CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

GRADO EN INGENIERíA SOFTWARE
SISTEMAS OPERATIVOS, 2022-2023

PROYECTO I

Este proyecto vale un 15% de la nota final de la asignatura (el proyecto II valdrá un 20%). Es necesario obtener un 4 para poder liberar esta materia.

En esta práctica se pide desarrollar un simulador de memoria caché de un sistema ficticio de los años 80 llamado MEGATRONIX,

MEGATRONIX tenía un bus de memoria de 12 bits y usaba memoria física, con una caché de 8 líneas con correspondencia directa y 16 bytes por línea. Con lo que hemos aprendido en teoría, sabemos que la caché interpretará cada dirección de memoria recibida de la CPU en tres campos: palabra (4 bits), línea (3 bits) y etiqueta (5 bits).

Hay que desarrollar en C sobre *Linux* un proceso MEMsym. El proceso dispone de un array de 16 elementos, del tipo `T_CACHE_LINE`, cuya definición es:

```
typedef struct {
    unsigned char ETQ;
    unsigned char Data[TAM_LINEA];
} T_CACHE_LINE;
```

El proceso tendrá una variable `globaltime` que inicializará a valor 0. También creará otra con el nombre `numfallos` inicializada a 0.

En el arranque, MEMsym inicializa a `xFF` (hexadecimal) los campos `Label` y a `x23F` (hexa) todos los campos de datos de la caché. Luego lee el fichero binario `CONTENTS_RAM.bin` en la variable `Simul_RAM`, que es un array de 4096 `unsigned char`. A continuación, comienza la lectura del fichero de texto `dirs_memoria.txt` que contiene una lista de direcciones de memoria en hexadecimal, una por línea. El proceso tiene que tratar adecuadamente los errores si alguno de los ficheros no existe y avisar con el mensaje correspondiente. En ese caso terminará el proceso con `return(-1)`.

Se repetirá el siguiente protocolo:

- MEMsym lee una dirección del fichero `accesos_memoria.txt`.

- Obtiene el número de línea y comprueba si la etiqueta de la dirección es igual a `Label` de la línea de la caché.
- Si no es así, incrementa el valor de `numfallos` y escribe una línea con el texto "`T: %d, Fallo de CACHE %d, ADDR %04X Label %X linea %02X palabra %02X bloque %02X`", siendo T el instante. Se incrementa en 10 el contador `globaltime`. Se copia el bloque correspondiente desde el array RAM y se imprime un mensaje indicando que se está cargando el bloque X en la línea Y. Se actualizan tanto el campo `Label` como los 8 bytes de datos de la línea.
- Por pantalla se escribe "`T: %d, Acierto de CACHE, ADDR %04X Label %X linea %02X palabra %02X DATO %02X`". Cada carácter leído se añade a una variable llamada texto, que es un array de 100 caracteres como máximo (no hace falta usar memoria dinámica).
- El proceso vuelca el contenido de la caché por pantalla con el siguiente formato:

```
T: 1, Fallo de CACHE 1, ADDR 0B2E ETQ 16 linea 02 palabra 06 bloque B2
Cargando el bloque B2 en la linea 02
T: 11, Acierto de CACHE, ADDR 0B2E ETQ 16 linea 02 palabra 06 DATO 20
ETQ:FF  Data 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
ETQ:FF  Data 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
ETQ:16  Data 70 20 65 69 64 69 74 6F 63 20 6D 61 75 71 69 6C
ETQ:FF  Data 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
ETQ:FF  Data 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
ETQ:FF  Data 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
ETQ:FF  Data 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
ETQ:FF  Data 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
```

  Los datos se imprimen de izquierda a derecha de mayor a menor peso. Esto significa que el byte situado más a la izquierda es el byte 15 de la línea y el situado a la derecha el byte 0.

- EL proceso hace un `sleep()` de 1 segundo.

Al final se imprimirá un mensaje con el número total de accesos, número de fallos y tiempo medio de acceso.

Debajo, otro mensaje con el texto leído carácter a carácter desde la caché.

```
Accesos totales: 12; fallos: 11; Tiempo medio: 10.17
Texto leído: ltiev, soonn
```

Antes de salir, el programa volcará los contenidos de los 128 bytes de información (8 líneas de 16 bytes cada una) de la caché en un fichero binario llamado `CONTENTS_CACHE.bin`. El byte 0 de ese fichero es el byte 0 de la línea 0 de la caché y el byte 128, es el byte 15 de la línea 15.

El proceso tendrá que desarrollar OBLIGATORIAMENTE las siguientes funciones:

```
void LimpiarCACHE(T_CACHE_LINE tbl[NUM_FILAS]);

void VolcarCACHE(T_CACHE_LINE *tbl);

void ParsearDireccion(unsigned int addr, int *ETQ, int
*palabra, int *linea, int *bloque);
```

```
void TratarFallo(T_CACHE_LINE *tbl, char *MRAM, int ETQ,
int linea, int bloque);
```

Entrega de la práctica

Cada equipo desarrollará el proyecto en `github`. Los equipos pueden ser de 1 o 2 alumnos. Se pueden 3 alumnos por grupo (de forma excepcional, previa aprobación por el profesor y sabiendo que la exigencia aumentará, pudiendo pedir el profesor implementaciones adicionales). En la clase donde se presenta este enunciado hay que comunicar al profesor los equipos. Solo hay que subir una versión por equipo pero tiene que haber commits de todos ellos. Subirá un fichero `README.txt` con los nombres delos estudiantes, el fuente `MEMsym.c` y el volcado de la salida de la ejecución en un fichero llamado `logcache.txt`. Para ello ejecutar:

```
./MEMsym > logcache.txt
```

Evaluación

- El repositorio existe, está declarado como privado y se ha dado acceso al usuario del profesor. El programa compila. En el README.txt se hace constar el nombre de todos los autores. Si no se verifican estas condiciones la práctica no se evalúa. Si se subsanan posteriormente se penaliza con 2 puntos el incumplimiento inicial.
- El programa arranca correctamente, inicializa las variables, lee el fichero `CONTENTS_RAM.bin` y abre el `accesos_memoria.txt` con el debido control de errores, incluyendo un `return(-1)` en caso de que alguno de los ficheros no exista (1 punto).
- El proceso vuelca bien por pantalla el contenido de la caché (1 punto)
- El proceso calcula correctamente la línea de caché de una dirección dada y comprueba si la etiqueta que contiene es la misma (1 punto).
- El proceso escribe correctamente en pantalla el contenido del byte seleccionado (0,5 puntos).
- El proceso hace correctamente el sleep de 1 segundo (0,5 puntos).
- El proceso escribe correctamente el número de accesos y fallos, el tiempo medio y el texto leído (1 punto).
- El proceso termina correctamente después de leer la última línea de `dirs_memoria.txt` y realizar las acciones pedidas en el enunciado (0,5 puntos).
- Los commits están correctamente documentados y se puede trazar la participación de cada uno de los componentes del equipo (1,5 puntos). Si alguno de los componentes no ha contribuido se considerará que no ha entregado el proyecto.
- La escritura final de la caché en el fichero `CONTENTS_CACHE.bin` es correcta (1punto)
- Se han desarrollado las 4 funciones pedidas siguiendo los prototipos de este enunciado (1 punto)
- Limpieza y documentación del código (1 punto). Se usan constantes en lugar de valores numéricos, hay comentarios, se emplean máscaras de bits para extraer los datos de la memoria.

La evaluación se hará usando ficheros `accesos_memoria.txt` y `CONTENTS_RAM.bin` diferentes del que se proporciona para el desarrollo, por lo que aconsejamos probar con varias combinaciones de direcciones de entrada.