Propuesta de lenguaje ensamblador para Te-La-Choco

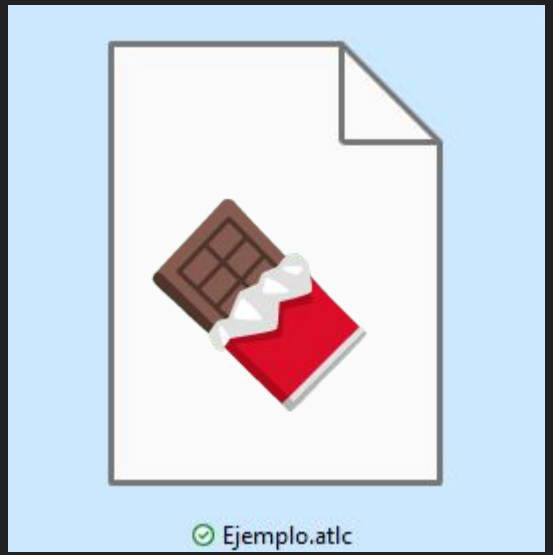
Rodrigo Caminero, Rodrigo Moreno

Índice de Contenidos

Introducción a ATLC

ATLC (Assembly Te-La-Choco) es un lenguaje ensamblador, cuya sintaxis está basada en el lenguaje **Te-La-Choco.**

El propósito de este lenguaje surge como necesidad para poder ejecutar instrucciones de bajo nivel empleando el lenguaje **Te-La-Choco**, permitiendo una interacción directa con el hardware y un control preciso de recursos del sistema.

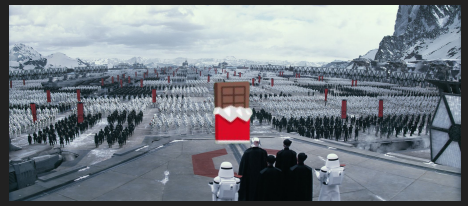


Objetivos de ATLC

Los objetivos principales que nos han llevado a desarrollar ATLC son los siguientes:

👉 Incorporar Te-La-Choco como lenguaje imperativo

👉 Simular una comunicación directa con el Hardware



👉 Simplicidad a la hora de desarrollar un lenguaje de alto nivel

👉 Adquirir mayor conocimiento sobre el funcionamiento de lenguajes de bajo nivel

Estructuras Empleadas

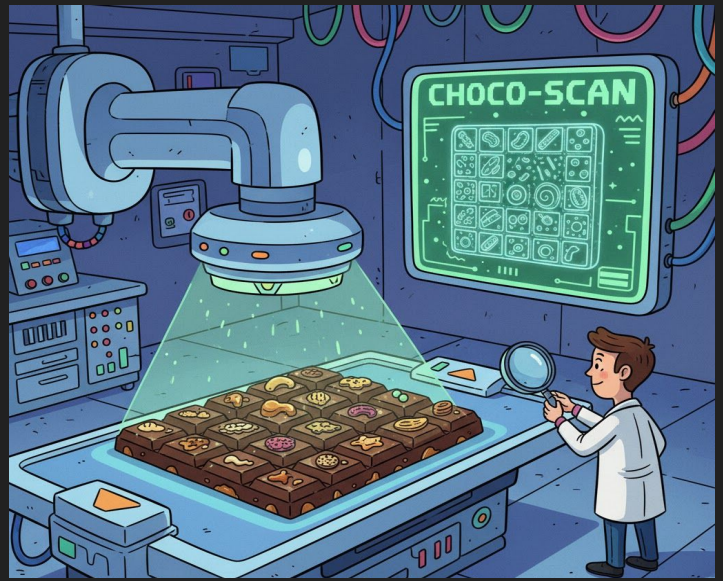
La estructura que se ha empleado en ATLC es la siguiente:

👉 **Mapa de registros**: Objeto JSON que casteamos a un diccionario en Python para poder ejecutar instrucciones. Tras la realización de las operaciones, se reescribirá el JSON, almacenando los valores del registro

👉 **Stack:** Empleamos el Stack para realizar operaciones aritmético-lógicas, simulando la ALU de un procesador

👉 **Queue:** Empleamos una cola básica para almacenar las instrucciones, con la finalidad de analizar y procesar las instrucciones de forma secuencial.

Funcionamiento Interno



Registros

Objetivo: Emplear pequeños bloques de memoria para simplificar el proceso de ejecución del programa.

Tipos de registros definidos:

👉 **Registros Standart**: Estos registros se acceden de forma directa por el programador, en función de la complejidad del programa.

👉 **Registros Temporales**: Estos registros están reservados ÚNICAMENTE para el procesador, se emplean principalmente para procesar operaciones Aritmético-Lógicas.

👉 **OPERATION:** Sirve para almacenar el tipo de operación que se va a ejecutar. No puede modificarse por el usuario.

AVISO: Los registros no se pueden sobrescribir.

Operaciones con Registros

Las operaciones que hemos definido con registros, son las siguientes

👉 **PR (Push Register)** -> Almacenamos los valores dentro del registro.

Ej: PR R1 4 -> Almacenamos el valor 4 en el primer registro

👉 **FR (Free Register)** -> Limpiamos el registro.

Ej: FR R1 -> Liberamos el primer registro

NOTA: Es muy importante que limpiemos los registros standart, pues no se limpian de forma automática

Lógica de registros

Flujo de ejecución del programa

1. Imprimimos un mensaje de presentación inicial
2. Se solicita el archivo
3. Vemos si es correcto el archivo
4. Se convierte el archivo a un texto plano
5. Mostramos la cola de instrucciones
6. inicializamos la VM y ejecutamos el programa

Operaciones Aritmético - Lógica

Objetivo: Optimizar el procesamiento de cálculos mediante el uso de bloques de memoria especializados.

Operaciones Soportadas: Suma, Resta, Multiplicación, División (Enteros), AND, OR, NOT, XOR, NAND, NOR, XNOR

Funcionamiento Operaciones aritméticas y lógicas

Saltos condicionales

Objetivo: Manejar el flujo del programa implementando lógica

1. IF: Comprueba que 2 condiciones sean verdaderas

   Ejemplo: IF R1 R2; -> Si R1 == R2

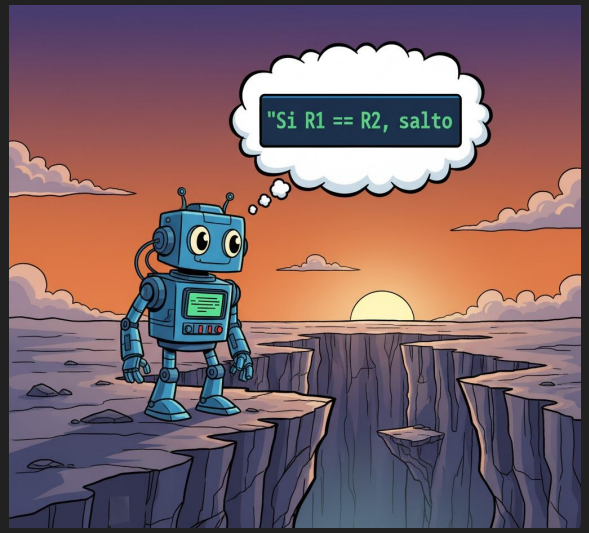
2. ELSEIF: Comprueba otras 2 condiciones

   Ejemplo: ELSEIF R2 R3-> Sino R2 == R3

3. ELSE: Ejecuta código si el condicional es verdadero

   Ejemplo: Si no se cumplen las 2 condiciones, ejecuta el resto

4. ENDIF: Finaliza el bucle



Funcionamiento saltos condicionales

Bucles

Objetivo: Ejecutar un bloque de instrucciones repetidamente mientras se cumpla una condición entre registros.

LOOP: Inicia un bucle comparando dos registros, ejecutando las instrucciones del bloque mientras sus valores sean distintos.
Ejemplo: LOOP R1 R2; → Mientras R1 != R2, se repite el bloque de instrucciones hasta ENDLOOP.

ENDLOOP: Marca el final del bloque de instrucciones del bucle. Cuando se alcanza, se vuelve a evaluar la condición del LOOP.
El flujo dentro del bucle puede incluir:

- **PR**: Actualizar un registro con un valor calculado o de otro registro temporal.
- **FR**: Liberar un registro para poder asignarle un nuevo valor.
- **Operaciones ALU**: Calcular valores (ADD, SUB, MUL, etc.) que luego se pueden guardar en registros temporales o finales.

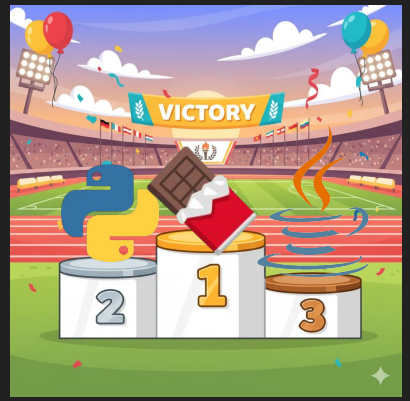
NOTA: No se pueden anidar 2 bucles, de momento no se ha implementado esa función

Flujo de ejecución del LOOP

Print del código

Este programa te permite imprimir por la consola una instrucción se pasa a un registro y del registro a la instrucción

Repositorio del proyecto



https://github.com/Romendesu/TraductorSignos