UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias. Compiladores

Practica 00

Reporte

Autor:

González Arceo Carlos Eduardo **318286488**

10 de Abril 2024

1. Ejercicios

3.b Compare el contenido de **programa.i** con **stdio.h**. Describa las similitudes más notables entre ambos archivos.

El archivo programa.i es una versión del código fuente del programa.c donde se han resuelto todas las inclusiones de archivos de cabecera. Esto resulta en que gran parte del contenido de stdio.h está en programa.i.

3.c Explique que ocurre durante la ejecución del comando cpp.

El comando hace referencia al preprocesador de C. El cual es una herramienta que procesa el código fuente (Elimina comentarios, Incorporar definiciones en el codigo, etc) antes de que el compilador lo compile. El resultado es un codigo intermedio.

4.a Describa la función de la opción **-Wall**.

La opción -Wall habilita las alertas del compilador, de esta forma el compilador avisa sobre prácticas de programación problemáticas, errores o situaciones que podrían llevar a errores.

4.b Explique el propósito de la opción **-S**.

La instrucción -s le da la orden al compilador de detenerse al momento de generar el código en ensamblador,pero que no continúe con las etapas posteriores de ensamblaje o enlazado.

4.c Indique que contiene el archivo de salida y cuál es su extensión.

El archivo de salida es un archivo .s que contiene instrucciones en ensamblador resultantes del código original. Este código ensamblador es específico para la arquitectura del procesador de la máquina en la que se está compilando.

4.d Identifique que programa es invocado por **gcc** al usar la opción **-S**.

Cuando se utiliza la opción -S con el compilador gcc, el programa invocado es el compilador de C en sí, como cc1 o cc1plus (para C++).

5.a Explique que se espera que contenga el archivo **.o**.

Esperamos que contenga código máquina (código binario) generado a partir del código ensamblador del archivo .s.

5.b Describa de manera general el contenido del archivo **.o** y por que es importante.

El archivo contiene instrucciones en binario que son enviadas directamente al cpu para ser ejecutadas.

Es importante porque es esencial para el proceso de enlace, además de que puede incluir información de depuración, lo que permite analizar y corregir problemas en el código.

5.c Identifique que programa se invoca con el comando **as**.

El comando as invoca al GNU assembler, que es el responsable de convertir código ensamblador a código objeto.

8.a Si el comando **ld** arroja errores, investigue como enlazar un programa utilizando **ld**.

Para enlazar un programa utilizando el enlazador ld, necesitas tomar uno o más archivos de objeto (.o) generados previamente por el ensamblador o el compilador y combinarlos para producir un archivo ejecutable. ld es el enlazador que forma parte de GNU Binutils y se encarga de esta tarea.

8.b Describa el resultado obtenido al ejecutar el comando anterior.

Al ser ejecutado, el comando da como resultado un ejecutable que contiene lo necesario para poder ejecutar el algoritmo.

10.b ¿Cambio algo en la ejecución final?

No que me fuera apreciable. Solo se pudo apreciar que el valor asignado a PI, hace que el calculo del área sea más preciso.

11 Escribe un segundo programa en lenguaje C en el agregue 4 directivas del preprocesador de C (cpp). Las directivas elegidas deben jugar algún papel en el significado del programa, ser distintas entre sí y ser diferentes de las utilizadas en el primer programa (aunque no están prohibidas, si las requiere). Explique la utilidad general de las directivas usadas y su función en particular para su programa.

```c
#include <stdio.h>
#include <assert.h>  // Necesario para usar la directiva #assert

// Identificacion del archivo
#ident "Programa de prueba - Ejercicio 11"

// Definici n de una constante simb lica
#define MAX_VALUE 100

// Funci n principal
int main(void) {
    int value = 120;

    // Uso de la directiva #assert para verificar una condici n
    assert (MAX_VALUE >= 100)

    // Comprobaci n de un valor que podr a causar un error
    if (value > MAX_VALUE) {
        #warning "El valor supera el m ximo permitido. Considera ajustar MAX_VALUE."
    }

    // Condicion que podr a generar un error critico
    #if MAX_VALUE < 50
    #error "El valor de MAX_VALUE es demasiado bajo. Debe ser al menos 50."
    #endif

    printf("Valor: %d\n", value);

    return 0;
}
```

Las directivas del preprocesador utilizadas sirven para mejorar el control y la seguridad del código durante la compilación:

- #assert: Verifica condiciones críticas antes de la compilación, evitando errores lógicos tempranos.
- #ident: Añade una identificación al archivo objeto, útil para la gestión de versiones y seguimiento del código.
- #error: Detiene la compilación si se detectan condiciones inaceptables, asegurando que el código cumpla requisitos mínimos.
- #warning: Genera advertencias para alertar sobre posibles problemas, sin detener la compilación, sugiriendo revisiones.

Estas directivas ayudan a prevenir errores y a mantener el código seguro y bien gestionado desde las

primeras fases del desarrollo.

2. Conclusiones

La practica nos permitió generar y analizar archivos intermedios como .i, .s, y .o, lo cual nos dejo ver y comprender en mayor detalle el proceso de compilación, empezando por el procesador de C, el cual es crucial para modularizar y gestionar el flujo de compilación, hasta el ensamblado y enlazado de código.

El proceso de enlace es esencial para combinar archivos objeto en un ejecutable funcional, garantizando que todas las dependencias estén correctamente resueltas. Dominar estos aspectos conduce a un software más robusto, eficiente y mantenible.