UNIVERSIDAD
COMPLUTENSE
MADRID

Proyecto de Innovación

Convocatoria 2022/2023

Nº de proyecto: 201

SUPERSONIC-V: deSarrollo de entornos virtUales Para dEspliegue de haRdware baSadO eN rIsC-V

Responsable del Proyecto:

Alberto Antonio del Barrio García

Facultad de Informática

Departamento de Arquitectura de Computadores y Automática

1. **Objetivos propuestos en la presentación del proyecto**

La docencia tradicional en asignaturas como Arquitectura de Computadores o Estructura de Computadores (Grados en Informática y similares) no incluye prácticas relativas a la implementación de un repertorio de instrucciones en un procesador concreto. Esto es debido a la complejidad de realizar tal implementación. No obstante, con la aparición de RISC-V las cosas han cambiado. RISC-V es un repertorio de instrucciones (Instruction Set Architecture, ISA) abierto y con un espacio para introducir instrucciones nuevas que permitan la especialización de la implementación resultante.

El proyecto RISC-V nació en 2010 en el Parallel Computing Laboratory de la Universidad de California Berkeley, dirigido por el Prof. David Patterson, de la mano del Prof. Krste Asanovic y de los estudiantes Yunsup Lee y Andrew Waterman. En el año 2015 nació la Fundación sin ánimo de lucro RISC-V con el objetivo de construir un ecosistema abierto y colaborativo de desarrollo tanto de hardware como de software basado en el ISA del RISC-V. Más de 40 compañías punteras, como pueden ser AMD, Google, Hewlett Packard Enterprise, IBM, Microsoft, NVIDIA, Oracle, Qualcomm, etc., se han unido a la Fundación RISC-V. También en 2015 apareció SiFive, la primera compañía en fabricar cores basados en RISC-V.

Debido a la naturaleza abierta y customizable actualmente existen 89 cores basados en RISC-V [1], muchos de ellos disponibles públicamente en repositorios tipo github o similar. Por tanto, la ISA RISC-V es un ejemplo real de aplicación de conceptos arquitectónicos que se estudian en asignaturas como Arquitectura, Estructura o Tecnología y Organización de Computadores.

No obstante, en la práctica es complejo ser capaces de replicar las settings de un proyecto RISC-V. Los repositorios donde se encuentran disponibles van siendo actualizados constantemente e incluso por varias personas. Como consecuencia de ello, es fácil encontrar inconsistencias entre las herramientas/versiones a instalar para que el core RISC-V sea simulable y/o sintetizable. Por tanto, los estudiantes dedican una mayor parte del tiempo a pelearse con la instalación que a trabajar con los cores RISC-V para aprovechar su capacidad de especialización mediante nuevas instrucciones.

Por tanto, los objetivos propuestos inicialmente en este proyecto fueron los siguientes:
- Objetivo 1. Facilitar el uso de cores RISC-V en asignaturas relacionadas o trabajos de fin de grado/máster.
- Objetivo 2. Proporcionar entornos virtuales, fácilmente instalables, con las herramientas necesarias para simular y/o sintetizar cores RISC-V.
- Objetivo 3. Proporcionar las herramientas necesarias para implementar programas y compilarlos para ser ejecutados en un core RISC-V.
- Objetivo 4. Facilitar al alumnado las herramientas para poder probar conceptos teóricos de asignaturas relacionadas con la Arquitectura y Tecnología de Computadores.
- Objetivo 5. Escritura de un manual de uso de la máquina virtual, así como de sus herramientas.
- Objetivo 6. Escritura de un artículo de carácter educativo para mostrar los resultados.

2. **Objetivos alcanzados**

En líneas generales se han cumplido los objetivos. Como muestra de ello, a la presente memoria se adjuntan la imagen de una máquina virtual y un docker con los entornos instalados.

En concreto, los Objetivos 1, 2, 3 y 4 se han cumplido al 100%. La máquina virtual desarrollada está basada en el sistema operativo Ubuntu 20.04 Focal Fossa y contiene una instalación de los siguientes elementos:

- *Verilator* versión 4.110. *Verilator* es una herramienta gratuita y de código abierto que permite simular códigos HDL desarrollados en Verilog ciclo a ciclo.
- La toolchain de *gcc* para RISC-V, versión 8.3.0. Gracias a gcc es posible compilar programas escritos en C y generar código objeto para arquitecturas basadas en una determinada arquitectura, en nuestro caso, RISC-V.
- El propio core *CVA6*, inicialmente *Ariane*, es un core RISC-V de 64-bits, con una datapath de 6 etapas y capaz de arrancar Linux.
- La herramienta *riscv-pk*, que es un entorno de ejecución de aplicaciones ligero que puede albergar archivos binarios RISC-V ELF enlazados estáticamente. Este paquete también contiene el Berkeley Boot Loader, *bbl*.

Asimismo, se ha desarrollado una pequeña guía de uso, por lo que podríamos considerar que el Objetivo 5 también se ha cumplido. Faltaría el Objetivo 6, el cual requiere aún tiempo para recabar datos de una población estudiantil significativa.

3. **Metodología empleada en el proyecto**

La metodología seguida en el proyecto se compone de 10 tareas (T0-T9). Para indicar quién está involucrado en cada una de ellas, usaremos las siguientes abreviaturas:

AB: Alberto A. del Barrio, GB: Guillermo Botella, LP: Luis Piñuel, CR: Carlos Roa, RM: Raúl Murillo, DM: David Mallasén

Las tareas han sido las siguientes:

- T0. Reuniones de coordinación mensuales (TODOS).
- T1. Estudio de los repositorios de referencia de cores RISC-V, como los del OpenHW Group o los de Berkeley Architecture Research. De esta manera podremos ver los requerimientos mínimos para utilizar las herramientas necesarias (DM, RM, LP, CR).
- T2. Creación de una máquina virtual basada en una distribución Linux de referencia, como Debian o Ubuntu LTS sobre un motor open-source como VirtualBox (AB, GB, CR).
- T3. Instalación de las herramientas de compilación cruzada de RISC-V (DM, RM AB).
- T4. Instalación de simuladores open source como Verilator (LP, GB, CR).
- T5. Creación de un docker que incluya las herramientas propuestas en las tareas T3 y T4. De esta forma se aligerará el tamaño de la imagen.
- T6. Realización de una prueba de concepto simulada con algún core RISC-V de referencia (RM, AB, GB).
- T7. Despliegue de un core RISC-V de referencia para validar la simulación (DM, LP, AB).
- T8. Escritura del manual de uso de la máquina virtual, docker y de las herramientas instaladas (TODOS).
- T9. Escritura de un artículo docente relativo al proyecto realizado (TODOS).

Con el objetivo de alcanzar todos los hitos del proyecto, la tarea T0 se ha realizado periódicamente para coordinarnos mensualmente.

Como entregables se han desarrollado una máquina virtual y un docker, más ligero, con la instalación de las herramientas necesarias para trabajar con CVA6, uno de los principales cores RISC-V de la literatura actual, y mantenido por el OpenHW Group.

4. **Recursos humanos**

El equipo que ha desarrollado el proyecto es interfacultativo (Informática y CC Físicas) y cuenta con PDI, PAS y estudiantes de doctorado, en concreto:
- Alberto A. del Barrio (PTU, Facultad de Informática).
- Guillermo Botella (PTU, Facultad de Informática).
- Luis Piñuel (PTU, Facultad de Ciencias Físicas).
- Carlos Roa (PAS, Facultad de Informática).
- Raúl Murillo (estudiante de Doctorado en la Facultad de Informática).
- David Mallasén (estudiante de Doctorado en la Facultad de Informática).

5. **Desarrollo de las actividades**

En el presente trabajo se han desarrollado dos entornos virtuales para permitir trabajar con el core CVA6, mantenido por el OpenHW Group y anteriormente conocido como Ariane [2], desarrollado por el grupo de Luca Benini en ETH Zurich.

CVA6 es un core de 64-bits capaz de arrancar Linux, lo cual le dota de gran atractivo. Implementa la ISA RV64GC, es decir, es un core de 64-bits con soporte para las extensiones I (instrucciones enteras), M (multiplicación y división), A (instrucciones atómicas), F (punto flotante de precisión simple), D (punto flotante de precisión doble) y C (instrucciones comprimidas. Es un core de 6 etapas, in-order, por lo que tiene un consumo energético bastante moderado. Además, CVA6 es el core base para la integración del formato numérico posit, proyecto en el que está involucrado el personal del proyecto [3-9].

Por un lado se ha desarrollado una máquina virtual cuyo sistema operativo es Ubuntu 20.04 LTS Focal Fossa. Dicha distribución es LTS, por lo que tiene soporte hasta 2030 [10]. Funciona con VirtualBox, motor open-source de virtualización.

Lo primero de todo será importar la máquina virtual (fichero .ova proporcionado con el proyecto) con el motor de virtualización que estemos utilizando. Posteriormente, arrancamos la máquina y hacemos login. Los datos para el login en esta máquina virtual serían los siguientes:

User: user
Password: riscv1234

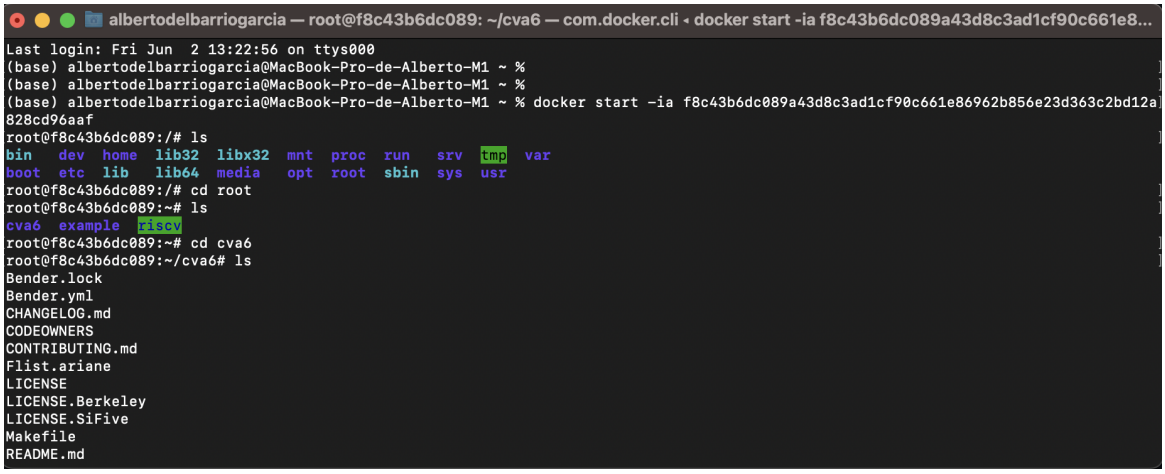
Una vez el usuario haya hecho login, simplemente tiene que abrir un terminal, ir al directorio $HOME/cva6, preparar un programa en C, compilarlo y simularlo con las siguientes instrucciones:

```
riscv64-unknown-elf-gcc programa.c -o programa.elf
make verilate
work-ver/Variane_testharness      $RISCV/riscv64-unknown-elf/bin/pk
programa.elf
```

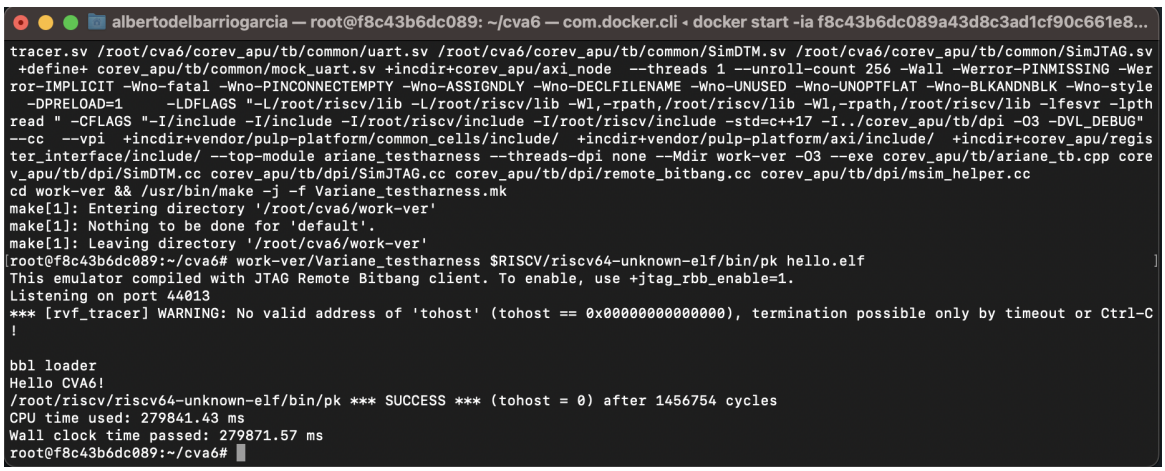
En el caso del docker, lo primero será abrir un terminal e importarlo mediante los siguientes comandos:

```
cat rvDocker.tar | docker import - examplerv:new
docker run -it examplerv:new /bin/bash
```

Una vez el docker esté creado, bastará con utilizar el comando start para arrancarlo.



Una vez hecho esto, hay que moverse al directorio /root/cva6, preparar un programa en C, compilarlo y simularlo con las mismas instrucciones que se han mencionado anteriormente.



6. Anexos

La máquina virtual se encuentra disponible a través del siguiente enlace:
https://drive.google.com/file/d/1-cUu1w8xfPu7gnm2l4ee5zbAXDevtetf/view?usp=sharing

El docker se encuentra disponible a través del siguiente enlace:
https://drive.google.com/file/d/1-ZZkrKJ5mi_xGKj6YQ6YGEnZzVwdOMNX/view?usp=sharing

Bibliografía:

[1] Alexander Dörflinger, Mark Albers, Benedikt Kleinbeck, Yejun Guan, Harald Michalik, Raphael Klink, Christopher Blochwitz, Anouar Nechi, and Mladen Berekovic. 2021. A comparative survey of open-source application-class RISC-V processor implementations. In Proceedings of the 18th ACM International Conference on Computing Frontiers (CF '21). Association for Computing Machinery, New York, NY, USA, 12–20. https://doi.org/10.1145/3457388.3458657

[2] F. Zaruba and L. Benini, "The Cost of Application-Class Process- ing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 11, pp. 2629–2640, Nov. 2019.

[3] David Mallasén, Alberto A. Del Barrio, Manuel Prieto-Matías: Big-PERCIVAL: Exploring the Native Use of 64-Bit Posit Arithmetic in Scientific Computing. CoRR abs/2305.06946 (2023)

[4] David Mallasén, Raul Murillo, Alberto A. Del Barrio, Guillermo Botella, Luis Piñuel, Manuel Prieto-Matías: PERCIVAL: Open-Source Posit RISC-V Core With Quire Capability. IEEE Trans. Emerg. Top. Comput. 10(3): 1241-1252 (2022)

[5] D. Mallasén, R. Murillo, A. A. Del Barrio, G. Botella, L. Piñuel and M. Prieto–Matias, "Customizing the CVA6 RISC-V Core to Integrate Posit and Quire Instructions," 2022 37th Conference on Design of Circuits and Integrated Circuits (DCIS), Pamplona, Spain, 2022, pp. 01-06, doi: 10.1109/DCIS55711.2022.9970026.

[6] R. Murillo, A. A. Del Barrio, G. Botella, M. S. Kim, H. Kim and N. Bagherzadeh, "PLAM: A Posit Logarithm-Approximate Multiplier," in IEEE Transactions on Emerging Topics in Computing, vol. 10, no. 4, pp. 2079-2085, 1 Oct.-Dec. 2022, doi: 10.1109/TETC.2021.3109127.

[7] R. Murillo, D. Mallasén, A. A. Del Barrio, G. Botella.2021. Energy-Efficient MAC Units for Fused Posit Arithmetic. In 2021 IEEE 39th International Conference on Computer Design (ICCD). 138–145.

[8] Murillo, R., Mallasén, D., Del Barrio, A.A., Botella, G. (2022). Comparing Different Decodings for Posit Arithmetic. In: Gustafson, J., Dimitrov, V. (eds) Next Generation Arithmetic. CoNGA 2022. Lecture Notes in Computer Science, vol 13253. Springer, Cham.

[9] Murillo, R., Mallasén, D., Del Barrio, A.A., Botella, G. (2023). PLAUs: Posit Logarithmic Approximate Units to Implement Low-Cost Operations with Real Numbers. In: Gustafson, J., Leong, S.H., Michalewicz, M. (eds) Next Generation Arithmetic. CoNGA 2023. Lecture Notes in Computer Science, vol 13851. Springer, Cham.

[10] Ubuntu release cycle. https://ubuntu.com/about/release-cycle. Accedido el 02/06/2023