Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Proyecto Integrador II

Equipo #1

Nombre	Matricula	Carrera
Jorge Alberto Morales Reyes	1895340	ITS
Integrante 2	Matricula 2	x
Integrante 3	Matricula 3	x
Integrante 4	Matricula 4	x
Integrante 5	Matricula 5	x

Ciudad Universitaria, San Nicolás de los Garza, Nuevo León

2026-01-01

Indice

1. Definicion del Proyecto

1.1. Nombre del Proyecto

[NOMBRE DE PROYECTO]

Razones porque?

1.2. Descripcion del Proyecto

El proyecto se concibe como un sistema compuesto por cuatro elementos principales: una **máquina virtual** basada en una arquitectura ARM32 simplificada, un **assembler** encargado de traducir código ensamblador a una representación binaria, una **interfaz gráfica** que permite la visualización y control del sistema, y una documentación técnica presentada en forma de manual. Estos componentes trabajan de manera conjunta para permitir la ejecución y el análisis de programas a bajo nivel en un entorno controlado, visual e interactivo.

1.3. Problematica

El aprendizaje de la arquitectura de computadoras, y en particular de arquitecturas ampliamente utilizadas como ARM32, presenta diversas dificultades en el ámbito educativo. Una de las principales problemáticas es la limitada disponibilidad de hardware adecuado para la enseñanza. Aunque existen dispositivos basados en ARM, como microcontroladores o placas de desarrollo, su acceso no siempre es inmediato, su configuración puede resultar compleja y, en muchos casos, su uso requiere conocimientos previos que van más allá de los objetivos de cursos introductorios.

Adicionalmente, el estudio directo sobre hardware real suele desplazar el enfoque del aprendizaje hacia detalles específicos de implementación, como configuraciones del sistema, periféricos o particularidades del entorno de ejecución. Esto puede dificultar que los estudiantes comprendan la visión general de la arquitectura de un CPU, es decir, cómo interactúan los registros, la memoria, el ciclo de ejecución y el conjunto de instrucciones, independientemente de una plataforma física concreta.

Desde la perspectiva de un tutor o docente, estas dificultades representan un reto importante en el proceso de enseñanza de la arquitectura de computadoras. En consecuencia, el tutor se enfrenta a la necesidad de contar con una herramienta que permita aislar los conceptos esenciales de la arquitectura de procesadores, facilite la experimentación controlada y sirva como apoyo didáctico tanto en el aula como en el estudio autónomo del estudiante. Estos son los problemas educativos que el presente proyecto busca abordar.

1.4. Propuesta

Para atender las necesidades identificadas desde la perspectiva docente, se propone el desarrollo de una máquina virtual educativa basada en una versión simplificada de la arquitectura ARM32, acompañada de un assembler propio y de documentación técnica diseñada como un manual de apoyo didáctico. Esta propuesta busca ofrecer una herramienta que permita enseñar los principios fundamentales de la arquitectura de un CPU sin depender de hardware físico ni de una implementación completa y compleja de ARM.

Como complemento, se desarrollará un assembler propio encargado de traducir programas escritos en lenguaje ensamblador a su representación binaria. Esto permitirá a los estudiantes comprender la relación entre la sintaxis del ensamblador, el formato de las instrucciones y su interpretación por parte del procesador, reforzando así la conexión entre el software de bajo nivel y la arquitectura subyacente.

Un componente central de la propuesta es la elaboración de documentación técnica clara y estructurada, presentada en forma de manual. Dicho manual describirá la arquitectura de la máquina virtual, el conjunto de instrucciones soportadas, el formato de las mismas, el funcionamiento del assembler

y ejemplos prácticos de uso. De esta manera, el proyecto no solo proporcionará una implementación funcional, sino también un recurso educativo reutilizable por tutores y estudiantes.

Finalmente, el proyecto será implementado en el lenguaje de programación Go, con el objetivo de garantizar su portabilidad y facilitar su ejecución en distintos sistemas operativos. Esto permitirá que la herramienta pueda ser utilizada en diversos entornos educativos sin requerir configuraciones complejas ni dependencias de hardware específico.

1.5. Cronograma

Actividades	*Enero 22-31*	*Febrero 1-15*	*Febrero 16-31*	*Marzo 1-15*	*Marzo 16-31*	*Abril 1-15*	*Abril 16-31*
Analisis y diseño							
Desarollo del assembler							
Implementacion de la maquina virtual							
Elaboracion de la documentacion							
Elaboracionde integracion y pruebas							
Refinamiento							

1.5.1. Análisis y Diseño

En esta fase se realizará el análisis conceptual del proyecto, definiendo la arquitectura general de la máquina virtual ARM32 simplificada. Se establecerá el modelo de registros, el esquema de memoria, el conjunto inicial de instrucciones y el formato básico de las mismas. Asimismo, se definirá la sintaxis del lenguaje ensamblador y la estructura general de la documentación.

1.5.2. Implementación de la máquina virtual

Durante esta etapa se desarrollará la máquina virtual, implementando el ciclo de ejecución (fetch–decode–execute), el manejo de registros de 32 bits, la memoria lineal compartida y los flags de estado. El objetivo es obtener una versión funcional que permite ejecutar instrucciones básicas y sirva como base para futuras mejoras.

1.5.3. Desarrollo del assembler

En esta fase se implementará el assembler encargado de traducir programas escritos en lenguaje ensamblador a palabras binarias de 32 bits compatibles con la máquina virtual. Se buscará mantener coherencia entre el formato de las instrucciones y su interpretación por la máquina virtual, permitiendo modificaciones en ambos componentes cuando sea necesario.

1.5.4. Integración y pruebas

La máquina virtual y el assembler serán integrados y evaluados mediante programas de prueba. Esta fase estará orientada a verificar el correcto funcionamiento del sistema, identificar errores y validar que el comportamiento de la máquina virtual sea coherente con el diseño propuesto.

1.5.5. Elaboración de la documentació

Se elaborará la documentación técnica en forma de manual, describiendo la arquitectura del sistema, el conjunto de instrucciones, el funcionamiento del assembler y ejemplos prácticos de uso. La documentación se desarrollará de manera paralela al proyecto y se actualizará conforme se realicen cambios en el diseño o la implementación.

1.5.6. Refinamiento

En la fase final se realizaron ajustes y mejoras al sistema en su conjunto, tanto a nivel de implementación como de documentación. Esta etapa permitirá optimizar decisiones de diseño, mejorar la claridad del manual y consolidar el proyecto como una herramienta educativa coherente y funcional.

2. Diseño y Arquitectura del Proyecto

2.1. Arquitectura General del Sistema

La arquitectura general del proyecto se organiza como una cadena de etapas claramente definidas, cuyo objetivo es facilitar la comprensión del flujo de ejecución de un programa desde su escritura hasta su interpretación por la máquina virtual.

El flujo del sistema comienza con la escritura de un programa en lenguaje ensamblador utilizando un editor de texto de preferencia del usuario. Este programa es posteriormente procesado por el ensamblador (assembler), el cual analiza las instrucciones simbólicas y genera como resultado una secuencia de palabras binarias de 32 bits, coherentes con el modelo de arquitectura definido para el sistema.

Las palabras binarias generadas por el ensamblador son cargadas en la memoria de la máquina virtual, donde representan tanto las instrucciones como los datos, Una vez cargado el programa, la máquina virtual inicia su ejecución aplicando el ciclo clásico de fetch–decode–execute, en el cual se obtiene la instrucción desde memoria, se decodifica su significado y se ejecuta la operación correspondiente sobre los registros y la memoria del sistema.

Con el objetivo de reforzar el carácter educativo del proyecto, se incorpora una interfaz gráfica de usuario (GUI) que permite visualizar de manera clara el estado interno de la máquina virtual durante la ejecución del programa. A través de esta interfaz es posible observar elementos como el contenido de los registros, el estado de la memoria y la instrucción actualmente en ejecución, facilitando el análisis del comportamiento del sistema paso a paso.

Esta arquitectura busca priorizar la claridad conceptual y la trazabilidad del proceso de ejecución, permitiendo al usuario comprender la relación entre el código ensamblador, su representación binaria y su efecto directo sobre el estado interno de la máquina virtual.

2.2. Arquitectura del Ensamblador

El ensamblador constituye el primer componente funcional del sistema y es el encargado de transformar el programa escrito en lenguaje ensamblador en una representación binaria ejecutable por la máquina virtual. Su diseño se enfoca en la claridad del proceso de traducción más que en la optimización extrema, con fines principalmente educativos.

2.2.1. Syntaxis General

La sintaxis del assembler está basada en convenciones utilizadas en sistemas reales. Cada línea puede contener una instrucción, una etiqueta o un comentario

2.2.1.1. Commentarios

```
// single line comments
;  single line comments
/* Multiline
 * comments
 */
```

2.2.1.2. Etiquetas (Lables)

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

2.2.1.3. Valores inmediatios

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

2.2.2. Ensamblador de Dos Pasadas

El ensamblador implementado en este proyecto sigue un modelo de dos pasadas, una estrategia clásica que permite resolver referencias simbólicas y generar código binario correcto sin incrementar innecesariamente la complejidad del sistema.

Durante la primera pasada, el ensamblador recorre el programa fuente con el objetivo de analizar su estructura general y construir una tabla de símbolos. En esta etapa se identifican etiquetas, se asignan direcciones de memoria a cada instrucción y se valida la organización básica del código. No se genera código binario definitivo en esta fase; en su lugar, se recopila la información necesaria para la traducción posterior.

En la segunda pasada, el ensamblador vuelve a procesar el programa fuente, utilizando la tabla de símbolos generada previamente para resolver referencias a etiquetas y direcciones. En esta etapa se lleva a cabo la traducción completa de las instrucciones a palabras binarias de 32 bits, ya con todos los operandos resueltos y codificados según el formato de instrucción definido por la arquitectura del sistema.

2.3. Arquitectura de la maquina virtual

2.3.1. Arquitectura Von Neumann

La arquitectura Von Neumann se caracteriza por utilizar un único espacio de memoria compartido tanto para las instrucciones como para los datos. En este modelo, el procesador accede a la memoria a través de un solo bus, por el cual se transfieren indistintamente instrucciones y datos.

Esta unificación simplifica el diseño del sistema, ya que no es necesario distinguir entre distintos tipos de memoria ni entre diferentes rutas de acceso. No obstante, introduce una limitación conocida como el cuello de botella de Von Neumann, debido a que el procesador no puede acceder simultáneamente a instrucciones y datos, lo que puede afectar el rendimiento.

A pesar de esta limitación, la arquitectura Von Neumann es ampliamente utilizada en sistemas educativos y en modelos conceptuales por su simplicidad y claridad estructural.

2.3.2. Arquitectura Harvard

La arquitectura Harvard se distingue por mantener memorias separadas para las instrucciones y los datos, cada una con su propio bus de acceso. Esta separación permite que el procesador obtenga una instrucción y acceda a datos al mismo tiempo, mejorando el rendimiento del sistema.

Este modelo reduce el cuello de botella presente en la arquitectura Von Neumann y permite optimizaciones como diferentes tamaños de palabra o tecnologías de memoria para instrucciones y datos. Sin embargo, esta separación incrementa la complejidad del diseño del hardware y del sistema en general.

2.3.3. Arquitectura de [NOMBRE PROYECTO]

Para este proyecto se decidio utilizar la arquitectura de X

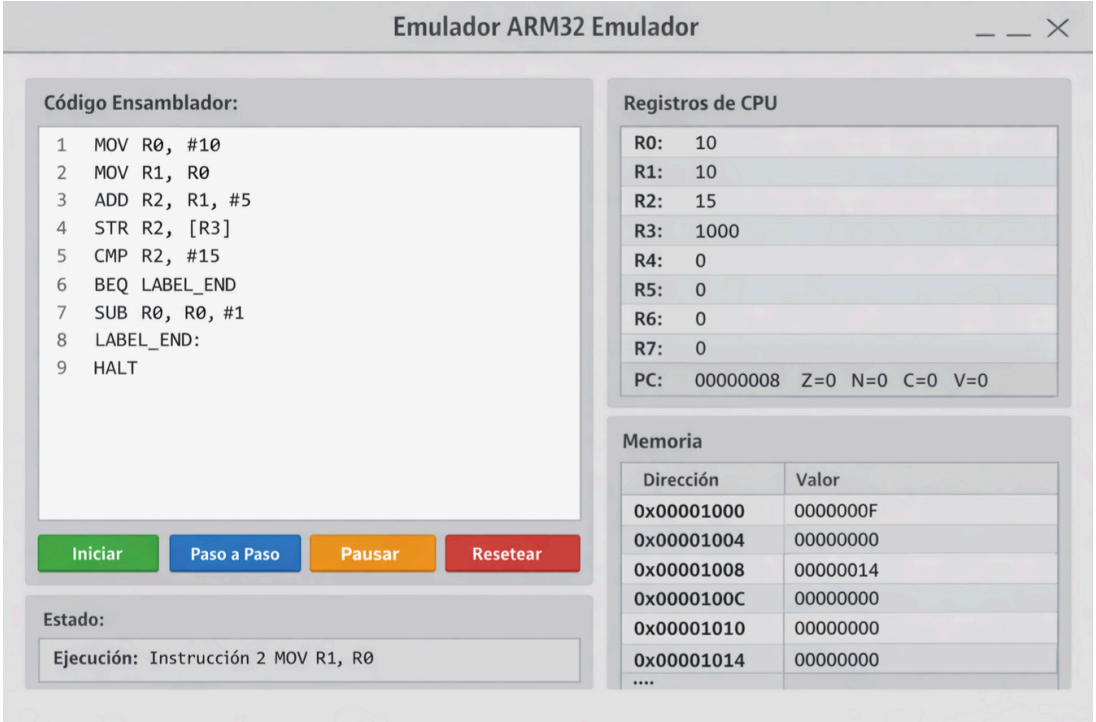
2.4. La Interfaz Gráfica de Usuario (GUI)

El objetivo principal de esta interfaz es facilitar la comprensión del funcionamiento interno de una arquitectura de CPU de forma visual e interactiva, especialmente con fines educativos.

La interfaz mostrará de manera clara el estado de los registros del procesador, permitiendo al estudiante observar cómo cambian sus valores conforme se ejecutan las instrucciones. Esto es especialmente importante para entender conceptos como la transferencia de datos, operaciones aritméticas y el uso de registros generales.

Adicionalmente, la GUI incluirá una vista de la memoria, ubicada en un panel lateral, donde se podrá observar el contenido de direcciones específicas. Esto permitirá relacionar directamente las instrucciones del programa con su efecto sobre la memoria, reforzando conceptos como direccionamiento, carga y almacenamiento de datos.

Un elemento central de la interfaz será la ejecución paso a paso del programa. Este modo permitirá avanzar instrucción por instrucción, actualizando en tiempo real los registros, la memoria y el contador de programa. De esta manera, el estudiante podrá analizar con detalle el efecto individual de cada instrucción, lo cual resulta especialmente útil para el aprendizaje y la depuración. Además del modo paso a paso, la interfaz permitirá la ejecución continua del programa, manteniendo siempre sincronizada la visualización del estado interno de la máquina virtual. En conjunto, la GUI no solo funcionará como un medio de ejecución, sino como una herramienta didáctica que conecta el código ensamblador con el comportamiento interno de la CPU.



Prototipo conceptual de la interfaz gráfica del sistema

2.5. Analisis Detallado

La arquitectura del proyecto se basa en una máquina virtual que modela una versión simplificada de la arquitectura ARM32, diseñada específicamente con fines educativos. El objetivo no es reproducir una implementación completa ni compatible a nivel binario con ARM real, sino capturar los principios fundamentales del diseño de un CPU.

2.5.1. Modelo de Central Processing Unit

La máquina virtual implementa un procesador con las siguientes características

- Arquitectura de 32 bits, donde tanto las instrucciones como los datos se representan como palabras de 32 bits.
- Habrá 13 registros de propósito general
- Registros especiales, para **Stack Pointer**, **Link Register** y **Program Counter**
- Registro de banderas incluye indicadores fundamentales
  - **Zero (Z)**: se activa cuando el resultado de una operación es igual a cero.
  - **Negative (N)**: se activa cuando el bit más significativo del resultado es uno, indicando un valor negativo en representación con signo.
  - **Carry (C)**: se activa cuando ocurre un acarreo fuera del bit más significativo en una operación aritmética, o un préstamo en el caso de una resta.
  - **Overflow (V)**: se activa cuando el resultado de una operación con signo excede el rango representable, indicando un desbordamiento aritmético.

2.5.2. Memoria

El sistema cuenta con **CUANTA MEMORIA?**

QUE TIPO DE ARQUITECTURA DECIDIMOS? HARDVARD/NUEMANN????

2.5.3. Conjunto de Instrucciones (ISA)

El proyecto define un Instruction Set Architecture (ISA) reducido, inspirado en ARM32. Se implementará un subconjunto de instrucciones básicas, tales como:

- Instrucciones de movimiento de datos
- Operaciones aritméticas y lógicas
- Instrucciones de comparación
- Saltos condicionales y no condicionales
- Instrucciones de carga y almacenamiento

Cada instrucción será codificada explícitamente en formato binario, y la máquina virtual se encargará de decodificar y reinterpretar esos bits como instrucciones ARM simplificadas. Esto permite estudiar directamente el proceso de fetch–decode–execute.

Instrucion	Descripcion	Ejemplo
mov	Copia el valor de un registro a otro registro	`mov Rd, Rs`
mov	Copia un numero a un registro	`mov Rd, #42`
add	Suma valor de dos registro a registro resultante	`add Rd, Rs1, Rs2`
add	Suma registro mas a valor, y lo guarda en registro	`add Rd, Rs1, #42`