Máquina de Pila para el proyecto

.1. Estructura General

La máquina de pila está diseñada para ejecutar instrucciones en un formato de código intermedio. Este código intermedio se compone de instrucciones que manipulan una pila, realizan operaciones aritméticas y lógicas, y controlan el flujo del programa.

2. Componentes Principales

Pila (`pila`): Es el componente central donde se almacenan los valores temporales y los resultados de las operaciones. La pila tiene un tamaño máximo definido por `TAM_MEMORIA`.

Contadores y Punteros:

`sp` (Stack Pointer): Apunta al tope de la pila.

`pc` (Program Counter): Apunta a la instrucción actual que se está ejecutando.

`fp` (Frame Pointer): Utilizado para gestionar los marcos de pila en llamadas a funciones.

`ep` (Environment Pointer) y `hp` (Heap Pointer): Utilizados para gestionar la memoria y el entorno de ejecución.

Instrucciones (`Instruction`): Cada instrucción tiene un tipo (`IType`), que determina la operación a realizar, y puede tener argumentos adicionales.

3. Tipos de Instrucciones

Operaciones de Pila: `IPUSH`, `IPOP`, `IDUP`, `ISWAP`, etc.

Operaciones Aritméticas y Lógicas: `IADD`, `ISUB`, `IMUL`, `IDIV`, `IEQ`, `IGT`, `IGE`, `ILT`, `ILE`, `IAND`, `IOR`, `INEG`, `INOT`.

Control de Flujo: `IGOTO`, `IJMPZ`, `IJMPN`, `IHALT`.

Gestión de Memoria: `ISTORE`, `ILOAD`, `IALLOC`, `ISTORER`, `ILOADR`, `IPUSHA`, `ILOADA`.

Llamadas a Funciones y Recursividad: `ICALL`, `IMARK`, `IENTER`, `IRETURN`.

IPUSH: IPUSH <valor>

Función: Empuja un valor entero en la cima de la pila.

Sintaxis: IPUSH 5 empuja el valor 5 a la pila.

IPOP: IPOP

Función: Elimina el valor en la cima de la pila.

Sintaxis: IPOP elimina el valor superior de la pila.

IDUP: IDUP

Función: Duplica el valor en la cima de la pila.

Sintaxis: IDUP duplica el valor superior de la pila.

ISWAP: ISWAP

Función: Intercambia los dos valores superiores de la pila.

Sintaxis: ISWAP intercambia los dos valores superiores.

2. Operaciones Aritméticas y Lógicas

IADD: IADD

Función: Suma los dos valores superiores de la pila y empuja el resultado.

Sintaxis: IADD suma los dos valores superiores.

ISUB: ISUB

Función: Resta el valor superior de la pila del segundo valor superior y empuja el resultado.

Sintaxis: ISUB realiza la resta.

IMUL: IMUL

Función: Multiplica los dos valores superiores de la pila y empuja el resultado.

Sintaxis: IMUL multiplica los dos valores superiores.

IDIV: IDIV

Función: Divide el segundo valor superior de la pila por el valor superior y empuja el resultado.

Sintaxis: IDIV realiza la división.

IEQ: IEQ

Función: Compara los dos valores superiores de la pila por igualdad y empuja 1 si son iguales, 0 si no.

Sintaxis: IEQ compara igualdad.

IGT: IGT

Función: Compara si el segundo valor superior de la pila es mayor que el valor superior y empuja 1 si es cierto, 0 si no.

Sintaxis: IGT compara si es mayor.

IGE: IGE

Función: Compara si el segundo valor superior de la pila es mayor o igual al valor superior y empuja 1 si es cierto, 0 si no.

Sintaxis: IGE compara si es mayor o igual.

ILT: ILT

Función: Compara si el segundo valor superior de la pila es menor que el valor superior y empuja 1 si es cierto, 0 si no.

Sintaxis: ILT compara si es menor.

ILE: ILE

Función: Compara si el segundo valor superior de la pila es menor o igual al valor superior y empuja 1 si es cierto, 0 si no.

Sintaxis: ILE compara si es menor o igual.

IAND: IAND

Función: Realiza una operación lógica AND entre los dos valores superiores de la pila y empuja el resultado.

Sintaxis: IAND realiza AND lógico.

IOR: IOR

Función: Realiza una operación lógica OR entre los dos valores superiores de la pila y empuja el resultado.

Sintaxis: IOR realiza OR lógico.

INEG: INEG

Función: Niega el valor superior de la pila.

Sintaxis: INEG niega el valor superior.

INOT: INOT

Función: Realiza una operación lógica NOT en el valor superior de la pila.

Sintaxis: INOT realiza NOT lógico.

3. Control de Flujo

IGOTO: IGOTO <etiqueta>

Función: Salta a la instrucción etiquetada.

Sintaxis: IGOTO L1 salta a la etiqueta L1.

IJMPZ: IJMPZ <etiqueta>

Función: Salta a la instrucción etiquetada si el valor superior de la pila es 0.

Sintaxis: IJMPZ L1 salta a L1 si el valor superior es 0.

IJMPN: IJMPN <etiqueta>

Función: Salta a la instrucción etiquetada si el valor superior de la pila no es 0.

Sintaxis: IJMPN L1 salta a L1 si el valor superior no es 0.

IHALT: IHALT

Función: Detiene la ejecución del programa.

Sintaxis: IHALT detiene la ejecución.

4. Gestión de Memoria

ISTORE: ISTORE <direccion>

Función: Almacena el valor superior de la pila en la dirección de memoria especificada.

Sintaxis: ISTORE 10 almacena en la dirección 10.

ILOAD: ILOAD <direccion>

Función: Carga el valor de la dirección de memoria especificada en la pila.

Sintaxis: ILOAD 10 carga desde la dirección 10.

IALLOC: IALLOC <n>

Función: Reserva n espacios en la pila.

Sintaxis: IALLOC 5 reserva 5 espacios.

ISTORER: ISTORER <offset>

Función: Almacena el valor superior de la pila en la dirección relativa al puntero de marco (fp).

Sintaxis: ISTORER 2 almacena en fp + 2.

ILOADR: ILOADR <offset>

Función: Carga el valor de la dirección relativa al puntero de marco (fp) en la pila.

Sintaxis: ILOADR 2 carga desde fp + 2.

IPUSHA: IPUSHA <etiqueta>

Función: Empuja la dirección de la etiqueta en la pila.

Sintaxis: IPUSHA L1 empuja la dirección de L1.

ILOADA: ILOADA <offset>

Función: Carga el valor de la dirección absoluta en la pila.

Sintaxis: ILOADA 10 carga desde la dirección absoluta 10.

5. Llamadas a Funciones y Recursividad

ICALL: ICALL

Función: Llama a una función, guardando el estado actual.

Sintaxis: ICALL realiza la llamada a función.

IMARK: IMARK

Función: Marca el estado actual de la pila para una llamada a función.

Sintaxis: IMARK marca el estado.

IENTER: IENTER <n>

Función: Prepara un nuevo marco de pila para una función, reservando n espacios.

Sintaxis: IENTER 3 reserva 3 espacios.

IRETURN: IRETURN <n>

Función: Retorna de una función, limpiando n elementos de la pila.

Sintaxis: IRETURN 2 retorna limpiando 2 elementos.

4. Código Intermedio Aceptado

La máquina acepta un código intermedio que consiste en una secuencia de instrucciones que manipulan la pila y controlan el flujo del programa. Este código puede ser generado por un compilador que traduzca un lenguaje de alto nivel a este formato de instrucciones.

5. Capacidades de Complejidad

Recursividad: La máquina soporta recursividad mediante el uso de instrucciones como `ICALL`, `IMARK`, `IENTER`, y `IRETURN`. Estas instrucciones permiten gestionar los marcos de pila necesarios para las llamadas recursivas.

Llamadas a Funciones: Las funciones pueden ser llamadas y retornar valores utilizando la pila para pasar argumentos y almacenar valores de retorno.

Control de Flujo Complejo: Las instrucciones de salto condicional (`IJMPZ`, `IJMPN`) y no condicional (`IGOTO`) permiten implementar estructuras de control complejas como bucles y condicionales.

6. Limitaciones

Tamaño de Pila: La pila tiene un tamaño fijo (`TAM_MEMORIA`), lo que limita la cantidad de datos que se pueden almacenar simultáneamente.

Gestión de Memoria: Aunque la máquina tiene capacidades básicas de gestión de memoria, no implementa un sistema de gestión de memoria dinámica como un recolector de basura.