia máxima de cómputo. Su salida está planteada para finales de 2024.

Referencia: <https://www.sifive.com/cores/performance-p870d>

**NV-RISCV:** NVIDIA se ha vuelto un gran impulsor de RISC-V estos últimos años, desde el 2015 aproximadamente NVIDIA ha ido sustituyendo los controladores “falcom” por cores RISC-V desarrollando a lo largo de los años hasta 20 extensiones especializadas sobre este estándar, y ahora se estima que cada chip incluye entre 10 y 40 cores RISC-V.

Referencia: <https://www.tomshardware.com/pc-components/gpus/nvidia-to-ship-a-billion-of-risc-v-cores-in-2024>

**Menciones adicionales:**

- Alibaba con T-Head XuanTie C910 y la CPU Sophon SG2042

- Western Digital's con su procesador SweRV Core

**Opciones Base**

A pesar de que RISC-V este montado sobre varias extensiones, este comparte algunas partes independientemente de las extensiones que se monten. Para empezar, esta arquitectura es una arquitectura load-store donde los accesos a memoria se solo hacen desde estas dos instrucciones y el resto de las operaciones se hace directamente en CPU, como por ejemplo operaciones aritméticas y lógicas. La longitud general de las instrucciones es de 32 bits, aunque soporta otros tipos de longitudes en sus extensiones, esto siempre que estén alineadas a media palabra (16 bits). Esto es decisión de diseño, de manera que las instrucciones quedan divididas en parcelas de media palabra. Para dar soporte a los dos tipos posibles de alineaciones (big-endian y little-endian) las parcelas se almacenan en memoria con alineación little-endian independientemente de la alineación del procesador. En la figura 1 podemos ver un formato para diferenciar la longitud de una instrucción de antemano.

Figura 1

Tabla

Descripción generada automáticamente con confianza baja

Continuando con el diseño general de esta arquitectura tenemos un total de 31 registros para propósito general (x1-x31), donde contamos con un registro x0 que estará puesto siempre a 0 independientemente de lo que se cargue en él (forzado por hardware). La longitud de estos registros depende de la versión base: para RV32 estos tienen una longitud de 32 bits mientras que para RV64 tienen una longitud de 64 bits. Si comparamos la versión de 32 bits nos encontramos con una situación muy parecida a la del procesador mc88110. Luego nos encontramos con 32 registros de 64 bits para coma flotante (f0-f31) que de forma general deberían tener soporte para simple y doble precisión, sin embargo, hay extensiones que soportan la media precisión y la cuádruple precisión, como se podrá ver en el próximo apartado. Para registros de carácter específico tenemos el PC (program counter) junto al registro FSR (floating point status register) que es un registro para el modo de operación, redondeo, y el estado de excepciones de la unidad de coma flotante. De forma adicional tenemos una serie de formatos de instrucción base como se puede ver en la figura 2.

Figura 2

Escala de tiempo

Descripción generada automáticamente

**Extensiones**

A partir de este punto voy a introducir algunas de las posibles extensiones junto con las instrucciones y partes que me han parecido curiosas o diferentes de las instrucciones del procesador motorola 88110, procesador RISC.

Empezado primero en que hay dos extensiones base sobre las que se monta el resto, siendo estas RV32 y RV64 que se diferencian sobre todo en la longitud de los registros de propósito general, y en el soporte que dan sobre distintas longitudes de direcciones de memoria, hasta 32 bits para RV32 y 64 bits para RV64. También está la versión RV128 para direcciones de memoria de 128 bits. Esta, al igual que RV64, se monta sobre su versión anterior, en este caso RV64, para este habría que incluir instrucciones para el acceso y calculo con direcciones y registros de 128 bits, sin embargo, como esta extensión es todavía muy prematura en su implementación en este documento me voy a centrar en las extensiones RV64 y RV32.

**RV32I**

Esta es la extensión base de instrucciones de enteros de 32 bits, en este caso se da soporte a direcciones de 32 bits asumiendo como se ha mencionado antes registros de 32 bits. Tenemos en este caso instrucciones básicas de operaciones con números enteros como: operaciones lógicas (AND, OR, XOR), operaciones aritméticas (ADD, SUB), de desplazamientos (SLL, SRL, SRA) junto con operaciones de carga (LW, SW, LBU, LB, SB, LH, LHU …) e instrucciones de control, tanto saltos incondicionales (JAL, JALR), como saltos condicionales (BEQ, BNE, BLT, BGE) también se cuenta con instrucciones de ordenación de memoria (FENCE) y llamadas al sistema (ECALL, EBREAK). Como punto a resaltar es que los datos inmediatos en la mayoría de las instrucciones son de 12 bits de signo extendido, por supuesto hay excepciones como es el caso de JAL donde el offset es de 20 bits. También se han encontrado algunas partes que me han parecido curiosas en la decisión de diseño, para empezar en los accesos a memoria; en las instrucciones de carga y almacenamiento la dirección de destino se calcula a partir del valor de un registro con un offset inmediato de 12 bits con signo extendido, a diferencia del procesador mc88110 donde para el cálculo de la dirección destino el offset se podía encontrar en un registro. La mayoría de las operaciones tanto lógicas como aritméticas cuentan con una versión que soporta datos inmediatos, sin embargo, para el caso de la operación resta (SUB) no se tiene una versión que permita restar a un registro datos inmediatos, al igual que no se cuentan con versiones ni para la suma un para la resta para operar con datos sin signo. También para el caso de los saltos condicionales a diferencia del procesador mc88110 no hay que hacer una llamada previa a una instrucción de comparación, sino que la comparación se realiza de manera interna en la propia operación.

**Continuar…….**