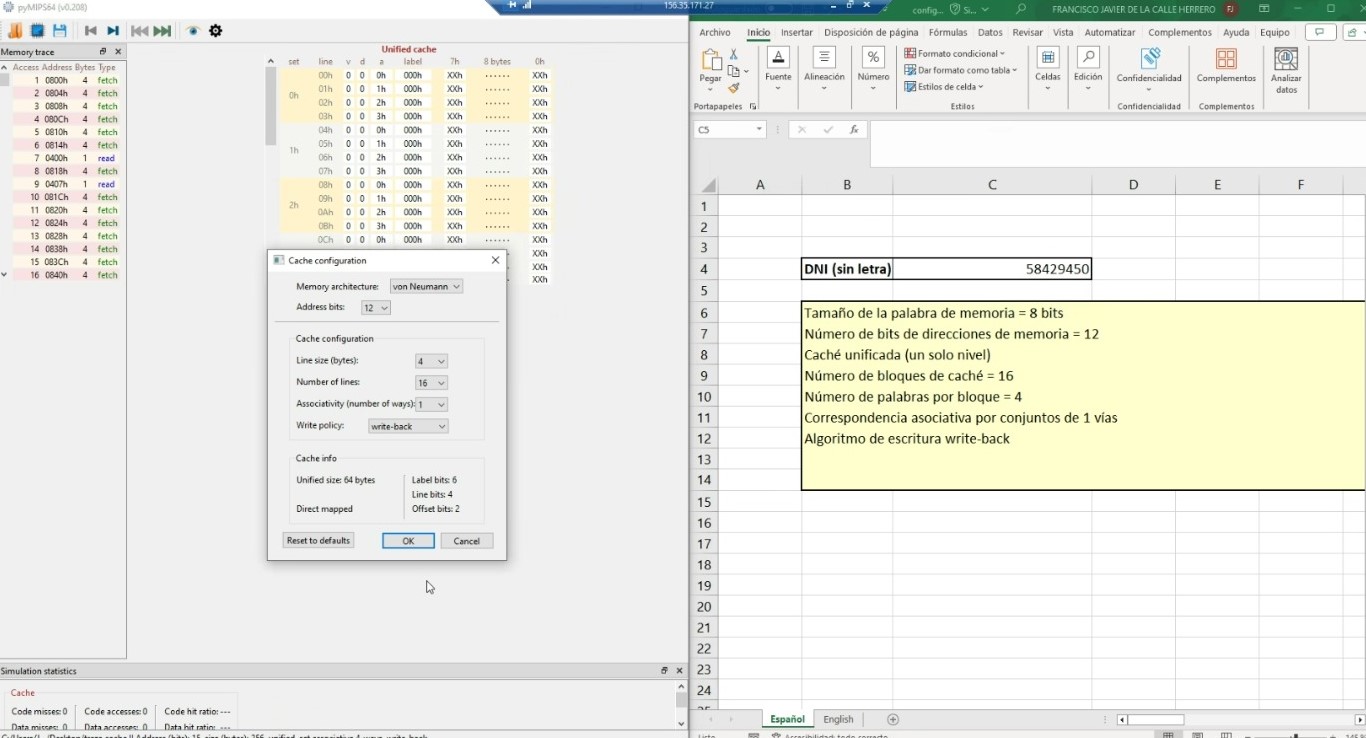
# RESUMEN EXAMEN PRACTICO2

## EJERCICIO 1 - Preguntas sobre PYMIPS varias

1. Configurar la caché con la configuración dada.



NUMERO BITS DE DIRECCIONES DE MEMORIA = Address Bits

CACHÉ UNIFICADA

NUMERO DE BLOQUES DE CACHÉ = Number of lines

NUMERO DE PALABRAS POR BLOQUE \* TAMAÑO DE LA PALABRA DE MEMORIA(Bytes) = Line Size (Bytes)

CORRESPONDENCIA ASOCIATIVA POR CONJUNTOS DE 1 VIA = Associativity

WRITE-BACK

1. Enumera un ejemplo de localidad temporal para el código y otro de localidad espacial para los datos.

La localidad temporal de los programas viene motivada principalmente por la existencia de bucles.

Localidad espacial: las palabras próximas en el espacio de memoria a las recientemente referenciadas tienen una alta probabilidad de ser también referenciadas en el futuro cercano.

1. 4 primeros accesos que provoquen fallo

Recuadro de la caché en rojo

1. 2 primeros accesos que provoquen reemplazo y el bloque reemplazado

Bit d (sucio) se pone a rojo y la etiqueta cambia y se pone en rojo también.

1. 2 primeros accesos que provoquen actualización y el bloque reemplazo

Lo mismo que el reemplazo pero hay que esperar a una orden write. Después de eso ejecutamos hasta que el bit d=1 cambia a d=0 y se ponen en rojo

1. Cuál fue el primer acceso de escritura y poner todos los bits de etiqueta, conjunto y desplazamiento.

Esperar a la primera orden write y calcular la etiqueta, conjunto y desplazamiento (pasando el ratón por el acceso en Memory trace)

1. Cuál sería el porcentaje de aciertos de la traza si fueran accesos aleatorios.

(Tamaño de cache / Tamaño memoria) \*100

## EJERCICIO 2.1 - Analizar los accesos de caché L1 o L3 (Bandwidth y Valgrind)

PARTE DE BANDWIDTH:

wget https://www.atc.uniovi.es/grado/2ac/files/session3-3.tar.gz

tar xvfz session3-3.tar.gz

Patrón de fondo

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Compilar el programa: make bandwidth32

Si te aparecen errores: sudo apt install nasm

Ejecutar programa: . /bandwidth32 | more

Ejecutar programa más rápido: sudo nice -n -2 ./bandwidth32 –fastest

Texto

Descripción generada automáticamente

**Cache L1 dividida en código y datos:**

* Datos: tamaño de línea (64 bytes), vías (8), conjuntos (64), tamaño (32 kbytes)
* Código: tamaño de línea (64 bytes), vías (8), conjuntos (128), tamaño (64 kbytes)

**Cache L2 unificada:** tamaño de línea (64 bytes), vías (16), conjuntos (4096), tamaño (4096 kbytes)

**CacheL3 unificada:** tamaño de línea (64 bytes), vías (12), conjuntos (40960), tamaño (30720 kbytes)

Gráfico, Histograma

Descripción generada automáticamente

Si te preguntan el ancho de banda de un experimento das el pico más alto para ese experimento.

El acceso secuencial es más rápido siempre ya que se aprovecha el principio de localidad.

Cada escalón es un nivel de memoria, comienza en L1, después va L2, luego L3 y por último Memoria Principal.

**PARTE DE VALDRIND:**

Compilar programa: gcc [archivo.c] -o [nombre ejecutable]

Ejecutar con valgrind: valgrind --tool=cachegrind ./ejecutable

Ejecutar programa con tiempo: time ./ejecutable

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## EJERCICIO 2.2 - MAPS

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

## EJERCICIO 3 – Memoria Virtual

IMPORTANTE!!!!! -> Mirar si está activo el linmem

systemctl status linmem 🡪 Tiene que poner una luz verde y tal.

systemctl enable linmem 🡪 Para activarlo.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

* **Para compilar acordarse de lmen!!!!!!:**

gcc {archivo.c} -o {nombre\_ejecutable} -lmen

* **Para que te salgan los warnings:**

gcc -Wall {archivo.c} -o {nombre\_ejecutable} -lmen

* **Para el resto de las librerías:**

gcc {archivo.c} -o {nombre\_ejecutable} -lmen -lrt -lm -pthread -lpthread

Para mirar que bits se activan.

Por ejemplo si la flag es 025h. En binario seria 0000 0010 0101. Serían los bits 0 a 11.

Se activarían el bit de presencia, el de usuario, y el de dirty.

Diagrama

Descripción generada automáticamente con confianza baja