

## Simulador de Memoria Caché

*Fecha de entrega límite: 11 Abril.*

### Objetivo

Se trata de realizar un simulador de memoria caché donde podamos simular diferentes configuraciones y los efectos de incorporar o no una Victim Caché:

*Tamaño de línea:* tamaño en bytes potencia de dos.

*Número de líneas:* número entero potencia de dos.

*Asociatividad:* número entero potencia de dos, igual o inferior al número de líneas, con algoritmo de reemplazo LRU

*Victim Cache:* número entero potencia de dos, indicando el número de líneas de esta caché, totalmente asociativa con algoritmo de reemplazo LRU. Si el valor es 0 significa que no hay Victim Caché.

El simulador leerá la configuración desde un fichero “config.txt” de este estilo.

```
Nlin: 1024
Tlin: 64
Asoc: 8
VC: 32
```

El simulador ha de calcular la **tasa de fallos** que produce la caché al acceder a memoria siguiendo una traza de accesos almacenada en “traza.txt”, cuyo formato será de este estilo:

```
0x42ad34f1 0x42ad5534 0x12345678 ...
```

donde la traza estará formada por direcciones en formato hexadecimal (“0x” significa que el conjunto de caracteres es un número hexadecimal)

### Lectura y escritura de ficheros

Para leer un número en hexadecimal de un fichero podemos utilizar la función “fscanf”:

```
unsigned long int n;
FILE *f;
fscanf(f, "%lx", &n);
```

Podemos generar nuestros propios ficheros de traza haciendo que los programas escriban las direcciones utilizadas mediante la función “fprintf”. Véase el ejemplo (\*):

```
double v[MAX], res;
FILE *f;
f=fopen("traza.txt", "w");
for (i=0 ; i<MAX ; i++) {
    res=res+v[i];
    fprintf(f, "%p ", &v[i]);
}
```

(\*) esto es un ejemplo concreto que guarda la traza de acceso al vector *v*, el programa deberá calcular la tasa de fallos para cualquier traza generada a partir de cualquier código, NO únicamente para este ejemplo.

Información de utilidad para abrir y cerrar ficheros en C:

[https://es.wikibooks.org/wiki/Programaci%C3%B3n\\_en\\_C/Manejo\\_de\\_archivos](https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C/Manejo_de_archivos)

### ***Cálculo del valor de un conjunto de bits***

Para calcular el valor que representa un conjunto de bits consecutivos pertenecientes a una dirección, utilizaremos la siguiente función. Esta función calcula el valor entre dos bits (menor y mayor) ambos inclusive. Los parámetros son: el valor de la dirección, el bit de menor peso y el bit de mayor peso del conjunto a considerar. *Es requisito imprescindible el uso de dicha función.*

```
#include <stdio.h>
#include <stdlib.h>
unsigned long int rangobits (int bitmenor, int bitmayor, unsigned long
int n)
{
    unsigned long int bit2=1, bit1=1;

    if (bitmenor<0 || bitmenor>47) {
        printf("Error en Bitmenor: %d\n", bitmenor);
        exit(0);
    }
    else if (bitmayor<0 || bitmayor>47) {
        printf("Error en Bitmayor: %d\n", bitmayor);
        exit(0);
    }
    else if (bitmenor>bitmayor) {
        printf("Error en Orden\n");
        exit(0);
    }
    else if (bitmayor-bitmenor+1<48 ) {
        /** calcula valor 2 elevado a bitmenor */
        bit1= bit1 << bitmenor;

        /** calcula valor 2 elevado a (bitmayor-bitmenor + 1) */
        bit2 =bit2 << (bitmayor-bitmenor + 1);

        n=n / bit1;
        n=n % bit2;
    }
    return n;
}
```

### ***Formato de entrega***

Un fichero “Gxx-SIMU-DosApellidosA-DosApellidosB.zip” que contenga el fichero “SimuladorCache.c” bien organizado y documentado explicando entre comentarios las diferentes funciones y/o partes del código.