Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores
(Computer Engineering School)

Programa de Licenciatura en Ingeniería en Computadores
(Licentiate Degree Program in Computer Engineering)

Curso: CE-4301 Arquitectura de Computadores I
(Course: CE-4301 Computer Architecture I)

Especificación Proyecto Grupal I
(Group Project I Specification)

Profesor:
(Professor)

Prof. Dr.-Ing. Jeferson González Gómez

Fecha de entrega: 6 de Junio, 2025
(Due Date: June 6th, 2025)

Proyecto Grupal 1. Arquitectura del Set de Instrucciones (ISA) Específica tipo RISC para Aplicaciones de Seguridad Informática

Grupo: 4 estudiantes

1. Objetivo

Mediante el desarrollo de este proyecto, el estudiante aplicará los conceptos de arquitectura de computadores en el diseño, y posterior implementación en un modelo de software, de una arquitectura del set de instrucciones (ISA)tipo RISC específica y propia de cada grupo de trabajo para un conjunto de aplicaciones relacionadas con la seguridad del hardware y la ciberseguridad.

2. Descripción General

En el transcurso de las dos décadas pasadas, las arquitecturas RISC han crecido significativamente en popularidad debido principalmente a su eficiencia. El mercado de los sistemas empotrados, en general, y en específico los dispositivos móviles actuales, se encuentra dominado por procesadores RISC y computadores basados en Sistemas en Chip (SoC) o microcontroladores (MCU), cuyo núcleo, es también un procesador RISC. Es por esta razón que, en el marco de la arquitectura de computadores moderna, es fundamental conocer las características de este tipo de arquitectura, aplicable directamente a tecnologías y tendencias de punta como lo son el Internet de las cosas (IoT) y los sistemas empotrados en general.

Con el auge en la popularidad de dichos sistemas, así como el incremento en la complejidad de los mismos, la seguridad de las aplicaciones y la información dentro de estos sistemas emergentes se ha convertido en un tema de interés global. Con el objetivo de promover ambientes seguros de ejecución para aplicaciones, los fabricantes de procesadores han optado por incluir extensiones específicas de seguridad en sus arquitecturas (e.g., ARM TrustZone, Intel SGX), que permiten la inclusión de diferentes primitivas de seguridad para un rango variado de aplicaciones de manera eficiente y con soporte de hardware.

En este proyecto, se dará una introducción a este tipo de sistemas, por medio del diseño de una ISA con soporte nativo para intrínsecas de seguridad relacionadas con cifrado eficiente y almacenamiento seguro de llaves criptográficas.

3. Especificación

3.1. Aplicaciones de Seguridad

En este proyecto se deberá dar soporte de hardware a instrucciones específicas para el manejo eficiente de operaciones relacionadas con la seguridad de la información. Específicamente, se deberá dar soporte a dos áreas: cifrado eficiente de datos y almacenamiento seguro de llaves criptográficas. En esta sección se detallan dichas áreas. Los requisitos de arquitectura que se derivan de estas áreas se detallan más adelante (Sec. 3.2)

```
// Cifrado de 64 bits usando TEA
void tea_encrypt(uint32_t v[2], const uint32_t key[4]) {
    uint32_t v0 = v[0], v1 = v[1];
    uint32_t sum = 0;

    for (int i = 0; i < 32; ++i) {  // 32 rondas
        sum += DELTA;
        v0 += ((v1 << 4) + key[0]) ^ (v1 + sum) ^ ((v1 >> 5) + key[1]);
        v1 += ((v0 << 4) + key[2]) ^ (v0 + sum) ^ ((v0 >> 5) + key[3]);
    }

    v[0] = v0;
    v[1] = v1;
}

// Descifrado de 64 bits usando TEA
void tea_decrypt(uint32_t v[2], const uint32_t key[4]) {
    uint32_t v0 = v[0], v1 = v[1];
    uint32_t sum = DELTA * 32;

    for (int i = 0; i < 32; ++i) {  // 32 rondas
        v1 -= ((v0 << 4) + key[2]) ^ (v0 + sum) ^ ((v0 >> 5) + key[3]);
        v0 -= ((v1 << 4) + key[0]) ^ (v1 + sum) ^ ((v1 >> 5) + key[1]);
        sum -= DELTA;
    }

    v[0] = v0;
    v[1] = v1;
}
```

Figura 1: Funciones de cifrado y descifrado de referencia usando TEA.

3.1.1. Cifrado eficiente (TEA)

Los algoritmos de cifrado aplican operaciones matemáticas con el fin de transformar datos legibles (típicamente información privada o sensible) en un formato ilegible (cifrado) con el fin de proteger su confidencialidad. Tradicionalmente, el proceso de cifrado es reversible y requiere de al menos una llave criptográfica para cifrar y descifrar la información.

En este proyecto se deberá utilizar el algoritmo de cifrado de bloque llamado *Tiny Encryption Algorithm* (TEA) [1]. Dicho algoritmo permite cifrar datos con baja complejidad computacional de manera eficiente. En la Fig. 1 se presentan las funciones de cifrado y descifrado de referencia utilizando el algoritmo TEA. Para efectos de implementación en este proyecto, el algoritmo deberá emplear una llave de 128 bits, y efectuar **32 rondas de cifrado** por bloque, **para un bloque de entrada de 64 bits** (o dos bloques de 32 bits), tal como se muestra en el código de referencia. Para efectos de validación, se deberá utilizar una constante DELTA (o *magic constant* en el algoritmo) con un valor de **0x9e3779b9**.

3.1.2. Almacenamiento seguro de llaves (bóveda)

Como se mencionó anteriormente, los algoritmos de cifrado requieren llaves criptográficas que son típicamente privadas (para el caso de los algoritmos de cifrado simétrico). Dado que la eficacia y fortaleza del algoritmo de cifrado dependen en gran medida de la llave utilizada, es fundamental que su manejo y almacenamiento sea seguro.

Uno de los mecanismos empleados para asegurarse de que las llaves privadas no sean atacadas (por ejemplo, si un atacante logra tener acceso al sistema) es no mantenerlas en archivos o memoria general, sino **almacenarlas directamente en el hardware** utilizando una *raíz de confianza* (*Root of Trust - RoT*), inaccesible para un usuario común del sistema, pero accesible para las aplicaciones seguras del sistema (e.g., cifrado).
Para este proyecto se deberá modelar una raíz de confianza de forma de una bóveda de llaves (memoria segura) que permita almacenar **al menos 4 llaves de 128 bits cada una**. Dicha bóveda deberá ser accesible directamente/solamente por el CPU, por medio de instrucciones del set específicas para escribir u operar con los datos de la misma. La bóveda NO debe poder ser accedida como una memoria tradicional.

3.2. Requisitos Específicos de las Aplicaciones de Seguridad para la ISA

Para la implementación eficiente de las aplicaciones de seguridad mencionadas arriba, se deberán considerar los siguientes requerimientos para el diseño del set de instrucciones:

3.2.1. Bóveda de llaves

- Para la bóveda de llaves se deberá contar una o más instrucciones específicas para almacenar llaves de 128 bits. Ya que sólo se permite almacenar 4 llaves, se deberá considerar esta limitante en el diseño de la(s) instrucciones.
- Toda operación criptográfica que involucre llaves deberá utilizaras desde la bóveda.
- Los datos almacenados en la bóveda NO podrán ser escritos en registros del CPU o memoria general del sistema.
- Solamente las instrucciones que operen directamente sea de manera parcial o total con las llaves pueden acceder a la bóveda como operando fuente. Tome en cuenta la aplicación de cifrado (TEA) como referencia para este requerimiento.

3.2.2. Cifrado TEA

- El cifrado debe realizarse de manera eficiente. Deberá incluir en el set **al menos dos** instrucciones específicas para acelerar el algoritmo.
- Deberá considerarse una balance entre eficiencia y posible costo (área, complejidad) en el diseño de las instrucciones relacionadas con el cifrado. Por ejemplo, una sóla instrucción para el cifrado/descifrado completo resultaría en un muy módulo complejo y costoso.

3.3. Requisitos Generales de la Arquitectura del Set de Instrucciones

1. Debe diseñar un conjunto de instrucciones y arquitectura que permita solucionar el problema planteado, considerando detalles como:

   a) Modos de direccionamiento.

   b) Tamaño y tipo de datos.

   c) Tipo y sintaxis de las instrucciones.

   d) Registros disponibles y sus nombres.

   e) Codificación y descripción funcional de las instrucciones

   Tome en cuenta que estos detalles deben ser justificados desde el punto de vista de diseño (complejidad, costo, área, recursos disponibles). Las/los estudiantes deben realizar un análisis en un software de alto nivel para encontrar los valores idóneos.

2. Las instrucciones a desarrollar son libres así como el tipo de datos. Aunque no hay un límite en cuanto a la cantidad de instrucciones, es importante que provea al menos instrucciones para control de flujo, operaciones aritméticas-lógicas, acceso a memoria.

3. La arquitectura debe ser personalizada para las aplicaciones descritas y debe ser diseñada completamente por los estudiantes, **no se aceptarán ISAs basados en arquitecturas existentes (e.g., ARM, x86, RISC-V, otros)**. Debe justificar cada característica del mismo.

4. Los productos finales de esta etapa son los siguientes:

   a) Hoja de referencia rápida de instrucciones (*Instruction reference sheet* o *green card*).

   b) Documentación de aspectos de diseño de set de instrucciones incluyendo argumentación y resultados de scripts usados para justificar las características de la ISA, como parte del **Documento de diseño**, en la sección correspondiente a la ISA.

3.4. Requisitos de Implementación y Validación de la Arquitectura

3.4.1. Implementación

La implementación de la arquitectura del conjunto de instrucciones se debe realizar por medio de un modelo de software. Dicho modelo deberá ser ejecutable y funcionalmente equivalente a un procesador. Para esto, deberán modelarse adecuadamente los componentes internos del CPU (banco de registros, unidades funcionales, unidad de control, etc.), así como las interconexiones de los mismos. Además, se deberá implementar un modelo de memoria(s) que permita la carga y escritura de datos e instrucciones.

El modelo de software del procesador que implementa el set de instrucciones propio deberá brindar métricas de la ejecución de los programas (como mínimo, cantidad de ciclos). Adicionalmente, la simulación del procesador deberá permitir en todo momento la visualización de

elementos claves, como los registros internos del CPU, así como la bóveda de llaves criptográficas.

A pesar de tratarse de una solución de software, la eficiencia de la arquitectura y microarquitectura implementada deberá ser considerada en el diseño.
Cada grupo deberá realizar una aplicación tipo compilador/ensamblador que permita traducir las instrucciones del ISA a binario, con la finalidad de ser cargada al modelo de la memoria y ejecutarlo en el procesador.

El lenguaje de programación a elegir queda a criterio de cada grupo de trabajo. Los productos finales de esta etapa son:

1. El código fuente (repo) y comandos/scripts de compilación y ejecución del modelo funcional de software del CPU, así como el compilador/ensamblador.

2. Un diagrama de bloques de la microarquitectura y descripción de las interacciones entre ellos, incluido en el **Documento de diseño**, en la sección correspondiente a la organización.

3. Programa (código fuente en ensamblador) que permita evaluar todas las instrucciones incorporadas en el set, a diseñar por cada grupo de trabajo.

4. Documentación de diseño de todo el software del sistema. Incluido en el **Documento de diseño**, en la sección correspondiente al modelado de la organización/microarquitectura.

3.4.2. Validación en Aplicación de Seguridad

Para validar el diseño en cuanto a la inclusión de las instrucciones ara el almacenamiento seguro de llaves, así como el cifrado/descifrado eficiente de datos, a cada grupo se le suministrarán dos archivos de prueba para cifrar y descifrar según se describe a continuación:

- Para la primera prueba, el profesor brindará un archivo regular de tamaño fijo (1KB), y una llave única de 128 bits. Cad grupo deberá crear un programa que aplique el cifrado TEA utilizando su arquitectura/modelo de procesador para producir un archivo (real) cifrado. Posteriormente, cada grupo deberá calcular un hash MD5 (con Linux/online) sobre el archivo. Para verificar el proceso, el profesor comparará el hash obtenido con el valor esperado.

- Para la segunda prueba, el profesor brindará un archivo cifrado de tamaño fijo, así como otra llave única de 128 bits. Cada grupo deberá crear un programa que descifre el archivo por medio del algoritmo TEA utilizando su arquitectura/modelo de procesador para producir un archivo (real) cifrado. Posteriormente, el grupo deberá mostrar al profesor el contenido privado del archivo descifrado para verificar el proceso.

Nota: Todo proceso de cifrado/descifrado con llaves deberá utilizar la bóveda.
Salidas esperadas de esta etapa: código(s) fuente(s) en repositorio correspondiente.

4. Evaluación del proyecto

La evaluación del proyecto se da bajos los siguientes rubros contra rúbrica correspondiente:

- Presentación proyecto funcional (75 %): La defensa se realizará de **forma presencial** en un horario a definir (preferiblemente en horas de clase). La defensa se debe realizar en un espacio de 25 minutos, aproximadamente. Deberá mostrarse el sistema en funcionamiento, así como la ejecución de todas las instrucciones y de la validación con la aplicación de seguridad.

  Los entregables adicionales que se revisarán **durante** la defensa son los siguientes:

  1. Arquitectura:
     a) *Instruction reference sheet* o *green sheet*.
     b) Scripts usados para justificar las características del ISA.
  2. Microarquitectura:
     a) Diagrama de bloques de la microarquitectura.
     b) Estado de los registros y bóveda durante la ejecución
  3. Otro Software:
     a) Compilador/Ensamblador usado.
     b) Código fuente en lenguaje ensamblador para la arquitectura diseñada.
  4. Validación Seguridad
     a) Validación cifrado
     b) Validación descifrado

- Documentación de diseño (25 %): La documentación del diseño deberá contener las siguientes secciones:

  1. Elaboración de opciones de solución al problema: Para el problema planteado deberán documentarse al menos dos opciones de solución. Cada solución deberá ser acompañada de algún tipo de diagrama. **Estas opciones de solución no deben ser fácilmente descartables y deben llevar un análisis objetivo con base en criterios técnicos o teóricos.** Al ser un curso de Arquitectura de Computadores, las opciones deben ser esencialmente basadas en el ISA y su correspondiente organización.
  2. Comparación de opciones de solución: Se deberán comparar explícitamente las opciones de solución, de acuerdo con los requerimientos y otros aspectos aplicables de eficiencia, ética, de salud, seguridad, ambientales, económicos, culturales, sociales y de estándares aplicables.
  3. Selección de la propuesta final: Se deberá evaluar de forma objetiva, válida y precisa las soluciones planteadas al problema y escoger una solución final.

Referencias

[1] David J Wheeler and Roger M Needham. Tea, a tiny encryption algorithm. In *Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings 2*, pages 363–366. Springer, 1995.