Taller de Caché

Organización del Computador 1

Verano 2021

1. Introducción

En este taller, vamos a utilizar un simulador de una máquina RISC-V para entender el funcionamiento del caché en la computadora.

2. El simulador

Ripes es un simulador visual de la arquitectura de una computadora. Posee un editor de assembler hecho para el ISA RISC-V. Nos permite ver:

- Cómo el código es ejecutado en la máquina
- Cómo los diferentes diseños de caché impactan en la performance
- Cómo el codigo assembler es ensamblado y ejecutado en la máquina

2.1. Instalación

Para instalar el simulador deben seguir las instrucciones que figuran en la página: `https://github.com/mortbopet/Ripes#downloading--installation`. Está disponible para Windows, Mac y Linux.

2.2. Interfaz

Una vez que hayan instalado el simulador, es fundamental que ganen familiaridad con la interfaz del simulador. A continuación daremos brevemente un pantallazo sobre las principales funciones que necesitarán para completar el taller.

2.2.1. Cargar un programa

El simulador trae la posibilidad de compilar nuestros propios programas en C o escribir código ensamblador de RISC-V. También, trae ejemplos de código que vamos a usar durante este taller. En particular, vamos a utilizar la implementación de factorial escrita en Assembler.

Para cargar el programa factorial, en el menú, seleccionen **File** → **Load Example...** → **Assembly** → **Factorial function**. En el editor de código van a poder ver el programa cargado.

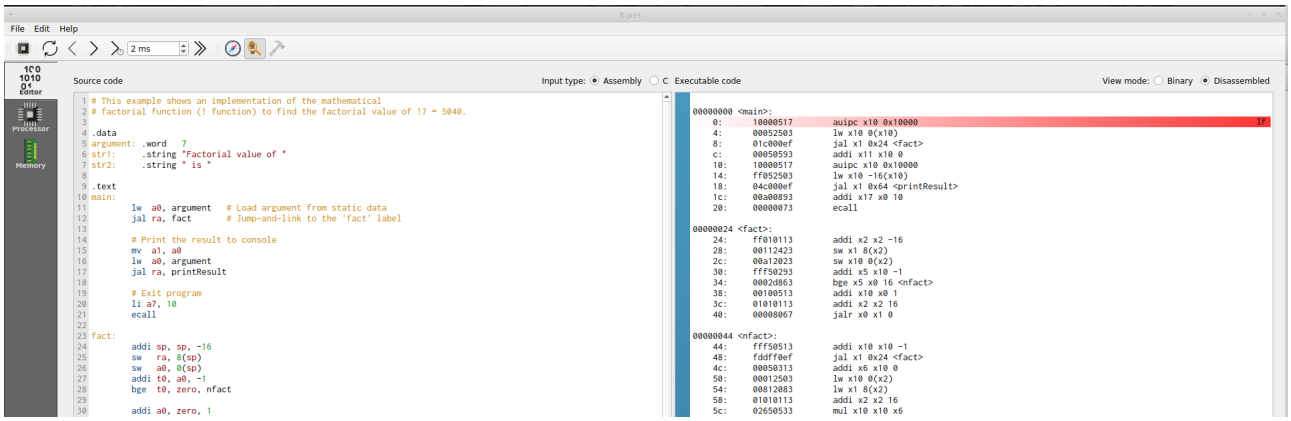

Figura 1: El código del programa factorial cargado en el simulador

2.2.2. La memoria principal y el caché

Durante el taller vamos a entender la organización de la memoria y las diferentes configuraciones de caché que podemos simular. Ripes brinda una documentación sobre la sección de memoria que les puede ser de utilidad: `https://github.com/mortbopet/Ripes/wiki/cache-Simulation`.

El primer ejercicio del taller es una actividad guiada para comprender las diferencias entre el simulador y los visto durante la clase práctica.

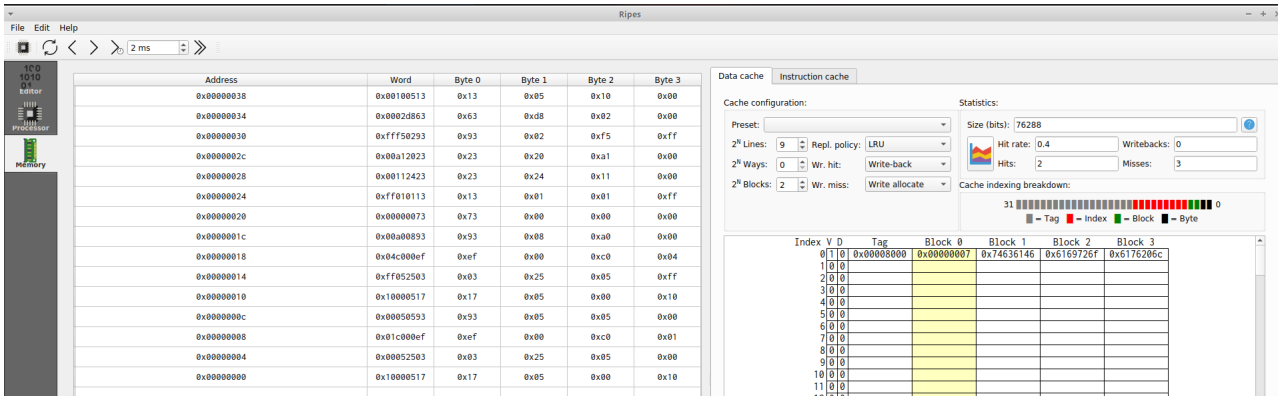

Figura 2: Sección memoria - Pueden encontrar la memoria principal y los cachés

2.2.3. Ejecución del programa

Para ejecutar el programa usaremos los controles que se muestran a continuación:

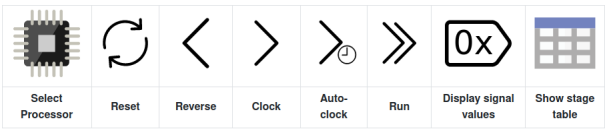

Figura 3: Controlador de ejecución

Mencionamos aquellos que más van a utilizar durante el taller:

- Reset: Resetea el procesador, setea el PC en el incio del programa, y resetea la memoria del simulador.

- Reverse: Undo del ciclo de reloj anterior.
- Clock: Ejecuta un ciclo y actualiza el estado del circuito.
- Auto-clock: Ejecuta varios ciclos a la frecuencia especificada en el intervalo de auto-clock. Auto-clocking va a frenar cuando se llegue a un breakpoint o al final de la ejecución.

En la sección editor, pueden observar la línea por la que va la ejecución y en que etapa del ciclo de instrucción se encuentra (IF - Instruction Fetch, ID - Instruction Decode, y EX - Execute).

Adicionalmente, podemos especificar la velocidad de reloj en milisegundos (Autoclock interval). Esto va a ser muy útil para ejecutar el programa de manera rápida y sacar conclusiones sobre el uso de la memoria.

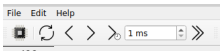

Figura 4: Velocidad del reloj : 1 ms

2.2.4. Bibliografía de consulta

Ya tienen todo lo necesario para comenzar el taller. Mientras avanzan, las siguientes referencias pueden serles de utilidad:

- para una introducción al simulador, visiten el siguiente link: `https://github.com/mortbopet/Ripes/wiki/Ripes-Introduction`
- documentación sobre la sección de memoria: `https://github.com/mortbopet/Ripes/wiki/cache-Simulation`
- adicionalmente, pueden encontrar la wiki del simulador en: `https://github.com/mortbopet/Ripes/wiki`

3. Ejercicios

3.1. Ejercicio 1 - El simulador

Con el primer ejercicio, buscamos entender el funcionamiento del simulador y cómo está organizada la memoria de la máquina. Es importante que configuren el procesador en ***Single Cycle Processor*** con ***Layout Extended***. Pueden hacerlo desde el ícono del procesador:

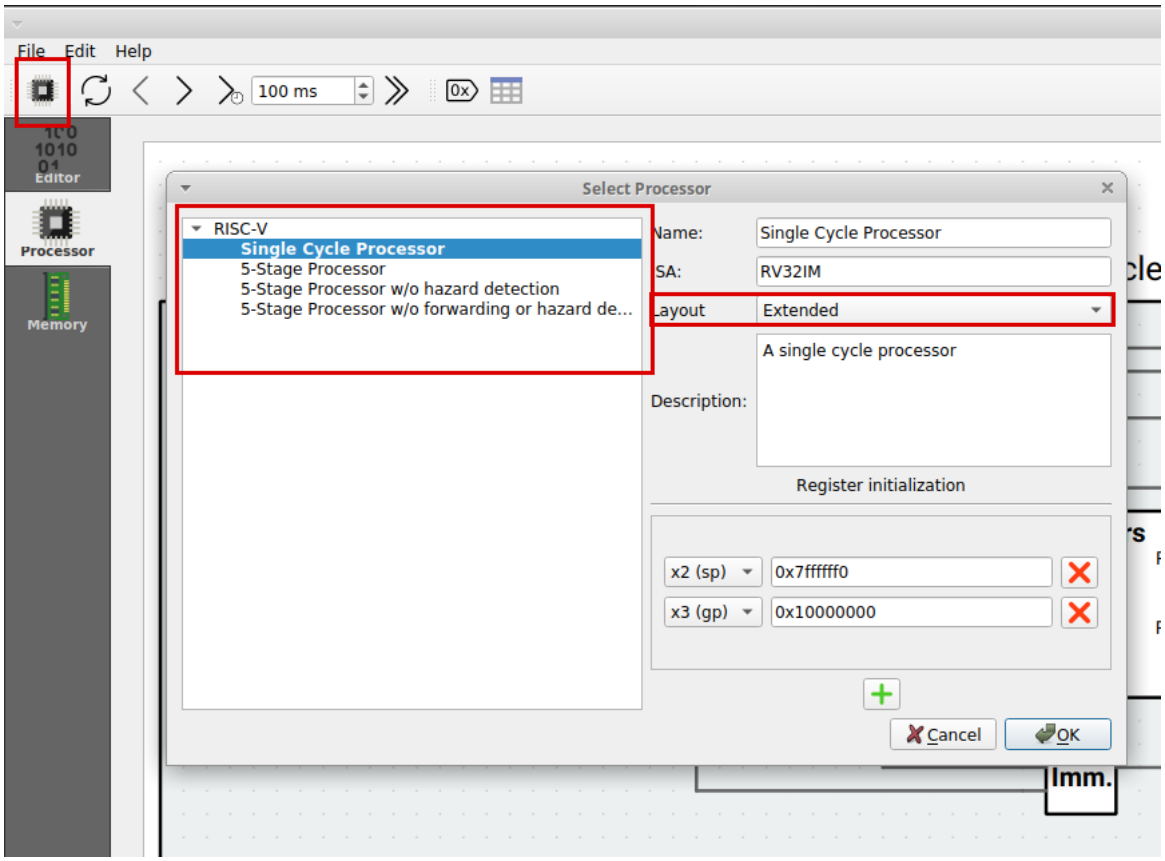

Figura 5: Configuración de la máquina como ***Single Cycle Processor*** con ***Layout Extended***

A continuación puede observar el diagrama de la máquina que vamos a utilizar:

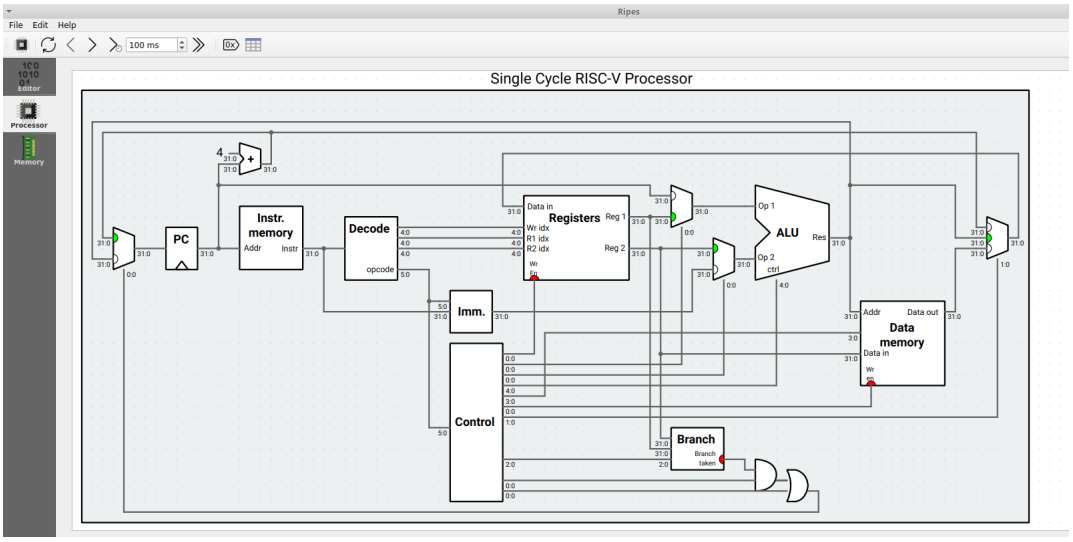

Figura 6: Diagrama de la máquina - Single Cycle RISC-V Processor

Primero, vamos a cargar el programa. Ir al menú opciones **File** → **Load Example...** → **Assembly** → **Factorial function** y, observando el simulador, respondan las siguientes preguntas.

3.1.1. La máquina

Recuerden haber configurado la máquina como ***Single Cycle Processor*** con ***Layout Extended*** y cargado el programa ***Factorial***.

a) Para empezar, observen las dos primeras instrucciones del programa. ¿Cuáles son y qué hacen?

b) Ejecuten la **primer instrucción** y observen las señales que se prenden en el diagrama del Procesador (en *Processor*). ¿Qué componentes intervienen en la ejecución de la primer instrucción?

c) Ejecuten la **segunda instrucción** y observen las señales que se prenden en el diagrama del Procesador (en *Processor*). ¿Qué componentes intervienen en la ejecución de la segunda instrucción?

d) ¿Qué conclusiones pueden sacar sobre la memoria del procesador? ¿Es un arquitectura de Von Neumann?

3.1.2. La memoria principal

a) ¿Cuál es el tamaño de palabra?

b) ¿Cuál es la mínima unidad direccionable?

c) ¿Cuántos bits se usan para indicar una dirección de memoria?

d) ¿Cuál es el tamaño total de la memoria?

e) Exploren las opciones de ***Go to section***, en la parte inferior de la memoria, ¿qué hace cada una de las opciones?

3.1.3. Las configuraciones del caché

En la sección *Memory* tenemos a la derecha los cachés. Exploremos las diferentes opciones que tiene el simulador.

a) ¿Qué cachés tiene el procesador? ¿Para qué se usa cada uno? ¿Qué tamaños tienen?

b) Exploren las opciones de ***Preset***. ¿Qué configuraciones de caché trae el simulador? Relacionen las diferentes opciones con los tipos de caché vistos durante la práctica.

c) Expliquen a que se refieren los términos *Index*, *Lines*, *Ways*, *Blocks* y *Byte* del caché en el simulador. Indiquen la relación con los términos vistos durante la práctica: *Indice*, *Líneas*, *Vias* y *Bloques*.

d) ¿Qué indica *Repl. policy* y qué opciones tiene? Compárenlas con las vistas en la materia, ¿cuáles estarían faltando?

e) Expliquen brevemente qué es y cómo funciona LRU acorde con lo visto en la práctica

Checkpoint 1

3.2. Ejercicio 2 - Correspondencia directa

A modo de aclaración, el formato de indexación en la caché incluye el byte. Los dos últimos bits van a ser 0. Pueden ayudarse con el *cache indexing breakdown* disponible en el simulador.

Ejecutemos hasta la segunda instrucción inclusive. El primer elemento se debe haber cargado en la caché. Ahora respondan observando **únicamente la caché de datos**:

a) ¿Qué indican las columnas de la caché: Index, V, D, Tag, Block 0, Block 1, Block 2 y Block 3?

b) Recordemos que estamos en mapeo directo, ¿cuál es el formato de mapeo de las direcciones de memoria a la línea de caché en el simulador? Detalle la cantidad bits necesarios para el tag, línea, e índice.

c) ¿A qué dirección de memoria corresponde el dato que se cargó en la caché? ¿Fue un HIT? Justifiquen la respuesta.

d) ¿Cuántas celdas (blocks) de memoria trajo y a qué direcciones corresponden?

e) Acorde con lo explicado en la práctica, detallen los motivos por los cuales en la caché se almacenan también las celdas contiguas a la única celda de memoria solicitada.

f) Completen la siguiente tabla usando el mapeo de dirección de memoria a línea de caché. Luego, utilizando el simulador verifiquen los resultados.

Dirección	Tag	línea (Index)	# Indice (Block+Byte)	Hit/Miss
0x10000000				
	0x003FFFFF	11110	1000	
0x7FFFFFE0				
0x7FFFFFD8				
0x7FFFFFD0				
	0x003FFFFF	11100	1000	
	0x003FFFFF	11100	0000	

Checkpoint 2

3.3. Ejercicio 3 - Completamente Asociativa

Ahora, vamos a trabajar con una caché completamente asociativa. Elijan en el simulador el *Preset*: *32-entry 4-word fully associative*, carguen nuevamente el programa factorial **File** → **Load Example...** → **Assembly** → **Factorial function**. Reinicien el simulador. Indiquen:

a) Al elegir la política totalmente asociativa, se agregó una columna LRU. Ejecuten el simulador varias veces, ¿con que valores inicia? ¿qué ocurre con los valores de la columna LRU? ¿para qué se usa la nueva columna?

b) Modifiquen el programa para calcular el factorial de 32. Ejecuten muchas más veces el simulador hasta llenar la caché (recomendamos a velocidad 1 ms). Predigan, una vez que se haya llenado el caché, ¿cuál sería la próxima línea a desalojar en caso de un MISS? Indiquen qué criterio utilizaron para elegir esa línea. Verifiquen con el simulador si es la línea que predijeron.

c) Recordemos que estamos utilizando una caché completamente asociativa, ¿cuál es el formato de mapeo de las direcciones de memoria a la línea de caché en el simulador? Detallen la cantidad bits necesarios para el tag e índice.

d) Completen el comportamiento de la caché de datos para los siguientes accesos a memoria:

Dirección	Tag	# Indice (Block+Byte)	Hit/Miss
0x10000000			
	0x07FFFFFE	1000	
0x7FFFFFE0			
0x7FFFFFD8			
0x7FFFFFD0			
	0x07FFFFFC	1000	
	0x07FFFFFC	0000	

e) Establezcan el autoclock a 50 ms, vayan al ***Editor*** y vean cómo se ejecuta el programa. Luego, repitan la ejecución pero observando la caché de instrucciones (no la de datos) en ***Memory***. ¿Qué conclusiones pueden sacar sobre la ejecución del programa y el uso del caché de instrucciones?

Checkpoint 3