Arquitectura RISC-V: un enfoque revolucionario para sistemas computacionales

Carranza, Giordano Hoyo, Oriz

Abstract—Este informe aborda el cambio en la innovación de la arquitectura de procesadores, enfocándose en los procesadores específicos de dominio en lugar de los procesadores de propósito general. Se analiza la arquitectura RISC-V como un conjunto de instrucciones de especial utilidad en esta situación y con un futuro prometedor. Se examinan los fundamentos de la arquitectura RISC, su diseño basado en instrucciones reducidas, y los principales rasgos distintivos de RISC-V, con énfasis en características relevantes de su implementación y extensiones estándar importantes. Además, se realiza una comparación entre las arquitecturas RISC-V, x86 y ARM, concluyendo con una comparación en cuanto al formato de sus instrucciones. Se destaca el objetivo de este informe de resaltar las ventajas y aplicaciones de RISC-V, incluyendo su exitosa integración de aceleradores de IA. En la conclusión, se resalta el sólido ecosistema de desarrollo de RISC-V y su potencial para impulsar la innovación en el diseño de procesadores.

- *Abril Giordano Hoyo - Estudiante, Licenciatura en Análisis de Sistemas - Facultad de Ingeniería UBA - agiordano@fi.uba.ar*
- *Lihuen Carranza -Estudiante, Licenciatura en Análisis de Sistemas - Facultad de Ingeniería UBA - lcarranza@fi.uba.ar*
- *Omar Oriz - Estudiante, Licenciatura en Análisis de Sistemas - Facultad de Ingeniería UBA - ooriz@fi.uba.ar*

1 Introduccion

En vista del debilitamiento de la Ley de Moore, se ha observado un cambio en la innovación de la arquitectura de los procesadores, impulsando un enfoque hacia los procesadores específicos de dominio en lugar de los procesadores de propósito general. Esta tendencia permite optimizar la arquitectura del procesador para aplicaciones específicas, mejorando así su eficiencia operativa [1].

En el presente informe, nos enfocaremos en caracterizar y analizar un conjunto de instrucciones de especial utilidad ante esta situación y que tiene un futuro prometedor: RISC-V.

2 Arquitectura Risc V

2.1 Introduccion

La arquitectura RISC (*Reduced Instruction Set Computer*) se define como una arquitectura de procesadores diseñada para ejecutar un número reducido de instrucciones simples. A diferencia de la arquitectura CISC (*Complex Instruction Set Computer*), RISC se caracteriza por su enfoque en instrucciones sencillas [2].

La filosofía de diseño de RISC tiene como principal objetivo ofrecer un conjunto limitado y reducido de instrucciones simples pero potentes. Esta característica simplifica el diseño y hace que el procesador sea más rápido y eficiente [3].

Sin embargo, no afecta únicamente a la velocidad, sino también a la previsibilidad. El tiempo consumido en ejecutar completamente una instrucción en una arquitectura de tipo CISC es mucho más variable dada la diversa naturaleza de todas las diferentes instrucciones [4].

RISC-V es una ISA de tipo RISC que fue lanzada en 2010 por un equipo liderado por el profesor Krste Asanovic en la Universidad de California, Berkeley. Originalmente diseñada para respaldar la investigación y educación en ciencias de la computación, RISC-V tenía como visión convertirse en una arquitectura estándar de código abierto para su implementación en la industria.

A pesar de la existencia de otras arquitecturas populares en el mercado, RISC-V surgió como respuesta a dos factores principales. En primer lugar, la falta de disponibilidad de ISAs abiertas, lo que implicaba la necesidad de adquirir licencias para su uso. En segundo lugar, la complejidad inherente de las instrucciones y las dificultades asociadas con su implementación en hardware debido a su especificidad [5].

2.2 Características

Una de las características más destacadas y novedosas de RISC-V es que es tipo abierta y libre, con lo cual cualquier persona puede diseñar un procesador que cumpla con esta ISA sin tener que pagar licencias por ello. Esto fomenta la proliferación de diferentes implementaciones de procesadores, cada una con sus propias características distintivas. Además, promueve la competencia, la innovación y ha despertado un gran interés por parte de numerosas empresas que ahora se orientan al mercado de hardware basado en RISC-V.

La base de esta arquitectura se limita a un conjunto mínimo de instrucciones suficientes para brindar una base sólida para compiladores, ensambladores, enlazadores y sistemas operativos. Se denomina "ISA de enteros base" debido a que su conjunto básico de instrucciones se centra en operaciones matemáticas y lógicas que involucran números enteros [6].

Los desarrolladores de RISC-V se enfocan en la simplicidad y la eficiencia del hardware con el objetivo de minimizar la complejidad en el diseño. Por esta razón,

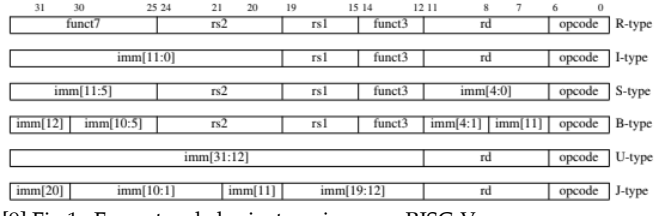
RISC-V es una arquitectura extremadamente modular. Esto permite la adición de extensiones según las necesidades específicas de cada proyecto, lo que brinda una mayor personalización de la arquitectura para adaptarse a diferentes requisitos. Esta modularidad evita la inclusión de instrucciones innecesarias que podrían afectar negativamente la eficiencia y la velocidad en ciertos escenarios [7][8].

RISC-V ofrece tres variantes principales de la "base entera": RV32I, RV32E y RV64I. La diferencia entre RV32E y RV32I radica en la cantidad de registros que poseen. RV32I posee treinta y un registros de propósito general y uno adicional, el “x0”, que siempre tiene el valor cero. En cambio, RV32E posee sólo dieciséis ya que está orientada a sistemas embebidos. Por otro lado, la diferencia entre RV32I y RV64I se encuentra en los tamaños de las direcciones, siendo de 32-bits y 64-bits respectivamente.

RISC-V opera exclusivamente con registros, lo que lo caracteriza como una arquitectura del tipo *load-store*. Las instrucciones siempre se almacenan en una secuencia de paquetes en formato *little-endian*, independientemente de la endianness del sistema de memoria, lo que evita posibles incongruencias.

RV32I consta de cuarenta y ocho instrucciones de las cuales 8 son instrucciones de sistema. Por lo tanto, existen 40 instrucciones netas para el usuario, que se pueden clasificar en tres tipos: instrucciones de cómputo, de control de flujo y de acceso a memoria.[5]

En cuanto a los formatos de instrucciones, existen seis tipos diferentes: de tipo-R para operaciones entre registros, de tipo-I para inmediatos cortos y *loads*, de tipo-S para *stores*, de tipo-B para *branches*, de tipo-U para inmediatos largos y de tipo-J para saltos incondicionales. Además, los bits de los registros a ser leídos y escritos van en la misma posición para todas las instrucciones, de manera tal que sea posible comenzar a acceder a dichos registros incluso antes de la decodificación [9].



[9] Fig 1 : Formatos de las instrucciones en RISC-V.

2.3 Extensiones

Existen dos tipos de extensiones. Por un lado se encuentran las estándar, diseñadas para ser ampliamente útiles y con la característica de que no entran en conflicto con otras extensiones de este tipo. Por otro lado, aquellas denominadas no estándar pueden ser altamente especializadas o pueden generar interferencias con otras extensiones, ya sean estas estándar o no. A continuación, detallaremos las principales extensiones estándar que posee RISC-V.

La extensión “I” abarca instrucciones computacionales para enteros, incluyendo operaciones de carga, almacenamiento y control de flujo. Es importante destacar que esta extensión es obligatoria para todas las implementaciones de RISC-V.

La extensión de multiplicación y división conocida como "M", incorpora instrucciones diseñadas para realizar operaciones de multiplicación y división tanto para enteros signados como no signados.

Para realizar operaciones con números de punto flotante de precisión simple o doble, existen las extensiones “F” y “D” respectivamente, las cuales usan registros separados llamados “f”. Las operaciones de precisión simple únicamente usan los 32 bits menos significativos de los registros de precisión doble.

La extensión estándar de instrucciones atómicas en RISC-V, denotada por "A", introduce un conjunto de instrucciones que permiten la lectura, modificación y escritura atómica de la memoria, brindando mecanismos de sincronización entre procesadores. Tiene dos tipos de operaciones atómicas para coordinación: *Atomic Memory Operations* y *load reserved & store conditional*. Esta diferencia surge del hecho de que existen dos casos de uso muy distintos.

Load reserved y store conditional proveen una operación atómica, *compare-and-swap*, a través de dos instrucciones. De esta manera, es posible realizar esta operación con instrucciones que posean únicamente dos registros fuente. Se decidió no implementar una única instrucción para esta operación ya que pasar de dos a tres operandos de origen complicaría la interfaz de memoria del sistema, el control y datapath de enteros, y el formato de instrucciones.

El motivo fundamental de tener instrucciones AMO es que escalan mejor en grandes sistemas de multiprocesamiento, y son útiles para comunicarse con dispositivos de I/O, dado que ejecutan una lectura y escritura en una sola transacción de bus atómica.

Otra extensión relevante es la llamada “C”, de instrucciones comprimidas. En este ámbito, RISC-V innovó al establecer que cada instrucción comprimida a 16-bits deba mapearse a una sola instrucción estándar de 32-bits. Esta tarea la realiza un decodificador antes de que se ejecuten las instrucciones.

La extensión de vectores, conocida como “V”, ofrece capacidades de procesamiento paralelo eficiente al incluir y aprovechar registros vectoriales y operaciones especializadas. A diferencia de las conocidas instrucciones SIMD, una forma más elegante de trabajar con paralelismo a nivel de datos es la arquitectura vectorizada. En este caso, el tamaño de los registros es determinado por la implementación, en lugar de establecerse por medio del opcode como ocurría en SIMD.

Esta extensión agrega treinta y dos registros vectorizados, cuyos nombres comienzan con “v”. Una característica importante es que el número de elementos por registro varía dependiendo del ancho de las operaciones y de la cantidad de memoria dedicada a estos registros vectorizados. Cada registro vectorial posee un tamaño en bits fijo determinado por la implementación, el cual debe ser una potencia de dos.

Si hablamos de saltos condicionales, las arquitecturas vectorizadas incluyen una máscara que suprime operaciones en algunos elementos del vector, escribiendo a cada elemento de la máscara un uno si la condición se cumple o un cero en caso contrario. Para esta tarea, RISC-V provee ocho registros predicados vectoriales. [9]

Las extensiones vistas hasta el momento forman parte del grupo de extensiones no privilegiadas. Existen a su vez otras de carácter privilegiado de las cuales hablaremos a continuación.

En la arquitectura RISC-V, los procesadores tienen diferentes niveles de privilegio que determinan qué tipo de operaciones pueden realizar y qué partes del sistema pueden acceder. Las instrucciones ya comentadas están disponibles en el “modo usuario”. Este está destinado para aplicaciones o programas de usuario convencionales, tiene menos privilegios y solo puede acceder a recursos y servicios autorizados.

Por otro lado, existen otros dos niveles de privilegio: el “modo supervisor” y el “modo máquina”. El “modo supervisor” está diseñado para el sistema operativo del dispositivo. En este caso, el código tiene más privilegios que en el modo usuario y puede acceder y gestionar recursos del sistema, como memoria y dispositivos de entrada/salida. En cambio, el “modo máquina” es el más alto y el único nivel de privilegio obligatorio para todos los dispositivos RISC-V. Se utiliza principalmente para gestionar entornos de ejecución seguros en RISC-V, lo que permite establecer configuraciones de seguridad y proteger áreas críticas del sistema [7].

Una extensión privilegiada de gran relevancia que resulta importante mencionar es la de Hipervisor, conocida como "H". Esta extensión de hipervisor transforma el “modo supervisor” en el modo “supervisor extendido por hipervisor”, permitiendo que un hipervisor o un sistema operativo aloje y administre máquinas virtuales. Adicionalmente, la extensión agrega una nueva etapa de traducción de direcciones que posibilita que las direcciones físicas utilizadas por los sistemas operativos huésped sean traducidas a direcciones físicas en el nivel de supervisor. Esta característica es fundamental para virtualizar la memoria y los subsistemas de entrada/salida de los sistemas operativos huésped.

En términos de funcionalidad, el modo "H" es similar al modo supervisor, aunque incorpora instrucciones y registros de control adicionales para gestionar estas máquinas virtuales en un “modo supervisor virtual”. En este modo, los sistemas operativos huésped operan como si estuvieran ejecutándose directamente en un hardware real, aunque en realidad son controlados y gestionados por el hipervisor [7].

3 Procesadores Tradicionales

Con el fin de proporcionar un análisis completo sobre RISC-V, examinaremos dos arquitecturas que actualmente dominan el mercado: x86 y ARM.

En primer lugar, es necesario destacar que tanto RISC-V como ARM se basan en RISC, a diferencia de x86 que se trata de una arquitectura de tipo CISC.

Si analizamos más en profundidad, existe una amplia diferencia en cuanto a la cantidad de instrucciones que poseen. ARM posee 1070 instrucciones, similar a x86 que tiene a disposición 1300. Sin embargo, RISC-V se diferencia ampliamente en este punto, teniendo menos de 50 instrucciones (47, reducibles a 38), lo que hace que el modelo se simplifique ampliamente. [5]

RISC-V alcanza los treinta y un registros, a diferencia de ARM y x86 que poseen 15 y 8, respectivamente. Esto influye a su vez en los modos de direccionamiento, los cuales son complejos en las dos últimas arquitecturas mencionadas. RISC-V adopta un único modo de direccionamiento, el cual a su vez no discrimina según el tipo de dato, como sí sucede en las otras arquitecturas.

Otra característica que contribuye a la eficiencia de RISC-V es la ausencia de códigos de condición. Estos implícitamente utilizan muchas instrucciones para establecer los *flags*, con lo cual la implementación de instrucciones *compare & branch* característica de esta ISA simplifica dicha operación. De esta manera, RISC-V usa una única instrucción, sin necesidad de que se vea afectado el rendimiento al tener que “setear” estos estados.

RISC-V omite instrucciones de rotación y detección de overflow aritmético, ya que ambas pueden ser calculadas con un par de instrucciones de las ya existentes. A pesar de poder realizar operaciones con la pila mediante el uso de instrucciones y registros generales, RISC-V tampoco incluye instrucciones de stack tales como *push* y *pop*, las cuales sí están incluidas tanto en ARM como en x86.

RISC-V es la única que posee un registro particular en cero. De esta manera, dado a que el mismo se utiliza en una amplia cantidad de instrucciones, se evita una carga innecesaria, mejora la eficiencia y simplifica el hardware.

Tanto x86 como ARM requieren de las instrucciones de multiplicación y división para ejecutar el *software stack*. Esto las diferencia de RISC-V, en donde dichas operaciones forman parte de una extensión opcional.

Entre RISC-V y ARM podemos marcar una diferencia con respecto a los valores inmediatos. A diferencia de la primera arquitectura en donde el inmediato es una constante, los 12 bits del campo de inmediatos en ARM son la entrada de una función que produce el inmediato. Continuando con las diferencias entre estas dos ISAs, ARM necesita que los datos estén alineados en memoria, requisito que no precisa la arquitectura que nos concierne.

Una diferencia importante entre x86 y RISC-V es que este último utiliza regiones de memoria para los accesos a dispositivos de entrada y salida. En vez de tener instrucciones específicas que complejizan el modelo, se aprovechan las instrucciones de carga y almacenamiento convencionales.

Un tema relevante a tratar para analizar las diferencias es el de punto flotante. ARM posee 32 registros de punto flotante de precisión simple, con lo cual al momento de trabajar con números con precisión doble, la cantidad de registros disponibles se reduce a la mitad. x86, en su momento, ni siquiera contaba con registros para este tipo sino que simplemente eran almacenados en el stack.

RISC-V, a diferencia de las otras dos ISAs comentadas, incluye instrucciones para mover datos directamente entre registros enteros y de punto flotante. De no tenerlas, la única opción es acceder a memoria para almacenar el valor y luego cargarlo en un registro. En esto pierden eficacia x86 y ARM, ya que deben realizar un

acceso innecesario a memoria para luego realizar efectivamente la carga en el registro entero. [9]

A fin de visualizar claramente sus diferencias, compararemos los formatos de instrucción propuestos por RISC-V (figura 1) y la extensión de x86, IA-32. En el caso de RISC-V, específicamente en RV32I, todas las instrucciones tienen una longitud fija de 32 bits, independientemente del tipo de instrucción o la cantidad de registros involucrados. En el caso de AI-32, es importante notar que la longitud de las instrucciones puede variar significativamente debido a diversos factores técnicos. Entre ellos se encuentran ciertos prefijos adicionales de un byte que alteran el comportamiento de la instrucción, la necesidad de incluir cuatro bytes para representar valores inmediatos, algunos códigos de operación que ocupan dos bytes o incluso casos en los que se requiere un byte extra debido al uso del modo escalado [24].

[24] Formatos de tres instrucciones diferentes en IA-32, extensión de x86.

4 APLICACIONES

En la actualidad, a pesar de no haber adquirido todavía una gran influencia en el mercado, podemos mencionar ciertos ámbitos donde RISC V obtuvo mayor presencia y reconocimiento. Entre ellos podemos mencionar la Computación de Alto Rendimiento (HPC), la internet de las cosas (IoT), la inteligencia artificial, aprendizaje automático y una amplia gama de sistemas embebidos. [1][10]

4.1 V-Extension

Una de las extensiones más destacadas en el ámbito de RISC-V es la de vectores, ya que ha adquirido una importancia significativa en aplicaciones que implican cálculos intensivos, como el procesamiento de imágenes, la inteligencia artificial y el aprendizaje automático [11]. Brinda una poderosa herramienta para abordar desafíos computacionales complejos al proporcionar una arquitectura versátil y eficiente para el procesamiento en paralelo de datos.

Las instrucciones vectoriales proporcionadas por la extensión de vectores permiten realizar operaciones aritméticas, lógicas y de desplazamiento en los registros vectoriales. Asimismo, se incluyen otras específicas para la carga y almacenamiento eficiente de datos en memoria, así como instrucciones destinadas al manejo y procesamiento de datos vectoriales, y la interacción con la unidad central de procesamiento. Estas instrucciones optimizadas facilitan la manipulación de conjuntos de datos masivos y mejoran el rendimiento en aplicaciones que se benefician del procesamiento paralelo y la aceleración vectorial.

4.2 Inteligencia Artificial y Aprendizaje automático

En el ámbito de la inteligencia artificial y el aprendizaje automático, es fundamental contar con aceleradores de hardware dedicados para realizar cálculos complejos de manera eficiente. Estos aceleradores pueden integrarse en la arquitectura RISC-V mediante interfaces estándar, lo que permite su utilización de manera efectiva en aplicaciones de *IA* y *ML* [10].

Existen modelos, bibliotecas, compiladores y herramientas predefinidas que brindan ayuda al momento de diseñar, entrenar, probar y ejecutar aplicaciones de inteligencia artificial y aprendizaje automático en SOCs basados en RISC-V [10]. Un ejemplo a destacar de estas herramientas es “Renode”. Se trata de un marco de código abierto de “Antmicro” que permite el desarrollo conjunto de hardware y software. Fue diseñado con el principal objetivo de facilitar la prueba y verificación de sistemas embebidos y de sistemas en chip.

Lo interesante de “Renode” es que no se limita a un solo tipo de arquitectura o procesador, por lo que puede manejar múltiples arquitecturas y plataformas. Permite ensamblar SoC virtuales a partir de bloques de construcción RISC-V, en donde también se pueden incluir CPU, GPU, controladores de memoria o interfaces de comunicación [12].

Después de una larga colaboración con Google, “Renode” introdujo la compatibilidad con la extensión de vectores de RISC-V. Esta adición fue significativa para ampliar las capacidades de aprendizaje automático e inteligencia artificial emergentes [13].

4.3 Caso Práctico

Recientemente, SiFive ha utilizado sus procesadores RISC-V para gestionar cargas de trabajo de inteligencia artificial en los centros de datos de Google [14]. El "SiFive Intelligence X280", un diseño multi-núcleo RISC-V con extensiones vectoriales, se combina con las unidades de multiplicación matricial (MXU) de Google para ofrecer una programación más flexible de cargas de trabajo de aprendizaje automático [15].

Con un enfoque específico en la inteligencia artificial y aprendizaje automático, el “X280” presenta una potente combinación: cuenta con un procesador de aplicaciones de 64 bits y una unidad vectorial expandida compatible con el estándar ratificado de vectores de RISC-V, lo que proporciona un procesador óptimo para ejecutar software de computación vectorial. SiFive ofrece las extensiones no estándar "SiFive Intelligence", que añaden instrucciones personalizadas para optimizar las cargas de trabajo de *IA* y *ML*.

El "X280" ejecuta una pila de software RISC-V completa, incluido un hipervisor, y admite hasta 48 bits de espacio de memoria virtual y cachés privados L1 y L2. Tiene 32 registros de 512 bits y permite agrupar hasta ocho registros vectoriales para crear un vector más largo, lo que facilita el procesamiento de vectores de hasta 4096 bits y contribuye a reducir el consumo de energía.

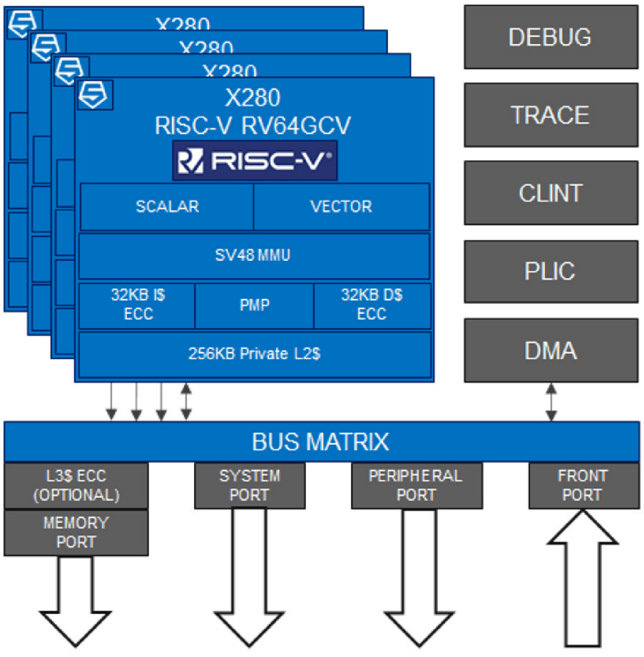
Después de introducir el “X280”, SiFive observó cómo muchos clientes lo utilizaban como un núcleo complementario. Se dieron cuenta de que era posible colocar el núcleo “X280” junto a un gran acelerador,

donde podría proporcionar código de mantenimiento y operaciones, ejecutar kernels que el gran acelerador no podía realizar o proporcionar otras funciones adicionales.

Al colaborar con clientes como Google, SiFive desarrolló la Extensión de Interfaz de Coprocesador Vectorial (VCIX), que permite conectar un acelerador directamente al archivo de registro vectorial del "X280" y comunicarse en un tiempo muy rápido de 10 ciclos de entrada y salida. La VCIX proporciona un mayor rendimiento y un ancho de banda de datos mucho mayor, lo que resulta en una latencia mucho más baja y un software simplificado.

Regresando al caso que nos concierne, VCIX permite el envío y recepción de instrucciones con mayor flexibilidad, lo cual permite una división del trabajo limpia y eficiente entre la unidad MXU de Google y el "X280" [16].

Este caso práctico demuestra las capacidades de adaptabilidad y extensibilidad que posee RISC-V y, en particular, los beneficios de combinar esta arquitectura con aceleradores de IA para mejorar la eficiencia y el rendimiento en aplicaciones de inteligencia artificial [14]. La integración de aceleradores personalizados en el ecosistema RISC-V proporciona flexibilidad y eficiencia en la ejecución de operaciones especializadas.



[18] Esquema de la estructura interna y los componentes del procesador SiFive "x280".

5 Ecosistema

La Fundación RISC-V juega un papel fundamental en el mantenimiento y evolución de la arquitectura. Su objetivo principal es mantener la estabilidad de RISC-V, permitiendo que evolucione de manera segura y cuidadosa. La fundación es responsable de decidir cuándo agregar nuevas opciones al conjunto de instrucciones, y estas decisiones se basan únicamente en razones técnicas sólidas, después de una discusión abierta por un comité de expertos en hardware y software.

Numerosas empresas se han orientado hacia los procesadores basados en RISC-V, diseñando e implementando diversas soluciones. Entre los fabricantes de chips establecidos en el mercado destacan nombres como "Andes Technology," "SiFive," y "Western Digital." Estas empresas no solo desarrollan diversas implementaciones de la arquitectura RISC-V, sino que también ofrecen herramientas y colaboran con otras compañías para alcanzar una mayor eficacia a nivel de rendimiento [17].

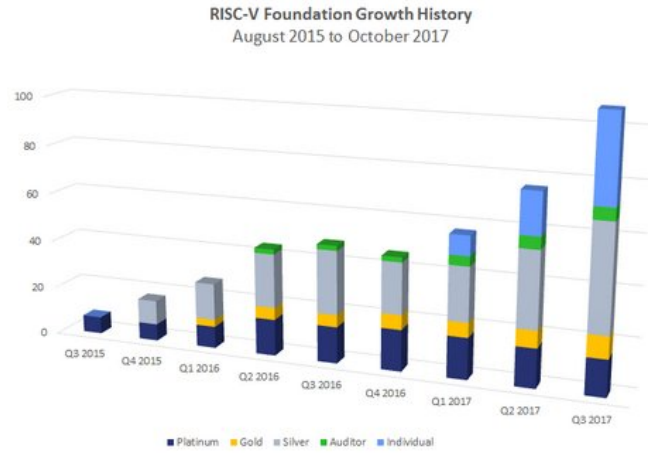
En el ámbito del hardware, el ecosistema de RISC-V continúa expandiéndose, ofreciendo una amplia gama de opciones flexibles y accesibles para diseñar y prototipar soluciones basadas en esta arquitectura. Por ejemplo, "Andes Technology" desarrolló el procesador "AndesCore™ N45", que está orientado al procesamiento de imágenes y videos. En este caso, este núcleo de CPU fue optimizado para tareas de procesamiento gráfico, lo que lo hace especialmente adecuado para esta tarea [19]. Por otro lado, como segundo ejemplo, "SiFive" creó el procesador "Essential™ 7-Series", diseñado para aplicaciones que requieren un alto rendimiento, pero que también enfrentan limitaciones de energía, como puede ser el caso de dispositivos de realidad virtual y aumentada [20].

Además del mencionado proyecto "Renode", el crecimiento y popularidad de la arquitectura RISC-V han dado lugar al desarrollo de diversas herramientas que complementan y amplían su ecosistema. Entre estas herramientas destacamos "RiskFree", un entorno de desarrollo integrado (IDE) basado en RISC-V, desarrollado por la empresa "Ashling", la cual es miembro de la Fundación RISC-V. "RiskFree" ofrece una plataforma unificada para el desarrollo de software dirigido a dispositivos basados en RISC-V, brindando a los desarrolladores una amplia gama de funcionalidades para facilitar el proceso de programación, compilación, depuración y análisis de rendimiento. "Ashling" también desarrolló una sonda de depuración de alta velocidad para dispositivos RISC-V, llamada "Opella-XD". Esta sonda es compatible con las versiones de 32 bits y de 64 bits de la arquitectura RISC-V y permite a los desarrolladores realizar una depuración detallada y eficiente de sus programas, facilitando la identificación y corrección de posibles errores y optimizando el rendimiento del sistema [21].

Asimismo, la disponibilidad de bibliotecas optimizadas para RISC-V permite mejorar el rendimiento y la eficiencia en aplicaciones específicas. En cuanto a este tema podemos mencionar, también de "Andes Technology", la biblioteca "AndesAIRE™ NN". Se trata de una biblioteca de cómputo de redes neuronales poderosa y eficiente que está diseñada para acelerar las tareas de redes neuronales en los procesadores producidos por esta misma empresa, del tipo "AndesCore™ RISC-V" [22].

En resumen, el ecosistema de RISC-V se destaca por su colaboración, crecimiento y compromiso con la innovación y la evolución de la arquitectura. La participación activa de la comunidad, el constante desarrollo de herramientas y bibliotecas optimizadas, así como la diversidad de opciones de hardware disponibles, continúan impulsando la adopción y expansión de

RISC-V en distintos sectores industriales.



[23] Gráfico que expone el crecimiento de la cantidad de miembros adheridos a la fundación RISC-V, en el periodo 2015-2017.

6 Conclusiones

El futuro de RISC-V se vislumbra excepcionalmente prometedor debido a su naturaleza sencilla, abierta y modular, que ha atraído una comunidad activa y colaborativa. Su flexibilidad, eficiencia y personalización hacen de RISC-V una opción atractiva en el campo de la informática. Con un sólido ecosistema de desarrollo respaldando su crecimiento y adopción en diversas industrias, RISC-V continúa impulsando la innovación y eficiencia en el diseño de procesadores y aplicaciones computacionales.

La integración exitosa de aceleradores de inteligencia artificial en procesadores RISC-V, como se evidencia en el caso práctico de SiFive y Google, destaca su capacidad para mejorar el rendimiento en aplicaciones de IA. En consecuencia, RISC-V emerge como una arquitectura revolucionaria con el potencial de transformar la industria y abrir nuevas posibilidades en un futuro cercano.

Referencias

[1] Enfang Cui, Tianzheng Li, Qian Wei, "RISC-V Instruction Set Architecture Extensions: A Survey", IEEE *Access*, vol.11, pp. 24696-24711, February 2023, doi: 10.1109/ACCESS.2023.3246491.

[2] Samuel O. Aletan, "An Overview of RISC Architecture", Louisiana Tech University, Ruston, Louisiana.

[3] Bansal, Malti, y Harsh. "Reduced Instruction Set Computer (RISC): A Survey." J. Phys.: Conf. Ser. 1916 (2021): 012040.

[4] "The RISC Design Philosophy", By Akash M J, https://learneradda.com/1-1-the-risc-design-philosophy/

[5] Andrew Waterman, "Design of the RISC-V Instruction Set Architecture", Technical Report, Electrical Engineering and Computer Sciences, University of California, Berkeley, 2016 (dissertation).

[6] "The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2.2", Editors Andrew Waterman and Krste Asanovi´c, RISC-V Foundation, May 2017.

[7] Andrew Waterman, Krste Asanovic, John Hauser, "The RISC-V Instruction Set Manual, Volume II: Privileged Architecture", University of California, Berkeley, December 4, 2021.

[8] Andrew Waterman, Krste Asanovic, "The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA", University of California, Berkeley, December 13, 2019.

[9] "Guia practica de Risc V", David Patterson, Andrew Waterman (11/07/2018). Recuperado de riscvbook.com/spanish/guia-practica-de-risc-v-1.0.5.pdf

[10] LinkedIn. (s.f.). How do you integrate AI and machine learning capabilities? Recuperado el 18/07/2023, de linkedin.com/advice/3/how-do-you-integrate-ai-machine-learning-capabilities.

[11] Imad Al Assir, Mohamad El Iskandarani, Hadi Rayan Al Sandid, and Mazen A. R. Saghir, "Arrow: A RISC-V Vector Accelerator for Machine Learning Inference", American University of Beirut. Beirut, Lebanon.

[12] Antmicro. (s.f.). Renode: A development framework for software/hardware virtual platforms. Recuperado de https://antmicro.com/platforms/renode/

[13] Antmicro. (2021, diciembre). RISC-V Vector Instructions in Renode. Recuperado de https://antmicro.com/blog/2021/12/riscv-vector-instructions-in-renode/

[14] Asanović, K.,&Young,C. (2019). SiFiveRISC-Vcorespicked forGoogleAIcomputenodes. Recuperado de https://www.theregister.com/2022/09/23/google_using_sifive_riscv_cores/

[15] CloudTensorProcessingUnit(TPU),(2022-09-02). Recuperado dehttps://cloud.google.com/tpu/docs/tpus?hl=es-4

[16] SiFive (21 de septiembre, 2022). "SiFive Intelligence X280 as AI Compute Host: Google Datacenter Case Study". Recuperado de https://www.sifive.com/blog/sifive-intelligence-x280-as-ai-compute-host-google

[17] Busto Pérez de Mendiguren, E. (Febrero 2021). "RISC-V y su ecosistema hardware-software".Facultad de Informática, Universidad Complutense de Madrid.

[18] "SiFive reveals X280 core and AndeSight RISC-V IDE moves to v5.0," LinuxGizmos, 22 de julio de 2023, linuxgizmos.com/sifive-reveals-x280

[19] Andes Technology. (s.f.). RISC-V N45 Overview- AndesCore™ Processors. Recuperado de andestech.com/en/products-solutions/andescore-processors/riscv-n45/

[20] SiFive. (s.f.). Essential™ 7-Series Cores. Recuperado de https://www.sifive.com/cores/essential-7

[21] Ashling. (s.f.). Ashling for RISC-V - Complete RISC-V Ecosystem. Recuperado de ashling.com/ashling-riscv/

[22] Andes Technology. (s.f.). AndesAIre™ NN Library Overview. Recuperado de http://www.andestech.com/en/products-solutions/andesaire-ai/nn-library/

[23] "RISC-V Ecosystem," Design & Reuse, 7 de noviembre del 2017, design-reuse.com/risc-v-ecosystem.html

[24] Nitay Artenstein, "x86 Instruction Encoding Revealed: Bit Twiddling for Fun and Profit" https://www.codeproject.com/Articles/662301/x86-Instruction-Encoding-Revealed-Bit-Twiddling-fo