2 RISC-V

RISC son las siglas de 'Reduced Instruction Set Computer', es decir, un computador con un conjunto de instrucciones reducido. Este concepto surgió en la década de 1970 a raíz de la necesidad de IBM por desarrollar un computador sencillo, pero capaz de ejecutar 12 millones de instrucciones por segundo, de modo que pudiera ser integrado en un sistema de enrutamiento de llamadas telefónicas [5]. Hasta la fecha los computadores incorporaban arquitecturas CISC[1] con las que eran capaces de llevar a cabo procesos complejos por medio de la ejecución de instrucciones igualmente complejas. Con frecuencia, esas instrucciones debían ser interpretadas a nivel arquitectural y traducidas a un microcódigo con el que se ejecutaban las operaciones necesarias para obtener los resultados esperados. No obstante, este enfoque requería contar, en primer lugar, con un intérprete de bajo nivel que tradujera las instrucciones en sus correspondientes microcódigos, además de una memoria dedicada en la que alojarlos.

Las arquitecturas RISC surgen como respuesta a la tendencia de los conjuntos de instrucciones a ganar complejidad con el tiempo, a pesar de que algunos estudios [6] apuntaban que la mayoría de los programas desarrollados por aquel entonces tan solo empleaban un subconjunto de las instrucciones disponibles. En estos estudios se proponía la creación de arquitecturas más sencillas, baratas de producir y eficientes, las cuales, aún siéndolo, fueran capaces de ejecutar programas complejos de igual modo que lo hacían las arquitecturas CISC[2].

2.1. Historia de RISC-V

El manual de la primera versión del ISA abierto RISC-V fue publicado en el año 2011. Por aquel entonces, los investigadores de la Universidad de California en Berkeley, Krste Asanović, Yunsup Lee y Andrew Waterman, bajo la dirección de David A. Patterson, integraban el equipo *ParLab*, trabajando en un proyecto de investigación sobre computación paralela, financiado por Intel y Microsoft, y del cual surgieron como derivados del trabajo de investigación tanto el ISA RISC-V, como el lenguaje de construcción de hardware (HDL) Chisel [7].

Más tarde, en el año 2015, se creó la RISC-V Foundation con el objetivo de organizar los esfuerzos de la comunidad global de desarrolladores de software y hardware,

[1] Siglas de 'Complex Instruction Set Computer'

[2] 'Complex Instruction Set Computer', un computador con un conjunto de instrucciones complejo

por impulsar la adopción del estándar, mantenerlo y hacerlo evolucionar de una manera organizada y eficaz. Además, en este mismo año, los integrantes del *ParLab* (a excepción de Patterson) fundarían SiFive, una compañía dedicada al diseño y venta (que no fabricación) de procesadores basados en RISC-V.

2.2. Especificaciones

La especificación la componen dos ISAs, uno estándar y otro privilegiado, siendo el primero aquel sobre el que se ha puesto el foco a la hora de proponer y desarrollar este trabajo. La especificación del ISA privilegiado recoge las instrucciones y modos de ejecución necesarios para ejecutar sistemas operativos y operar con periféricos [8], mientras que la especificación del ISA no privilegiado recoge las instrucciones necesarias para la construcción de arquitecturas RISC-V básicas pero completamente funcionales.

El ISA no privilegiado lo componen, a su vez, cuatro especificaciones básicas junto con sus respectivas extensiones. Esas especificaciones básicas deben estar presentes en toda implementación que quiera hacerse del ISA, es decir, que una implementación debería integrar una de las cuatro especificaciones base, junto con tantas extensiones como se requiera. Las especificaciones base recogen el conjunto de instrucciones básico que todo núcleo RISC-V debería ser capaz de ejecutar [9], y son las siguientes:

- **RV{32,64}I**: ISAs con un XLEN (longitud de registro) de 32 y 64 bits.
- **RV{32,64}E**: subconjuntos de RV32I y RV64I con la mitad de registros. Elaborados para propiciar la construcción de microcontroladores de tamaño reducido.

Las instrucciones de las especificaciones base posibilitan la ejecución de operaciones aritmético-lógicas (suma, resta, operaciones lógicas, *lui*[3] y *auipc*[4]), saltos condicionales y no condicionales, así como operaciones de lectura y escritura en memoria [10]. En el ISA base se definen, además, las operaciones de ordenación de memoria necesarias para garantizar la consistencia en sistemas con múltiples hilos de ejecución paralelos o (*harts*) [11].

Asimismo, se definen las instrucciones 'ECALL' y 'EBREAK' cuya utilidad es, respectivamente, realizar llamadas al sistema y pausar la ejecución de un *hart* para inspeccionar el estado del banco de registros y la memoria (útil en labores de depuración) [12].

Por último, la especificación indica el formato de otro tipo de instrucciones las cuales, aún siendo válidas, pueden no tener ningún efecto sobre el estado de la arquitectura. Estas instrucciones, denominadas 'HINTs', se reservan para casos en que se desee agregar una cierta funcionalidad al ISA, la cual no tenga por qué ser adoptada obligatoriamente por una arquitectura ya existente, puediendo esta simplemente ignorar la instrucción [13].

[3] *load upper immediate*: carga en un registro un entero de 32 bits construido con los 20 bits más significativos de la instrucción (inmediato), a los cuales se aplica un shift lógico de 12 bits hacia la izda.

[4] *add upper immediate (to) program counter*: construye un inmediato de 32 bits exactamente igual que *auipc*, lo suma al valor del contador de programa y lo guarda en un registro

2.3. El conjunto de instrucciones base RV32I

En el manual de la arquitectura se indica que en el conjunto RV32I las instrucciones deben poder direccionar a 32 registros de propósito general. No obstante, tan solo se restringe el uso del registro 'x0' para albergar una constante de valor 0, quedando recogido el uso del resto de registros en la ABI[5] de RISC-V [14].

En lo que respecta al formato de las instrucciones, existen cuatro formatos base [15]:

- **R**: operaciones aritmético-lógicas en las que los dos operandos son registros.
- **I**: operaciones aritmético-lógicas en las que uno de los operandos es un inmediato codificado en la propia instrucción. Este formato es el empleado, además, en la codificación de la instrucción de lectura de memoria *lw* y sus derivados, así como en la codificación de la instrucción de salto incondicional *jalr*[6].
- **S**: instrucción de escritura en memoria *sw* y sus derivados.
- **U**: es el formato empleado en la codificación de las instrucciones *lui* y *auipc*.

2.3.1. Formatos derivados

Formato B

El formato B es una variación del S y se emplea en la codificación de las instrucciones de salto condicional. La diferencia entre este formato y el formato S reside en la manera de interpretar los campos con que se construye el inmediato.

En las instrucciones de escritura en memoria, la dirección efectiva de acceso se forma concatenando las dos secciones del inmediato (la más significativa y la menos significativa) dentro de la instrucción, extendiendo el signo del resultado a 32 bits, y sumándolo al contenido del registro *rs1*. Por otro lado, en la decodificación de las instrucciones de salto condicional, el offset respecto del contador de programa se construye de una manera menos intuitiva, tal y como puede verse en la Figura 2.1.

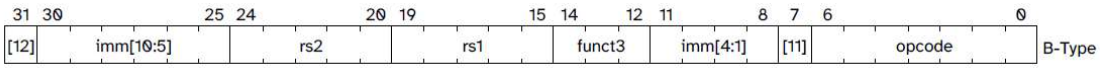

Figura 2.1: Esquema con el formato que deben seguir las instrucciones del tipo B

El offset siempre va a ser un múltiplo de 2, por lo que el último dígito binario del inmediato puede asumirse siempre nulo. Con esto, el inmediato en las instrucciones de

[5] *Application Binary Interface*: una especificación de la forma en que los programas deben ser llamados a nivel de lenguaje máquina (paso de parámetros, retorno de resultados, etc.), así como la forma en que deben construirse y enlazarse los ejecutables que los albergan

[6] *jump and link register*: avanza el contador de programa a una posición igual a su valor actual más el offset formado con la extensión de signo del inmediato a 32 bits. Además, guarda la dirección de retorno (instrucción siguiente al salto) en un registro de destino codificado en la instrucción

salto condicional podría codificar los 12 bits más significativos sigiuentes a ese primer dígito binario, quedando un offset de 13 bits que, posteriormente, habría que extender a 32 bits para sumarlo al contador de programa. En aras de favorecer la reutilización de la lógica de decodificación del inmediato del formato S, se decidió construir el formato B de forma que hubiera tanta superposición como fuera posible con el S, facilitando de este modo la decodificación. En resumidas cuentas, se sacrificó ligeramente la legibilidad del inmediato dentro del formato, favoreciendo la eficiencia en la implementación.

Formato J

El formato J debe emplearse en la codificación de las instrucciones de salto incondicional. Se trata de una variación del tipo U, que sigue la lógica del formato B en cuanto a la construcción del inmediato, tal y como puede observarse en la Figura TODO:: el offset se encuentra codificado en múltiplos de 2 por lo que el último bit va a ser siempre 0, con lo que puede codificarse dentro de la instrucción los 19 bits más significativos inmediatamente posteriores a ese último dígito binario.

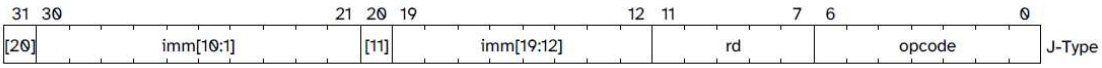

Figura 2.2: Esquema con el formato que deben seguir las instrucciones del tipo B

Bibliografía

[1] Krste Asanović y David A Patterson. «Instruction sets should be free: The case for risc-v». En: *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146* (2014). Consultado el 30 de agosto de 2025. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.pdf.

[2] OpenRISC Community. *Architecture*. Consultado el 28 de mayo de 2025. 2025. URL: https://openrisc.io/architecture.

[3] Tony Chen y David A. Patterson. *RISC-V Geneology*. Inf. téc. UCB/EECS-2016-6. Accedido el 28 de mayo de 2025. University of California at Berkeley, 2016, pág. 2. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-6.pdf.

[4] RISC-V International. *RISC-V Landscape: Members*. Consultado el 28 de mayo de 2025. 2025. URL: https://riscv.landscape2.io/?group=members.

[5] John Cocke y Victoria Markstein. «The evolution of RISC technology at IBM». En: *IBM Journal of research and development* 34.1 (1990). Consultado el 23 de agosto de 2025, págs. 4-11. URL: https://ieeexplore.ieee.org/document/5389855.

[6] Andrew S Tanenbaum. «Implications of structured programming for machine architecture». En: *Communications of the ACM* 21.3 (1978). Consultado el 30 de agosto de 2025, págs. 237-246. URL: https://dl.acm.org/doi/abs/10.1145/359361.359454.

[7] RISC-V International. *About RISC-V International*. Consultado el 30 de agosto de 2025. 2025. URL: https://riscv.org/about/#history.

[8] Andrew Waterman et al. *The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture (User-Level ISA)*. Versión 20250508, Consultado el 31 de agosto de 2025. RISC-V International, Unprivileged ISA Committee. Mayo de 2025, pág. 9. URL: https://drive.google.com/file/d/1uviu1nH-tScFfgrovvFCrj7Omv8tFtkp/view.

[9] Andrew Waterman et al. *The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture (User-Level ISA)*. Versión 20250508, Consultado el 31 de agosto de 2025. RISC-V International, Unprivileged ISA Committee. Mayo de 2025, págs. 15-17. URL: https://drive.google.com/file/d/1uviu1nH-tScFfgrovvFCrj7Omv8tFtkp/view.

[10] Andrew Waterman et al. *The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture (User-Level ISA)*. Versión 20250508, Consultado el 31 de agosto de 2025. RISC-V International, Unprivileged ISA Committee. Mayo de 2025, págs. 27-35. URL: https://drive.google.com/file/d/1uviu1nH-tScFfgrovvFCrj7Omv8tFtkp/view.

[11] Andrew Waterman et al. *The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture (User-Level ISA)*. Versión 20250508, Consultado el 31 de agosto de 2025. RISC-V International, Unprivileged ISA Committee. Mayo de 2025, págs. 35-37. URL: https://drive.google.com/file/d/1uviu1nH-tScFfgrovvFCrj7Omv8tFtkp/view.

[12] Andrew Waterman et al. *The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture (User-Level ISA)*. Versión 20250508, Consultado el 31 de agosto de 2025. RISC-V International, Unprivileged ISA Committee. Mayo de 2025, págs. 37-38. URL: https://drive.google.com/file/d/1uviu1nH-tScFfgrovvFCrj7Omv8tFtkp/view.

[13] Andrew Waterman et al. *The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture (User-Level ISA)*. Versión 20250508, Consultado el 31 de agosto de 2025. RISC-V International, Unprivileged ISA Committee. Mayo de 2025, págs. 38-40. URL: https://drive.google.com/file/d/1uviu1nH-tScFfgrovvFCrj7Omv8tFtkp/view.

[14] Kito Cheng y Jessica Clarke. *RISC-V ABIs Specification*. Ver. August 12, 2025-draft. Consultado el 31 de agosto de 2025. RISC-V International. 2025, pág. 6. URL: https://github.com/riscv-non-isa/riscv-elf-psabi-doc.

[15] Andrew Waterman et al. *The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture (User-Level ISA)*. Versión 20250508, Consultado el 31 de agosto de 2025. RISC-V International, Unprivileged ISA Committee. Mayo de 2025, págs. 25-26. URL: https://drive.google.com/file/d/1uviu1nH-tScFfgrovvFCrj7Omv8tFtkp/view.

[16] Jonathan Bachrach et al. «Chisel: constructing hardware in a scala embedded language». En: *Proceedings of the 49th annual design automation conference*. Consultado el 11 de julio de 2025. 2012, págs. 1216-1225. URL: https://dl.acm.org/doi/abs/10.1145/2228360.2228584.

[17] VirtusLab. *Scala Survey Results 2023*. Consultado el 11 de julio de 2025. 2023. URL: https://scalasurvey2023.virtuslab.com/.

[18] RISC-V International. *riscv-gnu-toolchain*. 2020. URL: https://github.com/riscv-collab/riscv-gnu-toolchain.

[19] «IEEE Standard Hardware Description Language Based on the Verilog(R) Hardware Description Language». En: *IEEE Std 1364-1995* (1996). Consultado el 13 de julio de 2025, págs. 207-218. DOI: 10.1109/IEEESTD.1996.81542.

[20] AMD. *7 Series FPGAs Data Sheet: Overview*. 2020. URL: https://docs.amd.com/v/u/en-US/ds180_7Series_Overview.

[21] Scott Chacon y Ben Straub. *Pro Git*. 2.ª ed. Consultado el 15 de julio de 2025. Berkeley, CA: Apress, 2014. ISBN: 978-1-4842-0076-6. URL: https://git-scm.com/book/en/v2.

[22] Stack Exchange Inc. *2022 Developer Survey*. Consultado el 15 de julio de 2025. 2022. URL: https://survey.stackoverflow.co/2022/#technology-version-control.