Arquitectura de computadores I

Grupo 2

Documento de diseño

Profesor: Jeferson González Gómez

Estudiantes: Gonzalo Acuña Madrigal

Adriel Chaves Salazar

Daniel Duarte Cordero

Marco Rodríguez Villegas

I Semestre 2025

Tabla de contenidos

Resumen Ejecutivo

El presente informe detalla la documentación de diseño para el "Proyecto Grupal 1: Arquitectura del Set de Instrucciones (ISA) Específica tipo RISC para Aplicaciones de Seguridad Informática". El objetivo central de este proyecto es el diseño de una Arquitectura del Set de Instrucciones (ISA) RISC personalizada y su microarquitectura correspondiente, con soporte nativo de hardware para el algoritmo de cifrado Tiny Encryption Algorithm (TEA) y una bóveda de llaves criptográficas segura.

Se exploraron dos opciones de diseño principales, cada una con enfoques distintos para la aceleración del cifrado y la gestión de la bóveda de llaves. La primera opción se centra en una aceleración granular de TEA y un acceso directo a la bóveda de llaves mediante instrucciones específicas. La segunda opción propone una aceleración de ronda completa de TEA y un modelo de bóveda de llaves basado en punteros seguros. Tras un análisis comparativo riguroso, la propuesta final seleccionada prioriza el equilibrio entre la eficiencia computacional, la complejidad del hardware y la robustez de la seguridad. Esta elección garantiza el cumplimiento de los requisitos del proyecto, ofreciendo una solución que maximiza la protección de datos sensibles y la eficiencia del cifrado, al tiempo que gestiona de forma efectiva los recursos de hardware.

Introducción y Contexto del Proyecto

El proyecto actual se enmarca en el curso CE-4301 Arquitectura de Computadores I del Instituto Tecnológico de Costa Rica, con el mandato explícito de diseñar una Arquitectura del Set de Instrucciones (ISA) tipo RISC específica para aplicaciones de seguridad informática. Este esfuerzo implica no solo la concepción de la ISA, sino también la implementación de un modelo de software funcionalmente equivalente a un procesador, capaz de ejecutar y validar las capacidades de la arquitectura diseñada.

El problema central por abordar es el diseño de una ISA con soporte nativo para el cifrado eficiente de datos utilizando el algoritmo Tiny Encryption Algorithm (TEA) y para el almacenamiento seguro de llaves criptográficas en una bóveda de hardware. Este documento de diseño se estructura para detallar los requisitos, las opciones de solución consideradas, una comparación objetiva de estas opciones y la justificación de la propuesta final seleccionada.

Análisis de Requisitos de Seguridad y Arquitectura

La definición precisa de los requisitos es el pilar de cualquier diseño arquitectónico robusto. Para este proyecto, los requisitos se derivan de la especificación del "Proyecto Grupal 1" y se dividen en categorías de aplicaciones de seguridad, requisitos generales de la ISA y requisitos de implementación y validación.

Requisitos de Aplicaciones de Seguridad

El diseño de la ISA debe proporcionar soporte de hardware robusto para dos áreas críticas de seguridad: el cifrado eficiente de datos y el almacenamiento seguro de llaves criptográficas.

Cifrado Eficiente (TEA)

El algoritmo de cifrado seleccionado es el Tiny Encryption Algorithm (TEA). Este algoritmo de cifrado de bloque es conocido por su baja complejidad computacional y eficiencia, lo que lo hace adecuado para sistemas con recursos limitados. Las especificaciones clave para la implementación de TEA son:

- **Algoritmo:** Tiny Encryption Algorithm (TEA).
- **Tamaño de Llave:** 128 bits.
- **Tamaño de Bloque:** 64 bits (o dos bloques de 32 bits).
- **Rondas:** 32 rondas de cifrado por bloque.
- **Constante:** DELTA (o "magic constant") con un valor de 0x9e3779b9.
- **Funciones de Referencia:** Se proporcionan funciones de cifrado y descifrado en C para referencia (tea_encrypt, tea_decrypt).

Un requisito explícito es la inclusión de al menos dos instrucciones específicas en el set para acelerar el algoritmo TEA. Es crucial considerar un equilibrio entre la eficiencia y el costo potencial en términos de área y complejidad del hardware. La especificación advierte específicamente que una única instrucción para el cifrado o descifrado completo resultaría en un módulo excesivamente complejo y costoso. Esta directriz es fundamental, ya que obliga a la estrategia de diseño a considerar instrucciones más granulares que aceleren *partes* del algoritmo TEA, en lugar de una operación macro. Por ejemplo, en el diseño de conjuntos de instrucciones para el Advanced Encryption Standard (AES), es

común encontrar instrucciones que realizan una única ronda de cifrado (como AESENC o AESDEC), en lugar de la operación completa.[3] Este enfoque granular permite una aceleración significativa de las operaciones criptográficas centrales, como las operaciones de desplazamiento, adición y XOR que caracterizan a TEA, al tiempo que mantiene la complejidad del hardware en un nivel manejable.

Almacenamiento Seguro de Llaves (Bóveda)

La gestión segura de llaves criptográficas es un pilar de la ciberseguridad. Para este proyecto, se debe modelar una raíz de confianza (Root of Trust, RoT) en forma de una bóveda de llaves, que funcione como una memoria segura. Los requisitos para esta bóveda son estrictos:

- **Capacidad:** Al menos 4 llaves de 128 bits cada una.
- **Acceso:** La bóveda debe ser accesible directa y *solamente* por la CPU, a través de instrucciones específicas del set. Es fundamental que la bóveda *no* pueda ser accedida como una memoria tradicional.
- **Aislamiento:** Los datos almacenados en la bóveda *no* pueden ser escritos en registros de la CPU ni en la memoria general del sistema.
- **Uso:** Solamente las instrucciones que operen directa, parcial o totalmente con las llaves pueden acceder a la bóveda como operando fuente, tomando la aplicación de cifrado (TEA) como referencia.

Estos requisitos de la bóveda de llaves definen un modelo de seguridad muy específico, similar al de un Módulo de Seguridad de Hardware (HSM) o el almacenamiento seguro de llaves en un Entorno de Ejecución Confiable (TEE). No se trata simplemente de una región de memoria con permisos restringidos, sino de un componente de hardware dedicado y aislado. Además, resuena con el principio de "aislar el estado del procesador" en el diseño de arquitecturas seguras.[6] La implicación directa es que la microarquitectura debe incluir una ruta de datos dedicada y segura para el material de la llave desde la bóveda directamente a las unidades funcionales que lo utilizan, evitando por completo los registros de propósito general y las cachés de datos. Esto tiene consecuencias significativas para el diseño de la unidad de control y la ruta de datos, garantizando un aislamiento estricto y previniendo la fuga de información a través de canales laterales que podrían involucrar registros de propósito general o la memoria principal.

Requisitos Generales de la Arquitectura del Set de Instrucciones

Más allá de las aplicaciones de seguridad, la ISA debe cumplir con criterios arquitectónicos generales:

- **Diseño Personalizado:** La arquitectura debe ser completamente diseñada por los estudiantes y no debe basarse en ISAs existentes (ej. ARM, x86, RISC-V), requiriendo una justificación para cada característica.
- **Instrucciones Esenciales:** Aunque la cantidad de instrucciones es libre, se deben proveer al menos instrucciones para control de flujo, operaciones aritméticas-lógicas y acceso a memoria.
- **Detalles de Diseño:** Se deben especificar y justificar aspectos como modos de direccionamiento, tamaño y tipo de datos, tipo y sintaxis de las instrucciones, registros disponibles y sus nombres, y la codificación y descripción funcional de las instrucciones.
- **Justificación de Diseño:** Todos los detalles deben ser justificados desde la perspectiva de diseño (complejidad, costo, área, recursos disponibles), idealmente con análisis de software de alto nivel.

Requisitos de Implementación y Validación

El proyecto exige la implementación de un modelo de software que simule el procesador y su interacción con la memoria.

- **Modelo de Software:** El modelo de la CPU debe ser ejecutable y funcionalmente equivalente a un procesador real, incluyendo componentes internos (banco de registros, unidades funcionales, unidad de control) y un modelo de memoria.
- **Métricas y Visualización:** El modelo debe proporcionar métricas de ejecución (mínimo: cantidad de ciclos) y permitir la visualización de elementos clave como registros internos de la CPU y la bóveda de llaves criptográficas durante la simulación.
- **Herramientas de Desarrollo:** Se debe desarrollar un compilador/ensamblador que traduzca las instrucciones de la ISA a binario para su carga y ejecución.

- **Documentación Adicional:** Incluir un diagrama de bloques de la microarquitectura y una descripción de las interacciones entre sus componentes en el documento de diseño.
- **Validación de Seguridad:** Se realizarán pruebas de cifrado y descifrado de archivos de tamaño fijo (1KB) utilizando el algoritmo TEA y las llaves proporcionadas, verificando la integridad con un hash MD5 y mostrando el contenido descifrado. Es crucial que todas las operaciones de cifrado/descifrado con llaves utilicen la bóveda.

Elaboración de Opciones de Solución para la ISA y Microarquitectura

Para abordar los requisitos del proyecto, se han conceptualizado dos opciones de diseño principales para la ISA y su microarquitectura asociada. Ambas opciones se adhieren a los principios RISC, buscando simplicidad, regularidad y facilidad de pipelining, tal como se especifica en el proyecto. Además, esta selección de base para la microarquitectura se justifica por su equilibrio entre los factores anteriormente mencionados. En comparación con un diseño uniciclo, el pipelining permite reducir significativamente la duración del ciclo de reloj, evitando que operaciones simples se vean penalizadas por las más complejas. Frente a un esquema multiciclo, se logra mantener una lógica de control menos intrusiva y más regular, facilitando la implementación de instrucciones personalizadas, como las del algoritmo TEA, sin introducir lógica secuencial específica para cada caso.

El pipeline además mejora la capacidad del sistema para escalar y ejecutar programas más complejos con mayor fluidez, permitiendo la integración de mecanismos de seguridad y control sin comprometer la eficiencia del sistema. Un principio rector fundamental en este diseño es la prioridad de la seguridad, lo que significa que las características de seguridad pueden introducir cierta complejidad o sobrecarga de área, aspectos que se aceptan como parte integral de un diseño seguro.[6] Aunque el proyecto prohíbe el uso de ISAs existentes como RISC-V , la justificación para las instrucciones personalizadas es análoga a la observada en las extensiones de RISC-V: optimizar cargas de trabajo especializadas, como la criptografía, donde las instrucciones de propósito general son ineficientes.[8] Los principios de diseño de RISC-V, incluyendo su simplicidad, modularidad, extensibilidad y eficiencia energética [11], son altamente relevantes para justificar las decisiones de diseño en esta nueva ISA RISC personalizada.

Ambas opciones de diseño contemplan la integración de unidades funcionales especializadas para operaciones criptográficas. Esta aproximación refleja una tendencia más amplia en el diseño de procesadores modernos, donde las CPUs de propósito general se complementan con aceleradores dedicados (como unidades vectoriales, unidades criptográficas o aceleradores de IA) para manejar eficientemente cargas de trabajo específicas.[3] Para el algoritmo TEA, esto implica la necesidad de bloques de hardware dedicados, diseñados para ejecutar las operaciones de desplazamiento, adición y XOR de manera significativamente más rápida que una ALU de propósito general, especialmente si se implementan con pipelining interno. Este compromiso micro arquitectónico es esencial para lograr un alto rendimiento en la aplicación objetivo.

Opción de Solución 1: ISA con Aceleración Granular de TEA y Bóveda de Llaves con Acceso Directo por Instrucción

Esta opción se enfoca en la aceleración de componentes específicos del algoritmo TEA y un modelo de acceso a la bóveda de llaves que garantiza la máxima contención del material criptográfico.

Descripción de la ISA

- **Tipos de Datos:** La arquitectura operará principalmente con datos de 32 bits, lo que se alinea con el tipo uint32_t utilizado en el código de referencia de TEA. Para el procesamiento de bloques de TEA de 64 bits, se soportarán operaciones que manejen dos palabras de 32 bits.
- **Registros:** Se dispondrá de un número moderado de registros de propósito general (por ejemplo, 16 o 32 registros de 32 bits) para cálculos generales. Crucialmente, existirán registros internos dedicados para la aceleración de TEA y las operaciones de la bóveda de llaves, los cuales no estarán expuestos como registros de propósito general.
- **Modos de Direccionamiento:** Se implementará una arquitectura simple de carga/almacenamiento (load/store), utilizando modos de direccionamiento como el registro-indirecto con desplazamiento, lo que es coherente con los principios RISC y el acceso a bloques de datos en memoria.
- **Instrucciones Generales:** Se incluirán instrucciones estándar para operaciones aritméticas (ADD, SUB), lógicas (AND, OR, XOR, SHL, SHR), control de flujo (BEQ, BNE, JAL, JR) y acceso a memoria (LOAD, STORE).
- **Instrucciones Personalizadas TEA:** Diseñadas para acelerar *partes* de una ronda de TEA, logrando un equilibrio entre la eficiencia y la complejidad del hardware, dado que una única instrucción para el cifrado/descifrado completo se considera demasiado compleja y costosa.
    - TEA_ROUND_UPDATE_V0 (tipo R): Realiza la lógica de actualización de v0 para una ronda: v0 += ((v1<<4) +key) ^(v1+sum) ^((v1>>5) +key); Esta instrucción tomaría v0, v1, sum y dos componentes de llave (desde un búfer interno de la bóveda) como operandos, produciendo un nuevo v0.

- TEA_ROUND_UPDATE_V1 (tipo R): Realiza la lógica de actualización de v1 para una ronda: v1 += ((v0<<4) +key) ^(v0+sum) ^((v0>>5) +key); Operandos similares, produciendo un nuevo v1.
- TEA_SUM_DELTA_ADD/TEA_SUM_DELTA_SUB (tipo R/I): Instrucciones para actualizar la variable sum (sum += DELTA o sum -= DELTA). Aunque podrían ser simples adiciones, una instrucción dedicada podría asegurar el uso eficiente de la constante DELTA o permitir optimizaciones específicas.
- La decisión de acelerar *partes* de una ronda en lugar de una ronda completa mantiene la complejidad del hardware manejable, evitando un "módulo muy complejo y costoso". Este enfoque se alinea con la granularidad observada en los conjuntos de instrucciones AES, que a menudo proporcionan instrucciones para rondas individuales.[4] Además, aprovecha directamente las operaciones centrales de TEA (SHL, SHR, ADD, XOR) en el hardware, maximizando el beneficio de las instrucciones personalizadas.

- **Instrucciones Personalizadas para la Bóveda de Llaves:** Diseñadas para aplicar un estricto cumplimiento del aislamiento de llaves, asegurando que las llaves nunca ingresen a registros de propósito general o a la memoria principal. Esto refleja el principio de KeyVisor de que las llaves son accesibles "solo por KeyVisor" y "nunca se filtran a la memoria".[4]
  - VAULT_STORE_KEY (tipo I/R): Almacena una llave de 128 bits (por ejemplo, desde dos registros de propósito general de 64 bits o dos cargas de memoria) en una de las 4 ranuras de la bóveda, especificada por un operando inmediato o de registro (0-3). Esta instrucción debe manejar internamente la transferencia segura.
  - VAULT_LOAD_KEY_PARTIAL (tipo R): Carga una *palabra específica de 32 bits* de una llave de 128 bits desde la bóveda a un *búfer interno, no de propósito general*, para ser utilizada *únicamente* por instrucciones TEA_ROUND_UPDATE subsiguientes. La instrucción especificaría la ranura de la bóveda (0-3) y el índice de la palabra de 32 bits (0-3). Esta "carga parcial" asegura que solo el componente de llave necesario esté disponible para la unidad funcional, minimizando la exposición.

Descripción de la Microarquitectura

La microarquitectura se basaría en un pipeline RISC estándar (Fetch de Instrucción, Decodificación de Instrucción, Ejecución, Acceso a Memoria, Writeback). Los componentes clave incluirían:

- **Banco de Registros:** Para los registros de propósito general.
- **ALU:** Para operaciones aritméticas y lógicas estándar.
- **Unidad Funcional Dedicada para TEA (TEA-FU):** Una unidad especializada capaz de realizar las complejas operaciones de desplazamiento-adición-XOR para las instrucciones TEA_ROUND_UPDATE_V0 y TEA_ROUND_UPDATE_V1. Esta unidad tendría entradas directas desde el banco de registros (para v0, v1, sum) y desde el búfer interno de la bóveda de llaves.
- **Módulo de Bóveda de Llaves Segura:** Un módulo de hardware dedicado (por ejemplo, basado en SRAM, aislado del bus de memoria principal) que almacena 4 llaves de 128 bits. Incluiría una lógica de control de acceso robusta que solo respondería a las instrucciones VAULT_STORE_KEY y VAULT_LOAD_KEY_PARTIAL, forzando el aislamiento.
- **Búfer Interno de la Bóveda de Llaves:** Un pequeño búfer interno de la CPU (por ejemplo, de 128 bits) dentro del Módulo de Bóveda de Llaves o la TEA-FU, para retener temporalmente partes de las llaves para la computación, asegurando que nunca lleguen a los registros de propósito general.
- Unidad de Control: Mejorada para decodificar y gestionar la ejecución de instrucciones personalizadas de TEA y la bóveda de llaves, orquestando el flujo de datos seguro entre el banco de registros, la TEA-FU y la bóveda de llaves.
  El flujo de datos implicaría que los valores de v0, v1 y sum fluyan desde los registros a la TEA-FU, mientras que las partes de la llave se obtendrían directamente de la Bóveda de Llaves a la TEA-FU, y los resultados se escribirían de nuevo en los registros de propósito general (para v0, v1, sum).

Justificación

- **Eficiencia:** Las instrucciones TEA granulares permiten una aceleración significativa de las operaciones criptográficas centrales, reduciendo los ciclos en comparación con las implementaciones puramente por software.[12]
- **Costo (Área/Complejidad):** Descomponer TEA en instrucciones dedicadas más pequeñas reduce la complejidad de cada unidad funcional, equilibrando la eficiencia con el costo del hardware.

- **Seguridad:** El estricto modelo de acceso a la bóveda de llaves proporciona un fuerte aislamiento forzado por hardware, previniendo la fuga de llaves.[1] Esto se adhiere a los principios de diseño de procesadores seguros.[6]

Opción de Solución 2: ISA con Aceleración de Ronda Completa de TEA y Bóveda de Llaves con Punteros Seguros

Esta opción busca simplificar la interfaz de programación al realizar una ronda completa de TEA en una sola instrucción, introduciendo un nivel de abstracción en la gestión de llaves.

Descripción de la ISA

- **Tipos de Datos, Registros, Modos de Direccionamiento, Instrucciones Generales:** Similares a la Opción 1, manteniendo los principios RISC.
- **Instrucciones Personalizadas TEA:** Diseñadas para realizar una *ronda completa* de cifrado o descifrado TEA en una sola instrucción, lo que simplifica la interfaz de software, pero potencialmente aumenta la complejidad del hardware. Esto es análogo a las instrucciones AESENC/AESDEC que se encuentran en ISAs comerciales.[3]
    - TEA_ENCRYPT_ROUND (tipo R): Realiza *una ronda completa* de cifrado TEA. Toma v0, v1, sum de los registros de propósito general y un índice de llave (0-3) de la bóveda. Internamente, obtendría los cuatro componentes de llave de 32 bits de la bóveda para la ronda. Produce los v0, v1 y sum actualizados.
    - TEA_DECRYPT_ROUND (tipo R): Realiza *una ronda completa* de descifrado TEA. Operandos y salidas similares, pero ejecutando las operaciones inversas de TEA.
- **Instrucciones Personalizadas para la Bóveda de Llaves:** Proporciona un nivel superior de abstracción para la gestión de llaves, similar a los "punteros de llave protegidos" de KeyVisor.[4] El puntero en sí no contendría material de llave, lo que mejora la seguridad por ofuscación y la aplicación por hardware.
    - VAULT_STORE_KEY_HANDLE (tipo I/R): Almacena una llave de 128 bits en una ranura de la bóveda y devuelve un *puntero seguro* (por ejemplo, un ID entero pequeño o un puntero de capacidad) a un registro de propósito general. Este puntero es opaco para el software, solo utilizable por instrucciones criptográficas específicas.
    - VAULT_USE_KEY_HANDLE (tipo R): Esta instrucción sería implícita en TEA_ENCRYPT_ROUND/TEA_DECRYPT_ROUND, donde el índice de la llave (puntero) se pasa como un operando. La CPU validaría internamente el puntero y recuperaría la llave.

Descripción de la Microarquitectura

La estructura del pipeline sería similar, pero con una etapa de ejecución más compleja para las operaciones criptográficas. Los componentes clave incluirían:

- **Banco de Registros, ALU:** Estándar.
- **Unidad Funcional Compleja para TEA (Complex-TEA-FU):** Una unidad especializada más compleja capaz de ejecutar una *ronda completa de TEA* (tanto las actualizaciones de v0 como de v1, y la actualización de sum) en un solo ciclo de instrucción (o en unos pocos ciclos pipelined). Esta unidad tendría acceso directo e interno a la Bóveda de Llaves.
- **Módulo de Bóveda de Llaves Segura con Gestión de Punteros:** Almacena las llaves y gestiona los punteros seguros. Incluye lógica para validar los punteros pasados por las instrucciones y recuperar la llave correspondiente internamente, asegurando un control de acceso estricto.
- Unidad de Control: Más compleja que en la Opción 1, responsable de orquestar las instrucciones TEA_ROUND de múltiples operaciones y de gestionar la validación de los punteros de llave y el acceso seguro a la bóveda.
  El flujo de datos implicaría que v0, v1 y sum fluyan desde los registros a la Complex-TEA-FU. El puntero de llave fluiría desde un registro a la Unidad de Control/Bóveda. El material de la llave se obtendría internamente de la Bóveda a la Complex-TEA-FU. Los resultados se escribirían de nuevo en los registros.

Justificación

- **Eficiencia:** Mayor paralelismo a nivel de instrucción; una sola instrucción realiza una cantidad significativa de trabajo criptográfico, lo que potencialmente reduce el número de instrucciones y los ciclos totales para las operaciones TEA.
- **Simplicidad del Software:** Simplifica el diseño del compilador/ensamblador para las operaciones TEA, ya que una sola instrucción se mapea a una ronda completa.
- **Costo (Área/Complejidad):** La Complex-TEA-FU sería más grande y compleja que las unidades granulares de la Opción 1, lo que podría aumentar el área y el esfuerzo de diseño. Esto aborda directamente el compromiso entre eficiencia y costo.[1]
- **Seguridad:** Mantiene un fuerte aislamiento de llaves, pero el concepto de "puntero" añade otra capa de abstracción, lo que podría dificultar el uso indebido del material de la llave por parte de exploits de software.

Comparación de Opciones de Solución

La selección de la propuesta final requiere una evaluación objetiva de las dos opciones de diseño presentadas, considerando criterios clave como la eficiencia, el costo del hardware, la complejidad del software y la seguridad. El diseño de una Arquitectura del Set de Instrucciones (ISA) implica inherentemente la navegación de objetivos en conflicto. Por ejemplo, optimizar el rendimiento puede aumentar la complejidad, mientras que priorizar la simplicidad podría sacrificar el rendimiento. Los diseñadores deben considerar cuidadosamente el dominio de la aplicación objetivo, los requisitos de rendimiento, las restricciones de energía y otros factores para lograr el equilibrio adecuado.

A continuación, se presenta una comparación detallada:

Criterio	Opción 1: Aceleración Granular de TEA y Acceso Directo a Bóveda	Opción 2: Aceleración de Ronda Completa de TEA y Bóveda con Punteros Seguros
Eficiencia	Permite pipelining más fino de sub-operaciones TEA individuales dentro de la Unidad Funcional. Requiere más instrucciones por ronda completa.	Potencialmente mayor paralelismo a nivel de instrucción, una sola instrucción realiza más trabajo. Reduce el conteo de instrucciones para TEA.
Costo de Hardware (Área/Complejidad)	La Unidad Funcional TEA (TEA-FU) es más simple por instrucción, lo que puede reducir la complejidad y el área.	La Unidad Funcional Compleja TEA (Complex-TEA-FU) es más grande y compleja, lo que aumenta el área y el esfuerzo de diseño.
Complejidad de Software	Requiere más instrucciones de ensamblador por ronda, lo que puede aumentar la complejidad del código. Ofrece	Simplifica la programación en ensamblador para operaciones TEA, ya que una sola instrucción mapea una ronda completa.

	flexibilidad para cálculos parciales.	
Seguridad	Fuerte aislamiento de llaves forzado por hardware, con acceso directo y controlado al material de la llave.	Mantiene un fuerte aislamiento de llaves; el concepto de "puntero" añade una capa de abstracción, lo que podría mejorar la seguridad por ofuscación y reducir la exposición directa de la llave al software.

Análisis de Compromisos (Trade-offs):

La elección entre estas opciones implica compromisos significativos. La Opción 1, con su enfoque granular, se alinea directamente con la advertencia del proyecto sobre la complejidad y el costo de una única instrucción de cifrado/descifrado completo. Al descomponer la operación de TEA en unidades más pequeñas, se reduce la complejidad de cada unidad funcional dedicada, lo que facilita el diseño y la implementación del hardware. Este enfoque también puede ofrecer una mayor flexibilidad para el pipelining interno de las operaciones criptográficas, lo que potencialmente conduce a una mayor eficiencia a nivel de microarquitectura, incluso si el número de instrucciones ejecutadas es mayor.

Por otro lado, la Opción 2 simplifica la tarea del programador al encapsular una ronda completa de TEA en una sola instrucción. Esto podría resultar en un código ensamblador más compacto y potencialmente un menor número de ciclos totales si la unidad funcional compleja puede ejecutar la ronda rápidamente. Sin embargo, la complejidad inherente de esta unidad funcional aumentaría los requisitos de área y el esfuerzo de diseño del hardware. La introducción de "punteros seguros" para las llaves, aunque añade una capa de abstracción de seguridad, también introduce una complejidad adicional en la lógica de gestión de la bóveda.

En cuanto a la seguridad, ambas opciones ofrecen un fuerte aislamiento de llaves, un requisito crítico del proyecto. La Opción 1 enfatiza el control directo y granular sobre las partes de la llave que se exponen a la unidad funcional, mientras que la Opción 2 introduce un nivel de indirección a través de punteros, lo que podría ser beneficioso para la gestión de políticas de uso de llaves, similar a los sistemas de capacidades.[16]

Selección de la Propuesta Final

Tras una evaluación objetiva de las opciones de solución, la **Opción de Solución 1: ISA con Aceleración Granular de TEA y Bóveda de Llaves con Acceso Directo por Instrucción** se selecciona como la propuesta final para este proyecto.

Esta decisión se basa en una serie de justificaciones que abordan directamente los requisitos y las limitaciones impuestas por la especificación del proyecto, buscando un equilibrio óptimo entre rendimiento, costo y seguridad.

1. **Adherencia a las Restricciones del Proyecto:** La especificación del proyecto establece explícitamente que "una sola instrucción para el cifrado/descifrado completo resultaría en un muy módulo complejo y costoso". La Opción 1, al proponer instrucciones granulares que aceleran *partes* de una ronda de TEA, cumple directamente con esta restricción. Este enfoque evita la creación de una unidad funcional monolítica y excesivamente compleja, lo que se alinea con la necesidad de equilibrar la eficiencia con la complejidad y el costo del hardware.
2. **Modularidad y Equilibrio en el Diseño de Hardware:** Al descomponer el algoritmo TEA en operaciones más pequeñas y dedicadas, la Opción 1 permite un diseño de hardware más modular. Cada instrucción granular corresponde a una unidad funcional más simple y manejable. Esto facilita la implementación, la depuración y la verificación del hardware, al tiempo que proporciona una aceleración efectiva para las operaciones criptográficas centrales.[1][2] Este enfoque demuestra una comprensión práctica de los compromisos de diseño en la arquitectura de computadores, donde la simplicidad de los componentes individuales puede conducir a una mayor robustez y eficiencia general.
3. **Seguridad Robusta y Control Granular de Llaves:** La Opción 1 mantiene el requisito fundamental de un fuerte aislamiento de llaves forzado por hardware, un pilar de la seguridad en este proyecto. El acceso directo y controlado a las partes de la llave desde la bóveda a la unidad funcional TEA, sin que las llaves transiten por registros de propósito general o memoria principal, proporciona una barrera de seguridad robusta. Este modelo de acceso directo y parcial para las llaves, inspirado en principios de diseño de bóvedas de llaves seguras [4], minimiza la superficie de ataque y reduce el riesgo de fugas de llaves a través de canales laterales, lo que se alinea con los principios de diseño de procesadores seguros.[6]
4. **Flexibilidad para Futuras Extensiones:** Las instrucciones granulares ofrecen una mayor flexibilidad. Si en el futuro se requirieran variaciones del algoritmo TEA o se necesitaran acelerar otros algoritmos criptográficos con operaciones similares

(como SHL, SHR, ADD, XOR), las unidades funcionales existentes podrían reutilizarse o adaptarse con mayor facilidad. Esta adaptabilidad es una ventaja en el contexto de un diseño de ISA personalizado.

5. **Rendimiento del sistema:** la incorporación de una microarquitectura con pipeline permite obtener una mayor eficiencia a la hora de ejecutar múltiples instrucciones de manera solapada, completando una por ciclo en estado estacionario. Está estrategia ofrece claras ventajas frente a arquitecturas uniciclo o multiciclo, al mejorar el throughput sin comprometer la simplicidad del diseño, además de que el uso de instrucciones granulares para TEA se adapta perfectamente a este diseño permitiendo acelerar partes del algoritmo sin necesidad de unidades funcionales más complejas.

En resumen, la Opción 1 no solo cumple con todos los requisitos funcionales y de seguridad del proyecto, sino que también lo hace de una manera que respeta las limitaciones prácticas de diseño de hardware. Proporciona una solución equilibrada que es eficiente, segura y factible de implementar dentro del alcance del proyecto.

El diseño y diagrama utilizado para el desarrollo del CPU seleccionado puede apreciarse en este vínculo.

Listado de funciones implementadas y comparación con ARM

El diseño del set de instrucciones desarrollados e implementación microarquitectónica toma inspiración parcial de arquitecturas existentes como ARMv4. A continuación se presenta un resumen de las funciones implementadas, clasificadas por categoría, seguidas de una comparación con ARM:

1. **Aritmética y lógica básica:** tomando como base los conceptos proporcionados por ARM para esta clase de instrucciones, se adaptaron funciones para permitir su uso en el decodificador diseñado de 64bits, además de que se utiliza una codificación personalizada en vez del formato condicional típico de ARM.
   - ADD, SUB, MUL, DIV, con versiones inmediatas e instrucciones con acarreo (ADC, SBC, etc.).
   - Lógica bit a bit (AND, ORR, EOR, BIC), con soporte para inmediatos.
   - Operaciones de comparación (CMP, CMPI, TST, TEQ, etc.).
2. **Acceso a memoria:** mientras que en ARM el acceso a regiones protegidas se realiza mediante control de privilegios, aquí el control está determinado por flags de

seguridad y el aislamiento físico de la bóveda de llaves, lo que añade un nivel de seguridad no presente en el set de instrucciones original.

- Instrucciones LDR, STR para acceso a memoria general y dinámica, con variantes byte (LDRB, STRB).
- Soporte explícito para acceso a memoria de llaves, contraseña y bóveda segura.

3. **TEA:** ARM como tal no incluye instrucciones criptográficas nativas, a diferencia del diseño propuesto donde se introducen instrucciones semánticamente ligadas a TEA, lo que optimiza específicamente el manejo de la aplicación de seguridad.
   - TEAENC #x, TEAD #x: ejecución parcial de rondas TEA.
   - Instrucciones dedicadas para mover y transformar componentes intermedios (LSL, LSR, ADDS, EOR, etc.).
   - Soporte para sumas y decrementos de ronda (MOVI w7, #32, SUBI w7, #1, etc.).
4. **Control de flujo y operación del sistema:** además de las instrucciones de las instrucciones de salto para alterar el flujo de ejecución del programa, se implementa un modelo de seguridad por sesión con flags, lo cual actúa como una capa adicional para la habilitación/deshabilitación de operaciones
   - Instrucciones de bifurcación (B, BEQ, BNE, etc.).
   - Instrucciones especiales (SWI, NOP).
   - Instrucciones de login y logout con control de flags y gestión de sesión (AUTHCMP, LOGOUT).

Conclusiones

El diseño de una Arquitectura del Set de Instrucciones (ISA) personalizada, como se ha explorado en este proyecto, es fundamental para optimizar el rendimiento y la seguridad en aplicaciones específicas, especialmente en dominios críticos como la ciberseguridad y los sistemas empotrados. La creciente demanda de protección de datos en dispositivos con recursos limitados, donde las arquitecturas RISC son predominantes, subraya la relevancia de integrar capacidades de seguridad directamente en el hardware del procesador.

El proceso de diseño ha puesto de manifiesto la necesidad de un análisis meticuloso de los requisitos, particularmente aquellos relacionados con la aceleración criptográfica y la gestión segura de llaves. La especificación explícita de evitar una única instrucción compleja para el cifrado completo de TEA guio la arquitectura hacia un enfoque de aceleración granular. Esta estrategia, que implica la creación de instrucciones personalizadas para partes específicas del algoritmo, no solo cumple con las restricciones de complejidad y costo del hardware, sino que también se alinea con las prácticas de la industria en el diseño de aceleradores criptográficos, como se observa en las extensiones de ISA para AES.

Asimismo, los estrictos requisitos para la bóveda de llaves, que prohíben el acceso tradicional a la memoria y la exposición de las llaves a registros de propósito general, impulsaron la concepción de un módulo de hardware dedicado y altamente aislado. Este enfoque garantiza que las llaves criptográficas permanezcan protegidas dentro del procesador, minimizando el riesgo de fugas y ataques. La implementación de un camino de datos seguro y directo desde la bóveda a las unidades funcionales es esencial para mantener la integridad de las llaves, lo que refuerza la seguridad general del sistema.

La selección de la Opción 1, con su aceleración granular de TEA y su modelo de acceso directo a la bóveda de llaves, representa una solución equilibrada. Esta propuesta logra un rendimiento eficiente para las operaciones de cifrado, mantiene la complejidad del hardware en un nivel manejable y proporciona una seguridad robusta para las llaves criptográficas. Este diseño no solo satisface los objetivos del proyecto, sino que también ofrece una plataforma sólida para futuras exploraciones en la integración de características de seguridad avanzadas en arquitecturas de procesadores personalizadas, contribuyendo al avance de la seguridad en sistemas embebidos y el Internet de las Cosas.

Por último, una vez tomada esta opción como nuestra propuesta final, podemos observar unos resultados realmente positivos ya que, hablando en términos de datos finales de lo

que puede llegar a ejecutar el CPU creado, se puede visualizar cuantitativamente esta ejecución. Como datos finales y que respaldan la elección de nuestra propuesta, nuestro CPU logra ejecutar alrededor de 3 280 748 de ciclos en cuestión de aproximadamente 236,489 segundos. En cuanto a performance finalmente, sería alrededor de 13873 ciclos por segundo. Estos datos nos demuestran que genuinamente se tomó una buena decisión en cuanto a la propuesta y se encuentra respaldada por muy buenos números incluso.

Bibliografía

1. TEA Algorithm - ScholarWorks, fecha de acceso: junio 13, 2025, https://scholarworks.calstate.edu/concern/projects/vd66w472p
2. A Detailed Overview of TEA-128 Encryption : Raku - MojoAuth, fecha de acceso: junio 13, 2025, https://mojoauth.com/encryption-decryption/tea-128-encryption--raku
3. AES instruction set - Wikipedia, fecha de acceso: junio 13, 2025, https://en.wikipedia.org/wiki/AES_instruction_set
4. KeyVisor – A Lightweight ISA Extension for Protected Key Handles with CPU-enforced Usage Policies - arXiv, fecha de acceso: junio 13, 2025, https://arxiv.org/html/2410.01777v1
5. KeyVisor – An ISA Extension for Protected Key Handles - arXiv, fecha de acceso: junio 13, 2025, https://arxiv.org/pdf/2410.01777
6. Principles of Secure Processor Architecture Design - Computer ..., fecha de acceso: junio 13, 2025, https://caslab.io/books/processor-security/Szefer_Sample.pdf
7. Principles of Secure Processor Architecture Design | Yale News, fecha de acceso: junio 13, 2025, https://news.yale.edu/2019/01/15/principles-secure-processor-architecture-design
8. Accelerating Homomorphic Encryption in RISC-V Architectures with Hardware Based Number Theory Transform - AFIT Scholar, fecha de acceso: junio 13, 2025, https://scholar.afit.edu/cgi/viewcontent.cgi?article=8784&context=etd
9. Optimised AES with RISC-V Vector Extensions - Radboud Repository, fecha de acceso: junio 13, 2025, https://repository.ubn.ru.nl/bitstream/handle/2066/306267/1/306267.pdf
10. Hardware-Efficient RISC-V Accelerator with Low-Latency AES Instruction Extension for IoT Security - arXiv, fecha de acceso: junio 13, 2025, https://arxiv.org/html/2505.11880v1
11. RISC-V Architecture: A Comprehensive Guide to the Open-Source ISA - NextPCB, fecha de acceso: junio 13, 2025, https://www.nextpcb.com/blog/risc-v-architecture
12. Hardware acceleration of TEA and XTEA algorithms on FPGA, GPU and multi-core processors - SlideShare, fecha de acceso: junio 13, 2025, https://www.slideshare.net/slideshow/fpga2013-poster-v1/16492803

13. High throughput implementations of cryptography algorithms on GPU and FPGA | Request PDF - ResearchGate, fecha de acceso: junio 13, 2025, https://www.researchgate.net/publication/261298842_High_throughput_implementations_of_cryptography_algorithms_on_GPU_and_FPGA
14. Challenges And Limitations In Instruction Set Architecture Design ..., fecha de acceso: junio 13, 2025, https://fastercapital.com/topics/challenges-and-limitations-in-instruction-set-architecture-design.html/1