UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

PROFESOR: Erick Bermudez Vazquez

Laboratorio de Sistemas Operativos

REPORTE PRIMER ENTREGA

Carlos Edwin Bautista Tzintzun.

María Elena Gabriel Nicolás

Magdalena Reyes Chavez

Introducción

El presente trabajo corresponde a la primera entrega del laboratorio de Sistemas Operativos y tiene como objetivo la elaboración de un programa en C que permita simular una Unidad Aritmético Lógica (ALU).

La ALU es un componente fundamental de la Unidad Central de Procesamiento (CPU) encargada de realizar operaciones aritméticas y lógicas. El programa realizado es capaz de recibir como entrada un archivo de instrucciones en formato .asm, leer las instrucciones y aplicar las operaciones a los registros indicados.

De esta manera, el proyecto busca comprender el funcionamiento básico de una ALU y su importancia dentro de la ejecución de programas en un sistema operativo.

Objetivos

Generales

El programa deberá simular una ALU, reconociendo comandos como “ejecutar a.asm” o “salir”. Además, tendrá que abrir y leer archivos para procesar y validar las operaciones que definirán su comportamiento. Para complementar su funcionamiento, se implementará una representación gráfica de la ALU utilizando la librería ncurses en C.

Específicos

- El programa debe diferenciar y validar los comandos de la terminal como ‘salir’ o ‘ejecutar x.asm’ así como el tipo de extensión del archivo.

- Se declaran cinco registros del tipo entero, estos registros responderán a las instrucciones válidas de los archivos con extensión .asm

Ax	Bx	Cx	Dx

- Los detalles de la impresión son las siguientes:

ID	Número de veces que se ejecutó un archivo con extensión .asm dentro de la ejecución
PC	Número de la siguiente línea que tendrá que ejecutar.
IR	Instrucción ejecutada, con o sin errores sintácticos.
STATUS	Dará el veredicto de la instrucción ejecutada, si es correcta o especificando el error.

- El intérprete implementado al momento de leer las instrucciones del archivo .asm deberá comprobar que la sintaxis sea correcta como el siguiente ejemplo.

MOV Ax,5

Validando que la instrucción contenga una operación válida, que el registro exista y dependiendo del caso, que el número sea válido para la operación.

- Las operaciones definidas tendrán que ser las siguientes:

MOV	Asignación
ADD	Suma
SUB	Resta
MUL	Multiplicación
DIV	División
INC	Incremento en 1
DEC	Decremento en 1

Es importante aclarar que las división entre cero no es válida y que así mismo las divisiones solo deberán mostrar su parte entera, no decimal.

Diseño

El programa cuenta con los siguientes archivos:

- principal.c
- interprete.c
- interprete.h
- operaciones.c
- operaciones.h

principal.c

Lleva el flujo de trabajo más importante, ya que en ella se encuentra la función main.

En este módulo se **inicializa la librería ncurses** que permite crear una interfaz en la terminal. El programa muestra un prompt ($) para que el usuario ingrese comandos.

Cada comando **se copia y se envía a la función interprete()**, que se encarga de validar e identificar si se trata de una instrucción correcta (ejecutar archivo.asm) o una orden de salida (salir).

El flujo se ejecuta dentro de un **ciclo do-while**, el cual permanece activo mientras no se reciba el comando de salida. Finalmente, se libera el entorno de ncurses con endwin(), cerrando la interfaz de manera adecuada.

interprete.c

Se tienen dos funciones desarrolladas: interprete y validar_archivo.

Interprete

Procesa el comando que el usuario escribe desde la terminal.

Detecta si el comando es "salir" (en minúsculas, mayúsculas o con mayúscula inicial).

- Si lo es, imprime "Saliendo...", espera un segundo y devuelve 1 como bandera para cerrar el programa.

Si no es "salir":

- Divide el texto en tokens usando espacios como delimitadores.
- Comprueba que la primera palabra sea ejecutar.
- Si la segunda palabra existe y termina en .asm, lo considera un archivo válido:
    - Imprime el nombre del archivo.

    - Incrementa el proceso_id.
    - Guarda el archivo como "archivo actual".
    - Llama a la función validar_archivo(nombre_archivo).
- Si no se cumple lo anterior, muestra mensajes de error como:
    - "Error: Falta el nombre del archivo".
    - "Error: El archivo debe tener extensión .asm".
    - "Instrucción no válida".
    - validar_archivo

validar_archivo

Abre y valida línea por línea el archivo a.asm, verificando las instrucciones y modificando los registros según lo indique cada operación en las líneas del archivo.

Imprime el encabezado de la tabla: (Id, PC, Ax, Bx, Cx, Dx, Proceso, IR, Status)
Abre el archivo en modo lectura.
Realiza el recorrido línea por línea.

- Elimina los saltos de línea al final de cada instrucción..
- Obtiene la primera palabra y lo guarda como instrucción
- Verifica si la instrucción es de tipo 1 o tipo 2
    - Tipo 1: MOV, ADD, SUB, DIV; ya que estas instrucciones requieren de dos parámetros, el registro y el valor.
    - Tipo 2: INC, DEC; solo requieren de un parámetro, ,que es el registro
- Si no encuentra ninguna instrucción válida regresa el error “Operación desconocida”.

Validación de registros

- Realiza las comparaciones para que los registros solo sean Ax, Bx, Cx, y Dx.
- Si no cumplen con ninguna de estas comparaciones regresa el error: “Registro inválido”.

Por último tenemos la ejecución de las operaciones:

- Como ya se guardó la operación pero aún no sabemos exactamente qué operación es hacemos otra comparación y ahora si hacemos la llamada a la función correspondiente, si es de tipo 1 le pasamos los dos parámetros y si es de tipo 2 solo un parámetro.

- Después de cada operación correcta llama a la función tabla() para la impresión.
- Si ocurre un error muestra el error en Status.

Al terminar cerramos el archivo y llamamos a la función reiniciar_registros() para reiniciar los valores de los registros y limpiamos la pantalla para que el encabezado no siga en la pantalla.

operaciones.c

El programa aquí directamente ya se implementan las operaciones de la unidad aritmético-lógica (ALU) que trabaja con los registros (Ax, Bx, Cx y Dx) previamente definidos he inicializadas, se realizan las operaciones de asignación, suma, resta, multiplicación, división, incremento y decremento.

Se tiene adicionalmente una función llamada ‘reinciar_registros’ la cuál es usada cada que se ejecuta un archivo con extensión .asm, esto para no tener problemas con el valor de los registros al ejecutar varios archivos o para no tener problemas de impresión.
Después con la ayuda de la librería ncurses, se despliega en la terminal una tabla que muestra en tiempo real el estado de los registros, junto con información adicional como el identificador del proceso, el número de línea ejecutada, el archivo en ejecución, la instrucción procesada y posibles errores. Utiliza controles para actualizar pantalla, esperar la tecla enter para la siguiente instrucción que va a leer, para limpiar la línea y actualizar la pantalla cada vez que se ejecuta una instrucción.

interprete.h y operaciones.h

En estas librerías tenemos los prototipos de las funciones desarrolladas en interprete.c y operaciones.c respectivamente.
Librerías utilizadas:

- stdio.h
- string.h
- stdlib.h
- ncurses.h
- unistd.h

Requisitos y compilación del programa.

IMPORTANTE: Para compilar el programa es necesario haber instalado la librería de ncurses.

Compilar: gcc -o ejecutable principal.c interprete.c operaciones.c -lncurses
Ejecutar: ./ejecutable