


UNIVERSIDAD AUTÓNOMA “TOMAS FRÍAS” CARRERA DE INGENIERÍA DE SISTEMAS				
Materia:	Arquitectura de computadoras (SIS-522)			
Docente:	Ing. Gustavo A. Puita Choque			N° Práctica 9
Auxiliar:	Univ. Aldrin Roger Perez Miranda			
Estudiantes:	Univ. Abraham Alberto Lupa Condori			
20/11/2024	Fecha publicación			
06/12/2024	Fecha de entrega			
Grupo:	1	Sede	Potosí	

1. ¿Qué es el 'stack' en el contexto del lenguaje ensamblador y cómo se utiliza?

En el contexto del lenguaje ensamblador, el stack es como una caja donde se guardan cosas de forma temporal. Funciona bajo la regla de último en entrar, primero en salir, lo que significa que lo último que pongas es lo primero en salir cuando lo necesites.

El stack se usa principalmente para: Guardar direcciones de retorno, pasar información a funciones, guardar registros y datos temporales.

Las dos acciones principales con el stack son: PUSH Guarda algo en el stack y POP Saca lo que guardaste y lo usa cuando lo necesites.

2. Describe un escenario práctico donde el uso de ensamblador sería más ventajoso que el uso de un lenguaje de alto nivel.

El uso de ensamblador es ideal cuando estamos trabajando con dispositivos que tienen recursos limitados, como un microcontrolador en un dispositivo embebido. Imagina que estamos programando el firmware para controlar un motor en una máquina industrial.

Controlar el hardware de forma precisa: Podemos interactuar directamente con los componentes del microcontrolador, como los puertos de entrada y salida, sin intermediarios.

Aprovechar al máximo los recursos limitados: Usamos menos memoria, lo que es crucial cuando el espacio disponible es muy reducido.

Obtener un rendimiento rápido y eficiente: El código ensamblador es muy rápido, lo cual es importante cuando el sistema necesita responder al instante, como en el control del motor.

Utilizar instrucciones especiales del procesador: Algunos microcontroladores tienen instrucciones que aceleran tareas específicas, y el ensamblador permite aprovecharlas al máximo.

3. Explique cada línea del siguiente código del lenguaje ensamblador y diga que es lo que se está haciendo.

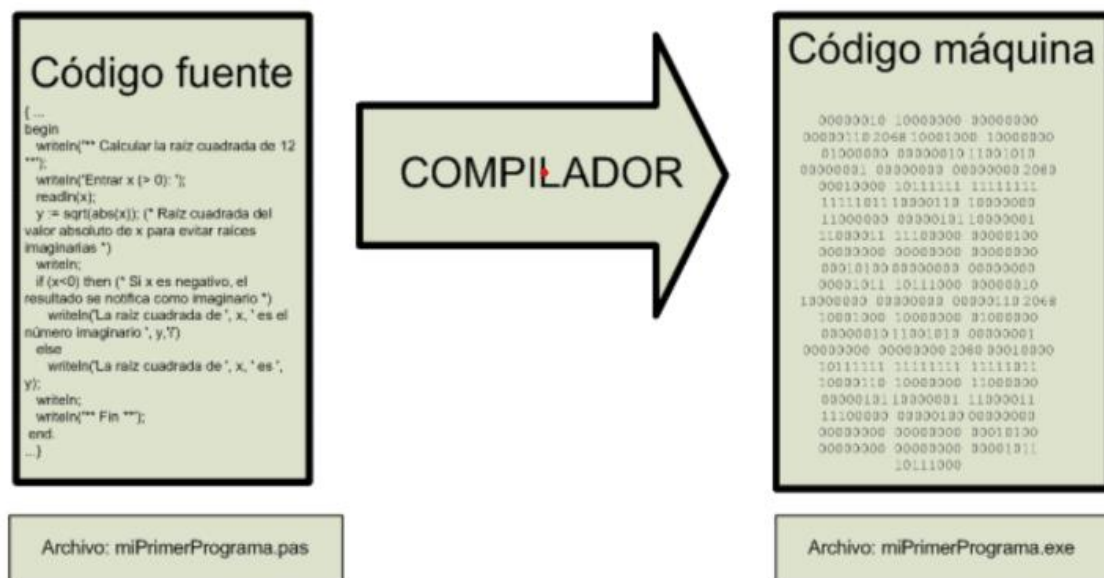
MOV AX, 5 ; Línea 1 pone el número 3 en el registro AX. Piensa en AX como una pequeña caja donde guardamos el número 3.

MOV BX, 10 ; Línea 2 coloca el número 8 en otro registro, BX, que es otra caja para guardar información.

ADD AX, BX ; Línea 3 suma lo que hay en BX (que es 8) al contenido de AX (que es 3). Ahora, AX se convierte en 11, porque $3 + 8 = 11$.

MOV CX, AX ; Línea 4 copia el valor que está en AX (que ahora es 11) y lo pone en CX. Entonces, tanto AX como CX contienen 11.

4. Explique detalladamente cómo funcionan los compiladores



Un compilador es una herramienta esencial que ayuda a transformar un programa escrito en un lenguaje de alto nivel como C, Java, Python a código máquina, que es el lenguaje que la computadora realmente puede entender y ejecutar.

Análisis Léxico (Lexical Analysis)

El compilador comienza leyendo todo el código fuente y lo divide en partes más pequeñas llamadas tokens, un token es simplemente una secuencia de caracteres que tiene un significado en el lenguaje de programación. Por ejemplo, en una línea de código como `int a = 5;`, los tokens serían `int`, `a`, `=`, `5`, `;`.

Análisis Sintáctico (Syntax Analysis)

Después de dividir el código en tokens, el compilador verifica que la secuencia de estos tokens siga las reglas del lenguaje, es decir, que tenga una estructura válida. Por ejemplo, revisa si las declaraciones y las expresiones están bien formadas.

ejemplo: Si escribes `5 = a;`, el compilador detectará que esta es una estructura incorrecta, porque no puedes asignar un valor a un número literal como si fuera una variable.

Análisis Semántico (Semantic Analysis)

Aquí, el compilador revisa si el código tiene sentido lógicamente, esto incluye la verificación de tipos de datos, asegurándose de que las operaciones sean válidas. Por ejemplo, sumar una cadena de texto con un número entero no tiene sentido y sería un error semántico.

ejemplo: En el caso de `int a = "hola";`, el compilador detectará un error porque estás tratando de asignar una cadena de texto a una variable de tipo entero.

Generación de Código Intermedio (Intermediate Code Generation)

Una vez que el código ha pasado todas las validaciones, el compilador lo convierte en una forma intermedia, este código no depende de una arquitectura específica de hardware, lo que facilita optimizaciones y cambios de plataforma sin tener que reescribir el programa completamente.

Un ejemplo: Puede generar algo similar al lenguaje ensamblador o a instrucciones más generales que el procesador no entiende directamente, pero que son una base para convertirlo a código máquina.

Optimización (Optimization)

El compilador intenta hacer que el código sea más eficiente, esto puede significar eliminar instrucciones que no son necesarias, reorganizar el código para que sea más rápido o reducir el uso de memoria.

Un ejemplo: Si hay un cálculo que se repite varias veces y no cambia, el compilador podría realizar ese cálculo una sola vez y luego reutilizar el resultado.

Generación de Código Objeto (Code Generation)

En esta etapa, el compilador convierte el código intermedio optimizado en código objeto, este código ya está en una forma que la CPU puede ejecutar directamente, pero aún no es un programa completo.

ejemplo: El compilador genera instrucciones específicas para la arquitectura del procesador de la máquina, como por ejemplo en el formato binario que entiende un procesador Intel o ARM.

7. Enlazado (Linking)

El enlazador es responsable de combinar el código objeto con otras bibliotecas o módulos necesarios para completar el programa, las bibliotecas son conjuntos de funciones ya escritas que el programa necesita para ejecutar tareas comunes, como mostrar texto en la pantalla o leer archivos.

ejemplo: Si usas una función como `printf` en C, el enlazador incluirá el código que hace posible esta función.

Generación del Archivo Ejecutable (Executable Generation)

Finalmente, después de que el código ha sido enlazado, el compilador genera un archivo ejecutable que la computadora puede ejecutar directamente, este archivo contiene todas las instrucciones necesarias para que el programa se ejecute en una máquina específica.

ejemplo: En Windows, este archivo podría ser un .exe, y en Linux un archivo binario.

5. Realizar sus propias capturas de pantalla del siguiente procedimiento:

