El diseño modular y extensible de RISC-V

Julieta Bloise y Juan Pablo Destefanis

Resumen—Este documento describe cómo el diseño modular y extensible de RISC-V resulta útil para los fabricantes de procesadores y de *cores*. Se la compara con otras ISAs y se explica el objetivo de su diseño.

Index Terms—RISC-V, Conjunto de Instrucciones, x86, ARM, Modularidad.

INTRODUCCIÓN

En la actualidad, los conjuntos de instrucciones (ISA, por sus siglas en Inglés) x86 y ARM dominan el mercado comercial con enfoques distintos. Mientras x86, sigue el modelo CISC, y es ampliamente utilizada en computadoras personales y servidores. ARM, es basada en el modelo RISC, se encuentra principalmente en dispositivos móviles debido a la eficiencia en el consumo de energía de sus implementaciones. A pesar de su éxito, estas ISAs pueden resultar complejas y abrumadoras en ciertas situaciones. Por un lado, x86 cuenta con un enorme número de instrucciones. ARM no tiene este problema. Pero sigue manteniendo como obligatorias algunas instrucciones a pesar de que no se necesiten en todos los dispositivos, como las de multiplicación de enteros [5]. Es en este aspecto en el que RISC-V destaca por su enfoque modular y extensible. Al ser relativamente nuevo, pudo aprender de los errores del pasado, convirtiéndose en una ISA con el potencial de adaptarse a una gran variedad de dispositivos creados para diferentes aplicaciones.

El objetivo de este documento es explicar qué implica que RISC-V sea modular y extensible, y por qué se decidió que sea de esta forma. Para tal fin, se organiza de la siguiente manera:

- La *sección I* es una introducción a la ISA. Trata sobre sus características principales y su funcionamiento.
- La *sección II* detalla la importancia del diseño modular de RISC-V para el futuro propio y de otras ISAs. Se lo contrasta con conjuntos de instrucciones incrementales, y se muestra las ventajas que presenta frente a ellos.
- La *sección III* trata sobre la posibilidad de crear nuevas extensiones para agregar funcionalidad a la ISA.

1. RISC-V

RISC-V es un conjunto de instrucciones de propósito general, *load-store*, que está ganando popularidad en la industria por ser abierta y libre de regalías. Esto significa que cualquiera puede usar la ISA en sus diseños, sin tener que preocuparse por incurrir en costos o en tener que adquirir licencias para su uso.

El objetivo de la ISA es crear un conjunto de instrucciones "universal". Por ello, se optó por que sea modular y extensible, lo que permite a los diseñadores de hardware personalizar el conjunto de instrucciones para satisfacer sus necesidades específicas.

1.1. RISC vs CISC

Desde su concepción, RISC-V fue diseñada como una ISA RISC. Una de las motivaciones para esta elección fue el deseo de que la ISA sea aceptada dentro de la comunidad *open-source*, con lo cual querían basar su elección en un historial de éxito comercial ya probado. Cuando RISC-V se estaba desarrollando, ya hacia tiempo que ARM dominaba el mercado de dispositivos móviles, siendo una ISA RISC. Con lo cual ya había buenos ejemplos que habían optado por ese mismo camino[1].

Por otra parte, considerando los estilos por sí mismos, los autores también creían que había pocas razones para usar instrucciones complejas, y que de hecho habían muchas desventajas. Algunas de ellas son:

- *Implementaciones irracionales*: La complejidad de las instrucciones es frecuentemente justificada con la posibilidad de crear programas más pequeños, por lo específico de las instrucciones. Sin embargo, que los programas sean más pequeños no implica siempre que se corra más rápido. Se han dado varios ejemplos de esto. Uno podría ser el encontrado por Peuto y Shustek, que descubrieron que en la IBM 370, una secuencia de instrucciones de *load* era más rápida que una única instrucción de *load* multiple si se usaban menos de 4 registros [13]. Con lo cual, en ese caso (que cabe destacar, cubre el 40 % de los casos de uso del *load* multiple en programas típicos [2]), usar la instrucción más compleja terminaba siendo contraproducente.
- *Mayor cantidad de errores de diseño*: La gran cantidad de instrucciones hace que el proceso de *debugging* de los diseños sea más extenso y complicado.

Por el contrario, consideraron que había varias ventajas para el uso de RISC:

- *Factibilidad de las implementaciones*: Siempre es más factible realizar implementaciones con ISAs simples, porque es más fácil que entren en un solo chip.

- *Tiempo de diseño*: Implementar un conjunto de instrucciones simple requiere menos tiempo que uno de mayor complejidad.
- *Velocidad*: La simplicidad de las implementaciones de RISC, resulta en menos componentes lógicos en los procesadores, contribuyendo así a ciclos de procesamiento más cortos. En los conjuntos de instrucciones CISC, esto significa que las instrucciones "extras" deberían lograr mejoras similares en este aspecto, pero en la práctica hay poca evidencia de que esto pase (de hecho, hay evidencia de lo contrario [2]).
- *Mejor uso del área de los chips*: Las ISAs RISC ocupan menos espacio en los chips. Y aún en casos en los que se dispone del área para implementar un conjunto de instrucciones CISC, se podría usar mejor el espacio extra para *on-chip* cachés, transistores más grandes y mejores, o incluso *pipelining*[2].

1.2. Sets base y extensiones estándar de RISC-V

A pesar de que es conveniente hablar de una única ISA, RISC-V es en realidad una familia de ISAs, en la cual hay cuatro ISAs base. Hay dos principales: RV32I y RV64I, que proveen espacios de 32 y 64 bits, respectivamente. Las dos restantes son RV32E y RV128I. La primera es un subset de RV32I para 16 registros, pensada para microcontroladores pequeños; y la segunda es una variante futura para un espacio de direcciones de 128 bits.

Para poder obtener más funcionalidades en estas ISAs, existen extensiones opcionales que agregan soporte para, por ejemplo, multiplicación y división de enteros, operaciones atómicas, y aritmética de coma flotante. En el Cuadro 1 se pueden ver sus diferentes denominaciones.

Se planea que tanto las extensiones como las bases se mantengan constantes en el tiempo. De agregarse nuevas instrucciones, se hará en forma de extensiones opcionales que no afectarán a las ya existentes[3].

Los diseñadores pueden adoptar solamente las instrucciones que necesiten en sus implementaciones, evitando así complejidad innecesaria. Esto no es posible, por ejemplo, con x86, que por razones de compatibilidad o legado, incluye un espectro más amplio de funcionalidades por defecto, independientemente de la aplicación final. Se hablará de esto en detalle en secciones posteriores.

1.3. Registros en RV32I

RV32I dispone de 32 registros de 32 bits de longitud. El registro x0 tiene todos sus bits permanentemente en 0. Los registros x1-x31 mantienen valores que se interpretan de distinta forma dependiendo de la instrucción. Existe un registro extra que es el *Program Counter (PC)*, que mantiene la dirección de la instrucción actual. [3]

1.4. Longitud de las instrucciones

La ISA base tiene instrucciones de una longitud fija de 32 bits, que deben estar alineada a 32 bits. Sin embargo, RISC-V está diseñado para contemplar extensiones de la ISA con instrucciones de longitud variable, en la que cada instrucción puede tener una longitud de 16 bits o multiplos de 16. Con lo cual, algunas extensiones pueden permitir que las instrucciones estén a lineadas a 16 bits. Sin embargo, ninguna extensión puede hacer que las instrucciones estén alineadas a algún otro valor que no sean 16 o 32 [3].

Cuadro 1
Set de instrucciones base y extensiones de RISC-V. [4]

ISA Base	Instrucciones	Descripción
RV32I	47	Instrucciones de enteros y espacio de direcciones de 32 bits
RV32E	47	Subset de RV32I, restringido a 16 registros
RV64I	59	Instrucciones de enteros y espacio de direcciones de 64 bits, junto a varias instrucciones de enteros de 32 bits.
RV128I	71	Instrucciones de enteros y espacio de direcciones de 128 bits, junto a varias instrucciones de enteros de 32 bits y 64 bits.
Extensión	Instrucciones	Descripción
M	8	División y multiplicación de enteros
A	11	Operaciones atómicas de memoria, load-reserve/store-conditional
F	26	Operaciones de coma flotante de precisión simple
D	26	Operaciones de coma flotante de precisión doble; requiere la extensión F
Q	26	Operaciones de coma flotante de precisión cuádruple; requiere las extensiones F y D
C	46	Instrucciones comprimidas de enteros

1.5. Formato de las instrucciones en RV32I

Existen seis formatos de instrucciones básicos:

Tipo-R, para operaciones entre registros.
Tipo-I, para inmediatos cortos y loads.
Tipo-S, para *stores*.
Tipo-B, para saltos condicionales.
Tipo-U, para inmediatos largos.
Tipo-J, para saltos incondicionales.

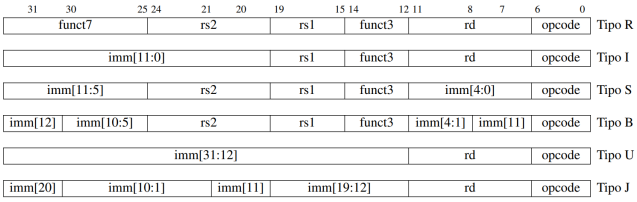

Figura 1. Formato de las instrucciones de RV32I con sus diferentes campos [5]

Como se puede observar en la Figura 1, el *encoding* de las instrucciones es muy regular. Todos los formatos están alineados a 32 bits, y el *opcode* y los operandos están en posiciones fijas.

1.6. Memoria

RISC-V tiene un espacio de direcciones de memoria de 2^{XLEN} bytes, siendo *XLEN* el ancho de un registro en bits (32 o 64, dependiendo de la base). Una palabra (*word*) equivale a 32 bits, una media palabra (*halfword*) a 16 bits, una

doble palabra (*doubleword*) a 64 bits y una cuádruple palabra (*quadword*) a 128 bits. Además, su espacio de direcciones es circular, lo que significa que el último byte en la máxima dirección de memoria es adyacente al byte en la posición cero.

RISC-V puede trabajar con memorias que usen el *little endian* o *big endian*. Las instrucciones se almacenan en memoria como secuencias de "parcelas" little endian de 16 bits, sin importar el *endianness* de la memoria. Si hay varias parcelas que forman una única instrucción, se guardan en direcciones sucesivas de *halfwords*, con la parcela de menor dirección siendo la que contiene los bits más pequeños en la especificación de la instrucción[3].

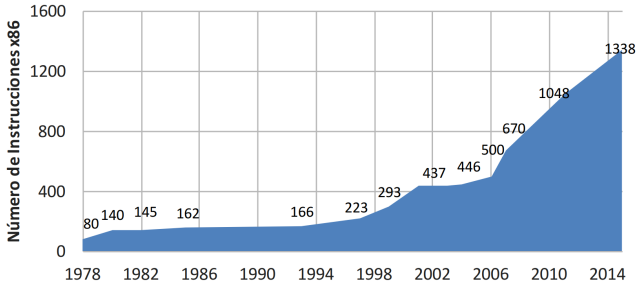

Figura 2. Crecimiento del set de instrucciones x86 a través de su existencia [5]

2. El impacto de la modularidad de RISC-V en el futuro

Considerando que las ISAs pueden durar un largo período de tiempo en uso, es importante analizar el panorama futuro de la tecnología para ver qué se necesita que tengan los nuevos conjuntos de instrucciones. Es muy posible que ese futuro se vea dominado por tres plataformas: **IoTs**, **dispositivos móviles personales** y **clusters de computadoras** [1]. Ante un panorama tan variado, lograr diseñar una ISA que le sirva a todos los dispositivos de la misma forma es desafiante, puesto que no todos los dispositivos necesitan de las mismas instrucciones. Por ejemplo, a un cluster de computadoras quizás le resultaría muy provechoso disponer de instrucciones para operaciones de coma flotante de precisión cuádruple, pero es muy improbable que lo mismo le pase a un *SoC* o a un dispositivo móvil. Con lo cual, una ISA que apunta a ser "universal" necesita poder adaptarse a las necesidades variables de los dispositivos. Por ello, se optó por que RISC-V contase con las siguientes características:

1. Un núcleo pequeño de instrucciones esenciales, del que puedan depender los sistemas operativos y los compiladores.
2. Extensiones opcionales que permitan agregar instrucciones que no son absolutamente necesarias en todos los dispositivos.
3. Espacio para opcodes nuevos, que permitan a los diseñadores agregar instrucciones que permitan sacar mayor provecho a sus dispostivos.

De esta forma, RISC-V adopta la modularidad y la extensibilidad como características claves de su diseño, cumpliendo así con su objetivo de adaptarse a las necesidades de los usuarios.

2.1. ISAs modulares vs ISAs incrementales

En la actualidad, el mercado de las PCs se encuentra dominado por x86 [1]. Esta ISA apareció en el año 1978 [7], y por ello tiene un enfoque más convencional, opuesto al de RISC-V: es una **ISA incremental**. La diferencia con RISC-V radica en que cada implementación de x86, por esta decisión de diseño, debe permitir el uso de todas las instrucciones de versiones anteriores. A lo largo de su historia, se estima que se han agregado aproximadamente tres instrucciones por mes a medida que se iban sacando nuevas versiones [5].

El objetivo de las ISAs incrementales es conservar la compatibilidad entre dispositivos, asegurando que los programas compilados de versiones anteriores sigan funcionando en procesadores más modernos. No obstante, en el proceso aumenta de forma significativa la cantidad de instrucciones, como queda claro en el ejemplo de la ISA de Intel. Esto impacta directamente en la complejidad de las mismas, que pueden terminar contando con miles de instrucciones, haciendo que sea prácticamente imposible que alguien las pueda conocer en su totalidad. Podemos usar de referencia la cantidad de páginas de los manuales de x86-32 y RISC-V para dejar esto en evidencia: el primero tiene 2198 páginas [6], y el segundo tan solo 236 [3], [5].

Por otro lado, las **ISAs modulares**, adoptan un enfoque distinto. Presentan una base inalterable de instrucciones mínimas que debe estar en cualquier implementación. La modularidad viene en forma de extensiones opcionales que el hardware puede incorporar de acuerdo a las necesidades de cada aplicación [5]. El hecho de que las extensiones sean opcionales hace que no se conviertan en un requisito para futuras implementaciones, lo que brinda una mayor flexibilidad y adaptabilidad. Este enfoque permite una evolución más eficiente y personalizada del conjunto de instrucciones, evitando la acumulación de componentes redundantes o anticuados, y contribuyendo así a implementaciones más eficientes y optimizadas.

Considerando que las ISAs cambian muy lentamente con el tiempo [1], las decisiones de diseño son críticas y las afectan significativamente en el largo plazo. Por eso la modularidad de RISC-V la hace una alternativa más atractiva a otros conjuntos de instrucciones, que no estuvieron planteados desde un principio para poder crecer sin arrastrar errores del pasado o mantener instrucciones obsoletas.

Las tendencias del mercado parecen indicar una predilección por ISAs más simples. Un claro ejemplo es que los dispositivos móviles no se encuentran dominados por x86, sino por ARM, que tiene un diseño RISC [1]. Incluso, algunas compañías están empezando a apostar por ARM para PCs [14]. En este contexto, RISC-V parece poder sumarse a esta tendencia perfectamente. No solo es RISC, lo que ya de por sí le permite tener pocas instrucciones. Sino que además, su modularidad le permite a sus implementaciones usar un conjunto elemental muy reducido de instrucciones que se limite a lo que se necesite solamente.

3. Extensiones Personalizadas de RISC-V y su impacto en la industria

Las extensiones estándar de RISC-V suelen cubrir la mayoría de las necesidades, pero existen situaciones donde se requieren soluciones más específicas. En estos casos, la ISA destaca por su capacidad para implementar instrucciones personalizadas. Esta flexibilidad permite a los diseñadores crear soluciones a medida, manteniendo la compatibilidad con la especificación principal de RISC-V, lo que asegura la ejecución de software estándar para estos procesadores.

Un claro ejemplo de esta capacidad se observa en Seagate, que desarrolló dos núcleos de CPU RISC-V personalizados para responder a necesidades específicas en almacenamiento computacional [15]. Un núcleo está optimizado para alto rendimiento en el control de discos duros, mientras que el otro es más compacto y está enfocado en tareas secundarias. Estos núcleos ilustran la manera en que el diseño modular de RISC-V posibilita la adaptación de la tecnología a requerimientos concretos, superando las limitaciones de las extensiones estándar.

Similarmente, la microarquitectura de RV-MLPU (RISC-V Machine Learning Processing Unit), pudo aprovechar también la adaptabilidad de la ISA [18]. RV-MLPU es una unidad pensada para realizar operaciones de *machine learning* en dispositivos móviles. Para ella, se uso RISC-V como ISA y se hizo uno extensión propia derivada de la extensión de vectores estándar. Esta decisión les permitió seleccionar solo un subconjunto de instrucciones, que eran las necesarias para su objetivo.

La relevancia de la modularidad de RISC-V también se destacó en el Primer Taller Internacional sobre RISC-V para *HPC* (High-Performance Computing), un evento que congregó a expertos y aficionados del sector. Aquí, la arquitectura abierta y modular de RISC-V se mostró como un facilitador clave para la especialización y adaptación a las demandas del área. Se debatió sobre la optimización de RISC-V para supercomputadoras y el co-diseño de hardware/software en *HPC*, abriendo nuevas vías para mejorar la velocidad, fidelidad y calidad de actividades de I+D. La capacidad de RISC-V para soportar vectorización a través de extensiones como RVV (RISC-V Vector Extension) es vital para aplicaciones de *HPC*, destacando su relevancia en el desarrollo de soluciones personalizadas para tareas de alto rendimiento computacional [16].

Estos avances reflejan la continua evolución y adaptabilidad de RISC-V en una variedad de aplicaciones de alto rendimiento.

En paralelo, la modularidad de RISC-V es crucial en el desarrollo de tecnologías avanzadas como los PolarFire SoC FPGAs de Microchip, porque facilita la integración de componentes clave en estos sistemas. En los PolarFire SoC FPGAs, se traduce en una mejor capacidad para integrar CPUs RISC-V multinúcleo, controladores de memoria y transceptores de alta velocidad, lo que simplifica el diseño de sistemas complejos. Los diseñadores tienen la flexibilidad de elegir configuraciones de potencia y rendimiento adecuadas para sus aplicaciones específicas, optimizando así el consumo de energía y el rendimiento. Además, la posibilidad de actualizar y reemplazar componentes individuales sin rediseñar todo el sistema subraya la sostenibilidad y adaptabilidad a largo plazo de los diseños basados en RISC-V. Esta modularidad no solo es fundamental en teoría y en aplicaciones de HPC, sino también en prácticas de hardware, como en los PolarFire SoC FPGAs, donde facilita el diseño, adaptación y actualización de sistemas complejos [17].

Además, la adopción y aplicación práctica de RISC-V por parte de empresas líderes en tecnología como Qualcomm, demuestra su versatilidad y potencial en el mercado actual. Qualcomm ha explorado activamente el uso de la ISA [19], resaltando cómo su flexibilidad y escalabilidad se alinean con sus objetivos de innovación y eficiencia en la tecnología de procesadores. Este enfoque subraya la relevancia contemporánea de RISC-V y su capacidad para adaptarse a las necesidades cambiantes de la industria.

3.1. Estado actual del desarrollo de las extensiones de RISC-V

Actualmente, varias extensiones de RISC-V están en desarrollo, según se detalla en el sitio web de la Fundación RISC-V [12]. Estas extensiones proponen nuevas instrucciones para diversos usos, como la implementación de lenguajes interpretados en RISC-V [8], o la operación con el formato *bfloat16* [10], diseñado por Google para procesadores destinados a trabajar con redes neuronales [11]. También se mantiene un registro de extensiones aceptadas y parte de la especificación oficial, incluyendo aquellas relacionadas con la manipulación de bits y la criptografía escalar y vectorial [9].

A diferencia de las ISAs privativas, el proceso de ratificación de estas extensiones por parte de la Fundación RISC-V es abierto y participativo. Esto representa una ventaja significativa, ya que permite la contribución de una amplia gama de expertos, asegurando que las extensiones sean lo más efectivas posible. Este proceso transparente garantiza que las decisiones sean justificadas públicamente y accesibles a todos los interesados.

4. Conclusión

Al ser abierto y libre de regalías, RISC-V se muestra como una alternativa accesible e innovadora frente a las ISAs dominantes del momento. Su existencia podría significar que los procesadores sean más accesibles para más dispositivos, impulsando la expansión del IoTs y de los SoCs. Sin embargo, el objetivo de sus creadores es más grande, y pretenden convertirlo en el estándar para todos los dispositivos computacionales [1]. Ante esta meta, la modularidad y la extensibilidad se presentan como decisiones de diseño destacables, y características fundamentales de la ISA, abriéndole las puertas a una gran cantidad de aplicaciones de las más diversas.

La aplicación práctica de estas características en proyectos específicos, como los de Seagate y los PolarFire SoC FPGAs de Microchip, confirma su relevancia en el presente. Además, el interés de Qualcomm en RISC-V refleja su capacidad para adaptarse a diversos desafíos y necesidades futuras, fortaleciendo la posición de RISC-V como una arquitectura de procesadores flexible y con gran potencial de adaptación.

En el contexto actual, en el que las ISAs más simples están ganando terreno y los conjuntos de instrucciones CISC son cada vez menos frecuentes, la modularidad y extensibilidad de RISC-V parecen una evolución natural. Es una cualidad que le va a permitir seguir mejorando en el futuro de forma constante sin problemas. De hecho, como se vio en la sección III, ya está pasando. Esta facilidad con la que se puede trabajar en varias extensiones al mismo tiempo, sumado a la naturaleza libre y abierta de RISC-V, la hacen una ISA con mucho potencial de crecimiento y un futuro prometedor.

REFERENCIAS

[1] Krste Asanović y David Patterson. "Instruction Sets Should Be Free: The Case For RISC-V". Universidad de California. Berkeley. 6 de Agosto, 2014.

[2] David Patterson y David Ditzel. "The Case for the Reduced Instruction Set Architecture". SIGARCH Computer Architecture News 8.6, Páginas 25 a 33. 1980.

[3] *The RISC-V Instruction Set Manual Volume I: Unprivileged ISA*, de Andrew Waterman, Krste Asanović. 13 de Diciembre, 2019.

[4] David Kanter. "RISC-V Offers Simple, Modular ISA", Microprocessor Report. 28 de Marzo, 2016.

[5] *The RISC-V Reader: An Open Architecture Atlas*, de David Patterson y Andrew Waterman.

[6] *Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 2: Instruction Set Reference*. Intel Corporation. Septiembre, 2016.

[7] Benj Edwards. "Birth of a Standard: The Intel 8086 Microprocessor." PCWorld. 17 de Junio, 2008.

[8] RISC-V J Extension. https://github.com/riscv/riscv-j-extension

[9] RISC-V Cryptography Extension. https://github.com/riscv/riscv-crypto

[10] RISC-V bfloat16 Specification. https://github.com/riscv/riscv-bfloat16

[11] Shibo Wang y Pankaj Kanwar. "BFloat16: The secret to high performance on Cloud TPUs". Google. 23 de Agosto, 2019.

[12] RISC-V Specification Status. https://wiki.riscv.org/display/HOME/Specification+Status

[13] Bernard Peuto y Leonard Shustek. "An instruction timing model of CPU performance". Cuarto Simposio de Arquitecturas de Computadora. Marzo, 1977.

[14] Katie Tarasov. "How Arm is gaining chip dominance with its architecture in Apple, Nvidia, AMD, Amazon, Qualcomm and more". CNBC. 9 de Noviembre, 2023.

[15] The Register. (2020, December 8). Seagate RISC-V Computational Storage. https://www.theregister.com/2020/12/08/seagate_risc_v_computational_storage/.

[16] EPCC at The University of Edinburgh. "First International Workshop on RISC-V for HPC". EPCC RISC-V Community. http://riscv.epcc.ed.ac.uk/community/isc23-workshop

[17] Microchip Technology Inc. "PolarFire SoC FPGAs". Microchip Official Website. https://www.microchip.com/en-us/products/fpgas-and-plds/system-on-chip-fpgas/polarfire-soc-fpgas

[18] Zahra Azad, Marcia Sahaya Louis, Leila Delshadtehrani, Anthony Ducimo, Suyog Gupta, Pete Warden, Vijay Janapa Reddi, y Ajay Joshi. "An end-to-end RISC-V solution for ML on the edge using in-pipeline support". Boston University, Google Inc., Harvard University. 2020.

[19] Qualcomm, "What is RISC-V, and why we're unlocking its potential". Qualcomm OnQ Blog. 8 de Septiembre 2023. https://www.qualcomm.com/news/onq/2023/09/what-is-risc-v-and-why-were-unlocking-its-potential