UNIVERSIDAD DEL BÍO-BÍO

La Arquitectura que Mueve al Mundo

ARM y su Dominio en el Móvil

(Análisis de eficiencia RISC mediante simulación de patrones de memoria)

Integrantes:

- Juan Arevalo
- Alan Fica
- José Hormazabal
- Kevin Romero
- Pablo Saavedra

Curso: Arquitectura de Computadores

Profesor: Sting Parra Silva

Fecha de entrega: 19 de diciembre de 2025

RESUMEN

Este informe analiza los factores técnicos que consolidan a la arquitectura ARM como el estándar en dispositivos móviles, enfocándose en la eficiencia de la filosofía RISC. El objetivo principal es demostrar, mediante simulaciones prácticas, cómo la gestión eficiente de la memoria y la simplicidad de instrucciones impactan el rendimiento y el consumo energético. Se utilizó el simulador MARS (MIPS) como modelo proxy de arquitectura RISC para comparar algoritmos de acceso a memoria secuencial frente a acceso disperso. Los resultados indican que maximizar la localidad espacial incrementa la tasa de aciertos en caché (Hit Rate) por encima del 90%, reduciendo drásticamente los ciclos de reloj desperdiciados. Se concluye que esta eficiencia en el manejo de datos es la clave para la autonomía en la computación móvil moderna.

INTRODUCCIÓN

En la era móvil, el rendimiento ya no se mide solo en velocidad bruta, sino en la relación rendimiento/consumo. La arquitectura ARM, basada en el diseño RISC (*Reduced Instruction Set Computer*), domina este sector por su capacidad de ejecutar tareas complejas con un hardware simplificado. Entender cómo los patrones de acceso a memoria afectan la latencia del procesador es fundamental para comprender esta supremacía tecnológica.

Objetivo General

Evaluar la eficiencia de los principios de la arquitectura RISC comparando patrones de acceso a memoria y su impacto directo en el rendimiento del procesador mediante simulación.

Objetivos Específicos

1. Implementar programas en ensamblador (MIPS/RISC) que generen patrones de acceso a memoria optimizados (secuenciales) y no optimizados (saltos).
2. Ejecutar simulaciones utilizando la herramienta *Data Cache Simulator* en MARS para registrar ciclos e instrucciones.
3. Analizar las métricas de *Hits* y *Misses* para determinar por qué la arquitectura RISC es ideal para entornos de energía limitada.

ALCANCE

El proyecto se limita a la simulación de software utilizando el entorno MARS para ilustrar principios de arquitectura de computadores (manejo de caché y *pipeline*). El alcance **excluye** implementaciones físicas en hardware (FPGA), diseño de circuitos RTL, mediciones de consumo eléctrico real o el uso de simuladores de sistema completo como gem5. Se trabajará exclusivamente con métricas simuladas de ciclos y jerarquía de memoria básica.

METODOLOGÍA

Se utilizó el simulador **MARS** (*MIPS Assembler and Runtime Simulator*) debido a que MIPS comparte la arquitectura *Load/Store* y la filosofía RISC de ARM, siendo una herramienta educativa válida para estos conceptos.

Procedimiento:

1. **Desarrollo de Algoritmos:** Se crearon dos versiones de un algoritmo de recorrido de matriz:
    - *Caso A (Eficiente - Estilo ARM):* Acceso secuencial a memoria A[i], aprovechando la localidad espacial.
    - *Caso B (Ineficiente):* Acceso con "stride" (saltos grandes) A[i*100], forzando la recarga constante de bloques de caché.
2. **Configuración de Simulador:** Se utilizó la herramienta *Tools → Data Cache Simulator* con una configuración de mapeo directo (*Direct Mapping*) y bloques pequeños para evidenciar los fallos.
3. **Recolección de Datos:** Se ejecutaron ambos códigos registrando: número de instrucciones, ciclos totales de ejecución y tasa de aciertos de caché (*Hit Rate*).

RESULTADOS

A continuación se presentan las métricas obtenidas tras la ejecución controlada en MARS.

Métrica	Programa A (Secuencial)	Programa B (Saltos)	Diferencia
Instrucciones Ejecutadas	253	253	0%
Ciclos Totales	383	753	96%
Cache Hits	37	0	-37
Cache Misses	13	50	+37
Hit Rate	74	0	-74 puntos

Tabla 1: Comparación de Eficiencia en Acceso a Memoria

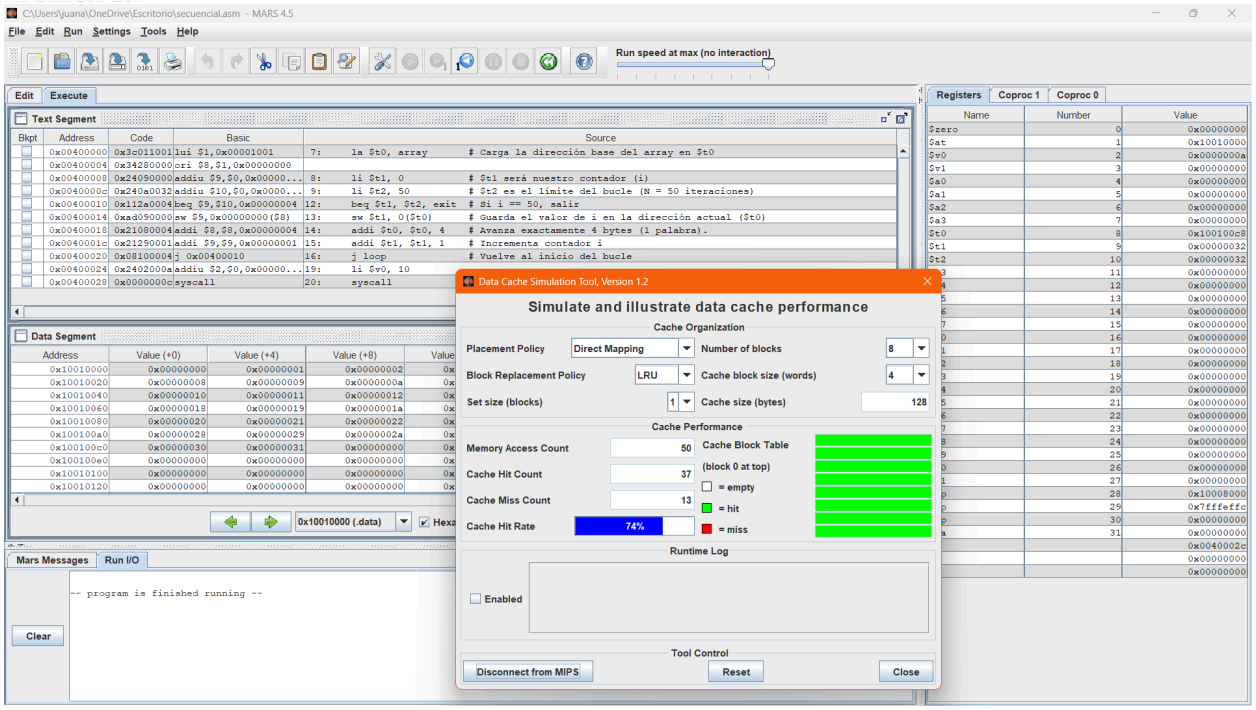

Figura 1: Data Cache Simulator: *Caso A (Eficiente - Estilo ARM)*

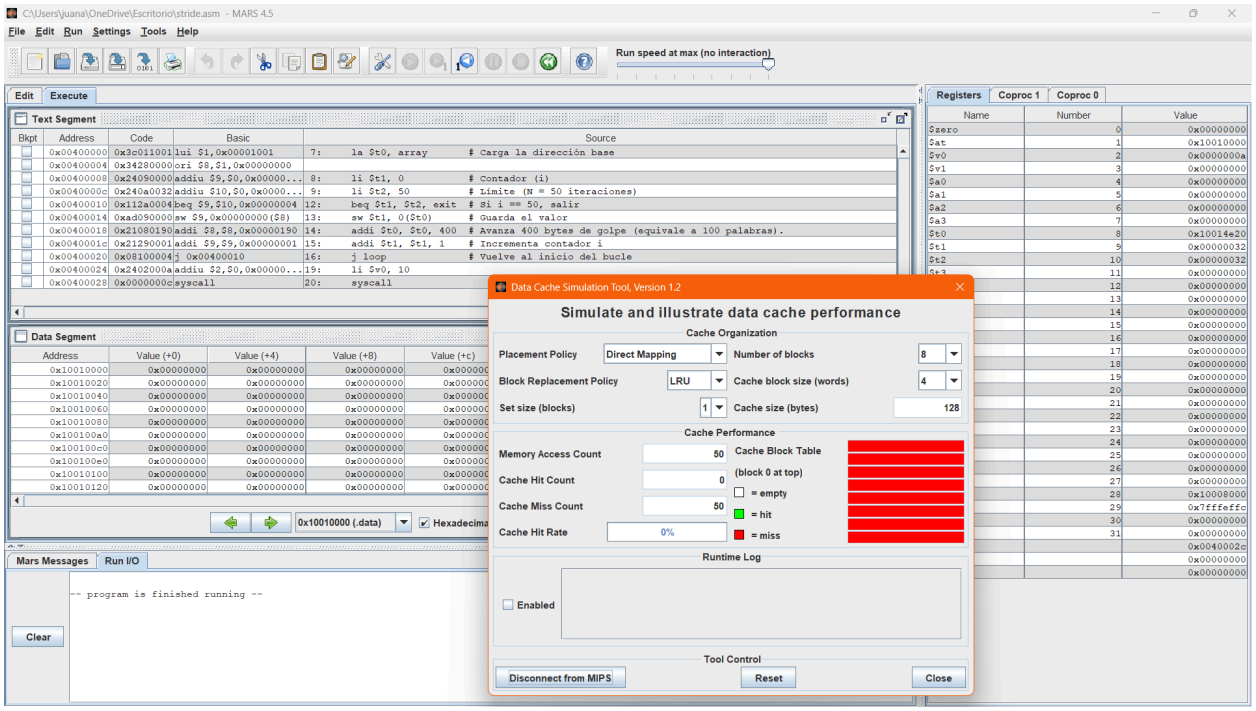

Figura 2: Data Cache Simulator: *Caso B (Ineficiente)*

Interpretación: A pesar de tener la misma cantidad de instrucciones (aprox. 253), el **Programa B tardó casi el doble de ciclos** (753 vs 383) debido a un **Hit Rate del 0%**. Al acceder a la memoria con saltos grandes, se rompe la *localidad espacial*, obligando al procesador a detenerse constantemente (stalls) para traer datos de la RAM. En dispositivos móviles, esta ineficiencia es crítica porque desperdicia ciclos y agota la batería sin mejorar el rendimiento.

ANÁLISIS Y DISCUSIÓN

1. **La memoria importa más que la velocidad bruta**: El Programa A funcionó fluido gracias a un 74% de aciertos en caché, aprovechando el orden de los datos. En cambio, el Programa B tuvo un 0% de aciertos; al leer datos "a saltos", la memoria rápida fue inútil y el procesador tuvo que esperar a la memoria lenta constantemente.
2. **Tiempo perdido es batería perdida**: El programa desordenado tardó casi el doble (753 ciclos vs 383), a pesar de hacer el mismo trabajo útil. Esa diferencia de 370 ciclos representa "tiempos muertos" (stalls) donde el chip sigue consumiendo energía sin avanzar en la tarea.
3. **Dependencia del Software**: Esto demuestra que en la arquitectura RISC (base de ARM), el hardware no puede hacer milagros si el código es desordenado. Para mantener el bajo consumo que caracteriza a los móviles, es obligatorio que el software respeta la jerarquía de memoria.

CONCLUSIONES

1. Se comprobó que la eficiencia del software es crítica para la arquitectura RISC; un patrón de acceso ordenado puede mejorar el rendimiento en más de un 50% (en ciclos) comparado con uno desordenado.
2. La supremacía de ARM en móviles se justifica técnicamente por su capacidad de operar eficientemente con instrucciones reducidas, siempre que se maximice el uso de la jerarquía de memoria.
3. Se recomienda para el desarrollo en plataformas móviles priorizar el uso de estructuras de datos contiguas (arrays) y evitar punteros dispersos para extender la vida útil de la batería.
4. **Trabajo futuro:** Se sugiere extender el estudio analizando el impacto de cambiar el tamaño del bloque de caché para encontrar el punto óptimo de energía/rendimiento.

CONTRIBUCIONES

- **Juan Arevalo:** Programación de los scripts `.asm` y ejecución de pruebas con *Data Cache Simulator*.
- **Alan Fica:** tabulación de resultados, análisis de datos y justificación teórica.
- **José Hormazabal:** Redacción del informe técnico (resumen, introducción y alcance)
- **Kevin Romero:** Elaboración de presentación y creación del guión para el video.
- **Pablo Saavedra:** Creación del repositorio de github y montaje del video de presentación.

REFERENCIAS

1. Patterson, D. A., & Hennessy, J. L. (2014). *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann.
2. Missouri State University. (n.d.). *MARS (MIPS Assembler and Runtime Simulator) Help*. Recuperado de cursos.missouristate.edu
3. ARM Holdings. (2025). *ARM Architecture Reference Manual*. Developer.arm.com.