Arquitectura de Computadores
MNEME

Sergio García de Armas

Alberto Fuentes Tarife

P. Najor Cruz Cruz

Escuela Técnica Superior Ingeniería Informática

Universidad de La Laguna

Contenido

1. Introducción ____ *4*

1.1. ¿Qué es Mneme? ____ **4**

1.2. Descarga y ejecución del simulador ____ **5**

1.2.1. Ejecución rápida ____ 5

1.3. Ayuda ____ **6**

2. Configuración ____ *7*

2.1. Asistente de configuración ____ **7**

2.1.1. Primer apartado ____ 7

- Configuración general ____ 8
- Envejecimiento de páginas ____ 8
- Asignación de memoria ____ 9
- Memoria principal ____ 10

2.1.2. Segundo apartado ____ 12

- TLB ____ 12
- Tabla de páginas ____ 14

2.1.3. Tercer apartado ____ 17

- Memorias Cachés (L1, L2, L3) ____ 17

2.2. Ficheros de configuración ____ **20**

3. Simulación ____ *22*

3.1. Ficheros de traza ____ **22**

3.1.1. Formato ____ 22

3.2. Interfaz de simulación ____ **23**

3.2.1. Memoria principal y cachés ____ 23

- Control de la simulación ____ 24
- Memoria principal ____ 24
- Memorias cachés (L1, L2, L3) ____ 27
- Código de colores ____ 29

3.2.2. Gráfico de simulación ____ 29

3.2.3. Tabla de páginas ____ 31

- Dirección virtual ____ 32
- Tablas de páginas ____ 33

- TLB 34
- Memoria principal 35

3.2.4. Procesos 35

- Procesos 36
- Alojamiento en memoria de procesos 37

3.2.5. Bkthr 38

1. Introducción

1.1. ¿Qué es Mneme?

Mneme es un simulador multiplataforma de jerarquía de memoria de carácter didáctico.

Ha sido desarrollado por Beatrice Popescu como un proyecto para el Departamento de Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores de la Universidad de La Laguna.

Su objetivo es servir de apoyo al aprendizaje de los conceptos de jerarquías de memoria, fundamentales en materias como Sistemas Operativos y Arquitectura de Computadores.

El simulador abarca, entre otros, los siguientes conceptos de jerarquías de memoria:

Memoria virtual

- Direcciones virtuales y direcciones reales.
- Paginación
- Mecanismos de traducción de direcciones y TLBs.
- Estrategias de búsqueda, colocación y reemplazamiento en memoria principal.

Niveles de memoria

- Memoria secundaria.
- Memoria principal.
- Memoria caché multinivel.
    - Nivel 3, Nivel 2 y Nivel 1 (conjunta o separada en datos e instrucciones)
    - Tamaños de línea configurables
- Estrategias de colocación, reemplazamiento y coherencia entre los diferentes niveles.

Ejecución multiproceso

Permite además cargar ficheros de traza ofreciendo como resultado una serie de estadísticas de rendimiento.

1.2. Descarga y ejecución del simulador

El simulador es un applet de Java que puede incrustarse en una página HTML y ser ejecutado sin más requerimientos que un navegador web.

La versión más reciente puede descargarse de la web del departamento: http://www.isaatc.ull.es/portal/proyectos/mneme

Para ejecutar el simulador basta con descomprimir el fichero "*.tgz"* descargado de la web y abrir el fichero "*sj.html"* en un navegador con soporte Java.

1.2.1. Ejecución rápida

Para poner en funcionamiento de manera inmediata el simulador pueden seguirse los siguientes pasos:

1. Cargar un fichero de configuración local (*sjlocal/files/conf/conf_2.xml*)
   Config → Load local config file

Ilustración 1: Barra de herramientas

2. Hacer clic sobre el botón 

del asistente de configuración
3. Entrar en el modo simulación
   Actions → Start
4. Cargar un fichero de traza (*sjlocal/files/trace/*.trd*)
   Trace → Load local trace file
   NOTA: En la parte superior del diálogo de selección de fichero de traza ha de introducirse un valor en el campo *tUnits* (e.g. 1)

5. Iniciar la simulación

   En la pestaña *findPages* hacer clic sobre el botón 

o 

según se prefiera

Ilustración 2: Pestañas del simulador

1.3. Ayuda

La ayuda con la que cuenta el simulador actualmente no ofrece demasiada información al usuario. Se accede a ella a través de *Help → Help* en la barra de herramientas.

No obstante se comentan algunos aspectos que pueden resultar de interés como los formatos de los ficheros de configuración y traza y una concisa descripción de los campos que muestran algunas de las tablas del simulador.

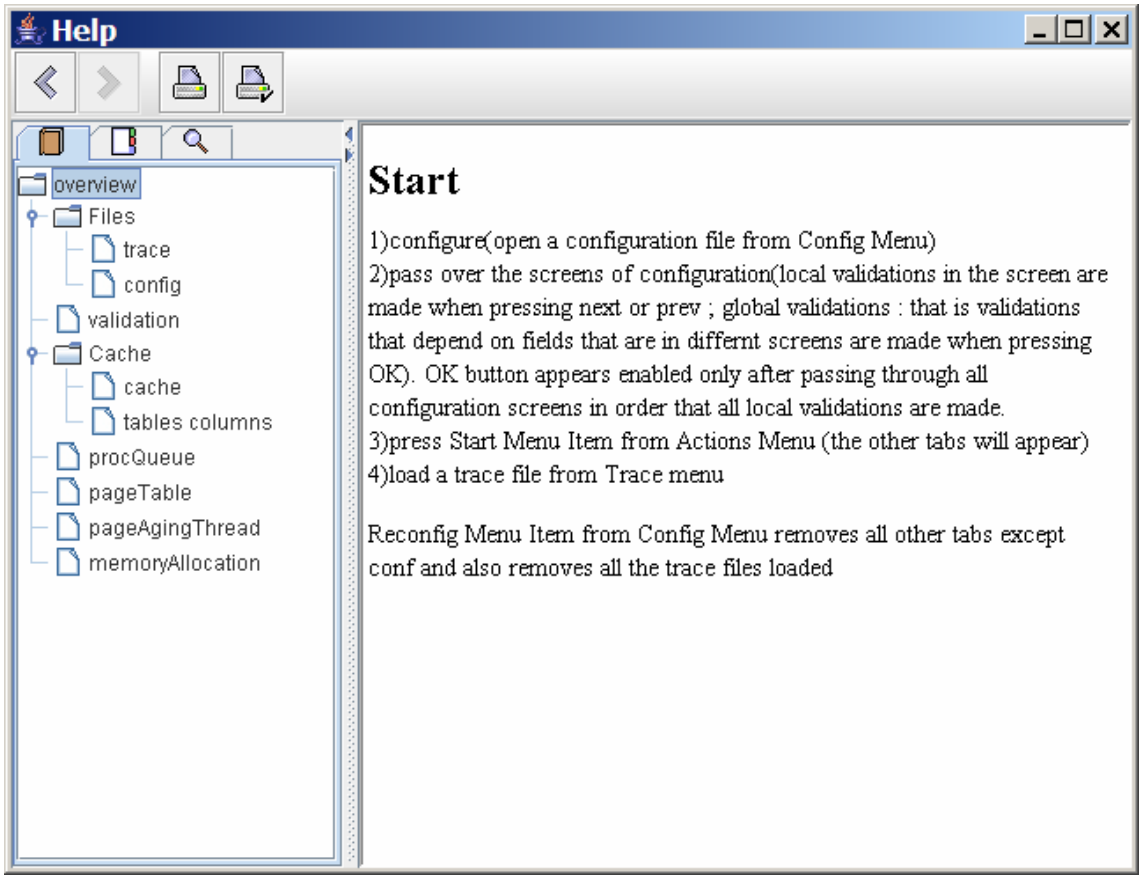


Ilustración 3: Ventana de ayuda

2. Configuración

2.1. Asistente de configuración

Mneme presenta al usuario el asistente de configuración al iniciar el programa. Se divide en tres apartados a través de los que se puede navegar utilizando los siguientes botones:

Ilustración 4: Botones del asistente de configuración

Se utilizan muchas abreviaturas para los distintos términos utilizados en el simulador, así como la notación ***2**n*** para indicar la n-ésima potencia de 2.

2.1.1. Primer apartado

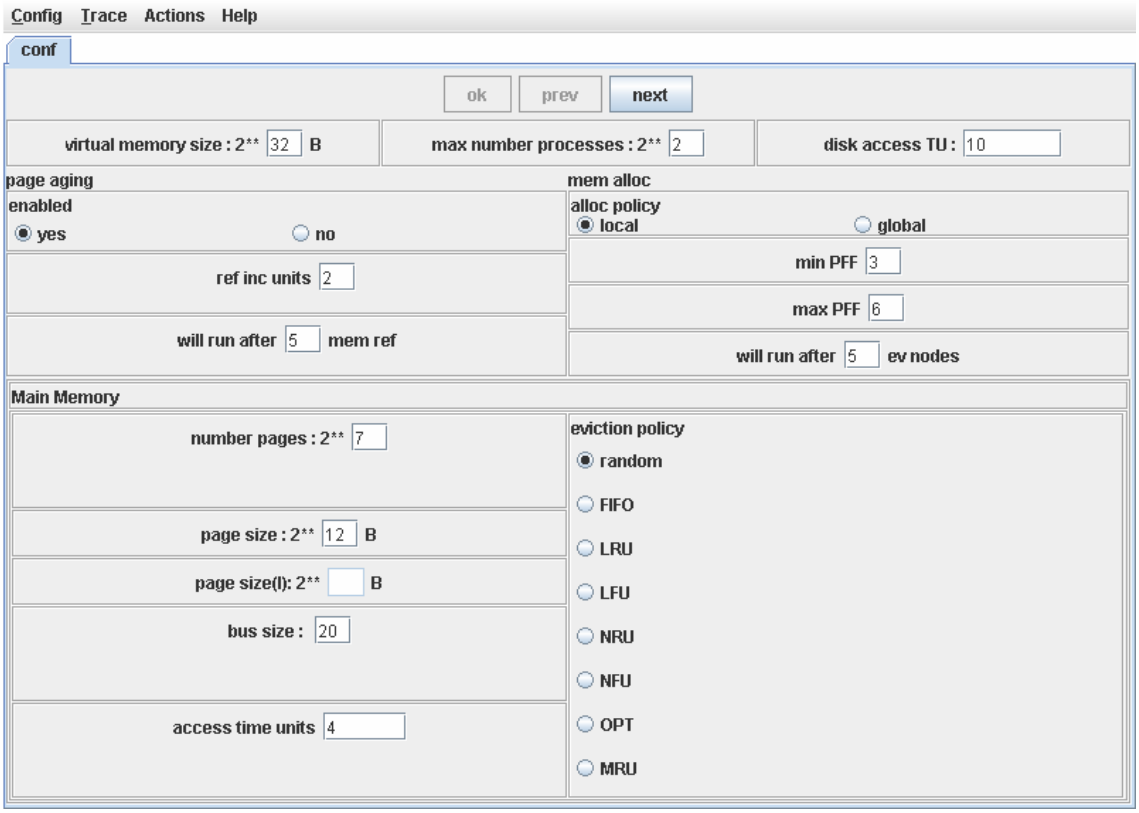


Ilustración 5: Primer apartado de configuración

- *Configuración general*
  - Tamaño de la memoria virtual
  - Número máximo de procesos que soportará el sistema
  - Unidades de tiempo que costará el acceso a disco

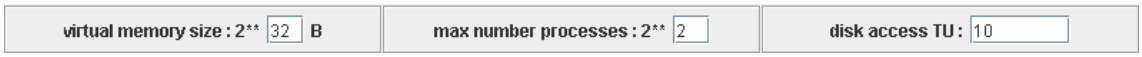


Ilustración 6: Configuración general

- *Envejecimiento de páginas*

- *Conceptos previos*

El envejecimiento de páginas es una técnica que se utiliza para eliminar páginas de la memoria principal, gracias a un thread (hilo de proceso) que es lanzado cada cierto número de referencias a memoria. Este thread incrementa o decrementa un contador asociado a cada página después de una o varias referencias a memoria (lecturas o escrituras). A las páginas que hayan sido referenciadas, se les incrementará la edad de página (*page age*), y a las que no hayan sido referenciadas, se les decrementará. Si una página alcanza un valor igual o inferior a cero es eliminada.

- *Opciones de configuración*
  - Activar o desactivar el envejecimiento de páginas (*page aging*)
  - Cantidad en que se incrementará la edad de la página (*ref inc units*)
  - Número de referencias a memoria principal (lectura o escritura) tras las que se ejecutará el thread que incremente la edad de las páginas (*page aging thread*)

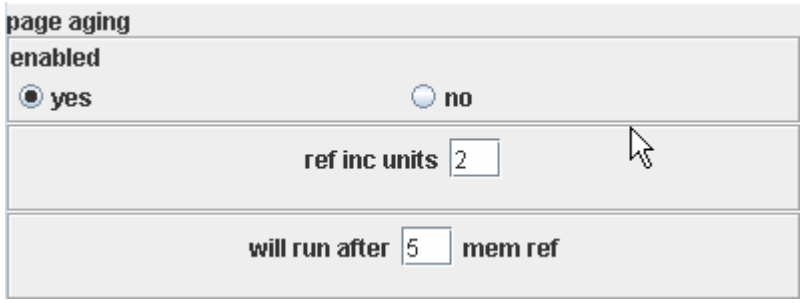


Ilustración 7: Envejecimiento de páginas

- *Asignación de memoria*

  - *Conceptos previos*

La *asignación de memoria* es un procedimiento por el cual se le asigna memoria principal a una página de un proceso. La memoria principal se encuentra divida en *marcos de página*, cuya finalidad es albergar una página de un proceso.

Hay dos políticas de asignación de memoria, *global* y *local*. En la *asignación global* las páginas se alojan en el primer marco de página que encuentren, y en caso de no encontrar ningún marco, eliminarán cualquier página de la memoria para que deje libre un marco de página.

La *asignación de memoria local*, es algo más compleja, ya que asigna un espacio de memoria principal (varios marcos de páginas) a cada proceso, y es en este espacio donde el proceso carga sus páginas. La cantidad de espacio de memoria principal que se le asigna a cada proceso, depende del tipo de asignación de espacio de memoria que usemos. Existen dos tipos de asignación de espacio de memoria, estática (asigna un número fijo de marcos a cada proceso) o dinámica (asigna un número variable de marcos según el proceso).

PFF (Page Fault Frequency) es un valor asociado a cada proceso, que nos indica la cantidad de fallos de página que ha tenido ese proceso. La finalidad de este valor, es el control de la tasa de fallos de páginas de la memoria. Cuantos menos fallos de página tenga un proceso, más rápido se ejecutará. Para controlar la tasa de fallos de página disponemos Mneme utiliza un thread que es lanzado cada cierto número de fallos de páginas (ev nodes), cuyo objetivo es analizar los PFF y actuar en consecuencia. En caso de que el PFF sea mayor que el máximo PFF permitido (maxPFF), se asignan marcos de páginas libres a ese proceso, y si es menor que el mínimo PFF permitido (minPFF) se liberan marcos de página pertenecientes a ese proceso usando un algoritmo de reemplazo.

- *Opciones de configuración*
  - Política de asignación local o global
  - Frecuencia de fallos de página mínima y máxima (para política local)
  - Número de páginas reemplazadas tras las que se ejecutará el thread de asignación de memoria

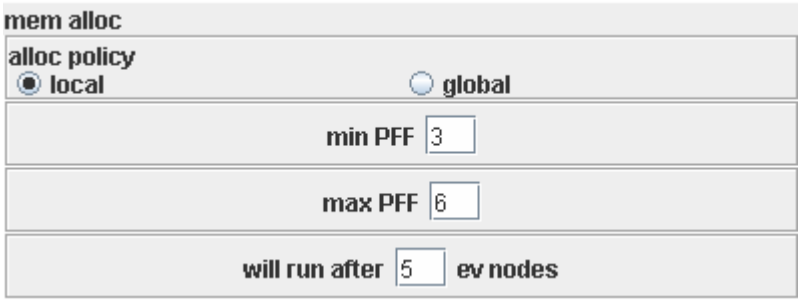


Ilustración 8: Configuración de la asignación de memoria

- *Memoria principal*

  - *Conceptos previos*

La memoria principal se divide en marcos de páginas y los procesos en páginas. Una página es un trozo de memoria contiguo que contiene información de un proceso, y cuyo tamaño debe ser igual o menor al tamaño de los marco de páginas. Los procesos que estén en ejecución tienen sus páginas cargadas en los marcos de la memoria principal, que luego son ejecutadas por la CPU.

Cuando un proceso necesita un marco de página y están todas ocupadas, debemos de liberar una página de la memoria principal. Para saber qué página debemos liberar, tenemos diversos algoritmos de reemplazo de páginas. Los algoritmos que se encuentran implementados en este programa, están descritos a continuación:

- Random (aleatorio): reemplaza una página al azar.
- FIFO (First In First Out, primero en entrar primero en salir): selecciona la página que más tiempo lleva en memoria.
- LRU (Less Recently Used, menos recientemente usado): reemplaza la página que lleva más tiempo sin ser referenciada.
- LFU (Less Frequently Used, menos frecuentemente usado): se reemplaza la página que menos veces ha sido referenciada.
- NRU (Not Used Recently, no usado recientemente): reemplaza la página que no haya sido usada recientemente. Para determinar esa situación se usan dos factores, si la página ha sido referenciada o si ha sido modificada.
- NFU (Not Frequently Used):
- OPT (Optimum, óptimo): reemplaza la página que más tiempo a estar sin ser referenciada. Este algoritmo es sólo teórico, ya que en la realidad no podemos predecir qué páginas va a necesitar un proceso.

- MRU (Most Recently Used, más recientemente usado): reemplaza la página a la que se accedió por última vez.

- *Opciones de configuración*
  - Número de páginas de la memoria principal
  - Tamaño de cada página de la memoria
  - Tamaño del bus
  - Unidades de tiempo que costará el acceso a la memoria principal
  - Política de reemplazo de páginas

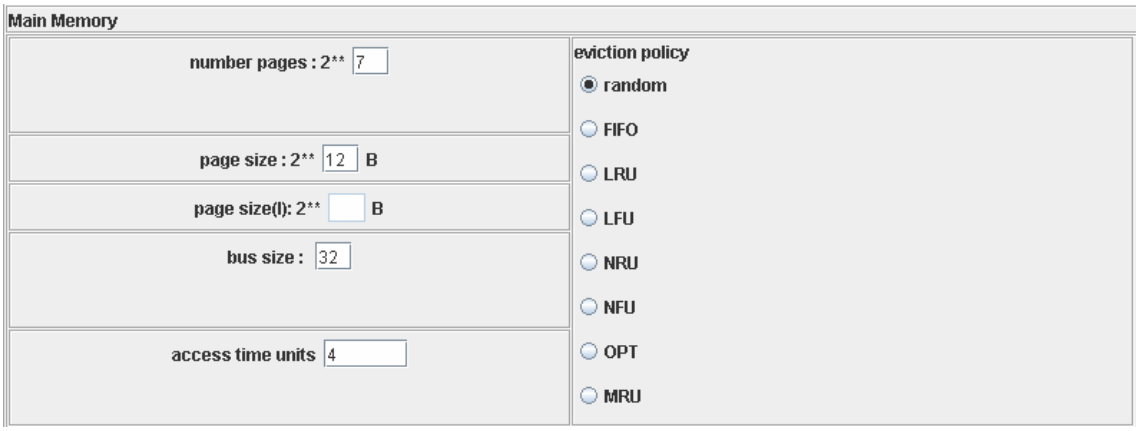


Ilustración 9: Configuración de la memoria principal

2.1.2. Segundo apartado

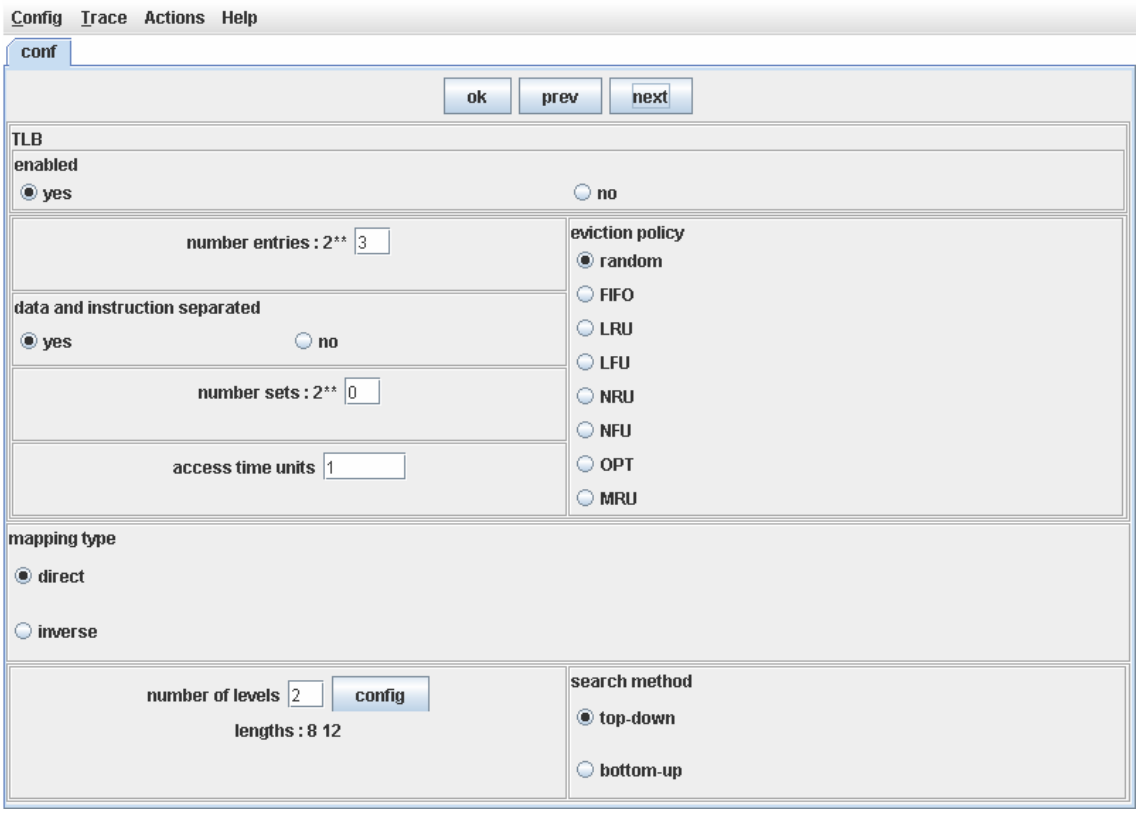


Ilustración 10: Segundo apartado de configuración

- *TLB*

  - *Conceptos previos*

TLB (*Translation Lookaside Buffer*, buffer de traducciones anticipadas) es un buffer que contiene partes de la tabla de paginación (tabla que contiene las relaciones entre dirección virtual y real) y es utilizada para la traducción rápida de direcciones. Una TLB es una memoria caché, por tanto, la manera en que se organiza son las mismas en la que se puede organizar una memoria caché: mapeado directo, totalmente asociativa, o asociativa por conjuntos (explicados en el tercer apartado, memoria caché).

El objetivo de una TLB es acelerar todo lo posible la traducción de direcciones, guardando las últimas traducciones que se han efectuado. Existen dos formas de implementar las traducciones de direcciones con TLB: directa e instrucciones y datos separados.

La traducción de direcciones directa con TLB, se ilustra en la siguiente imagen:

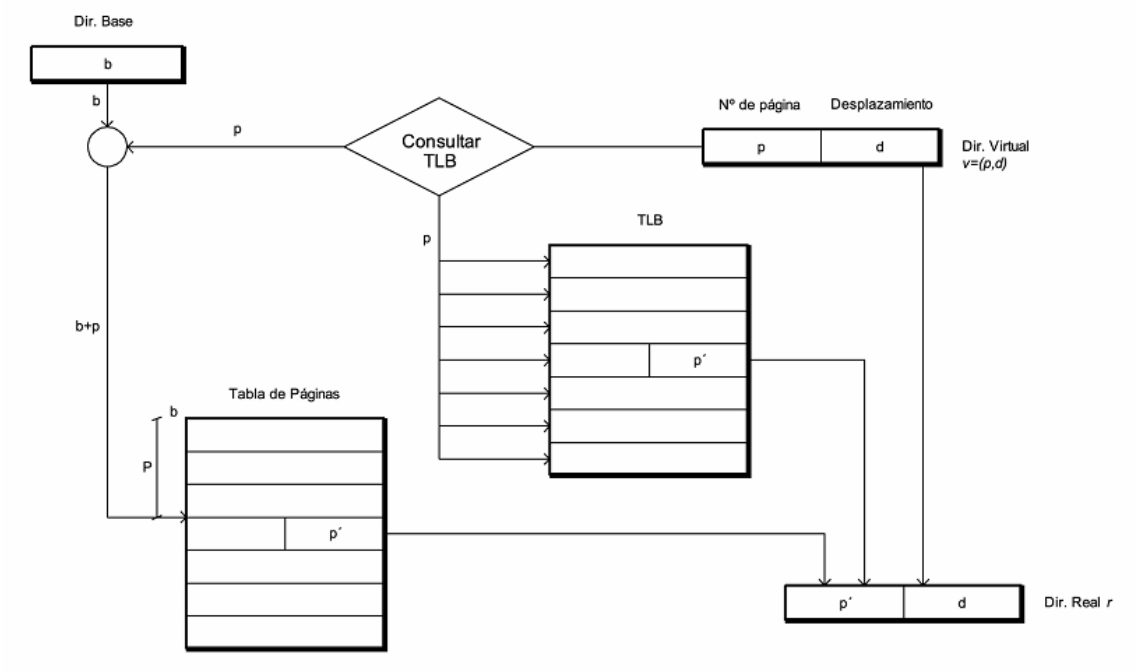


Ilustración 11: Traducción de direcciones directa con TLB

Primero se consulta la tabla de TLB para ver si existe la página. En caso de fallo, se busca en la tabla de páginas que se encuentra en memoria principal.

La segunda manera de implementar una tabla TLB, es separar las traducciones que se corresponden con datos, y las traducciones que se corresponden con instrucciones. La siguiente imagen muestra el funcionamiento de este método:

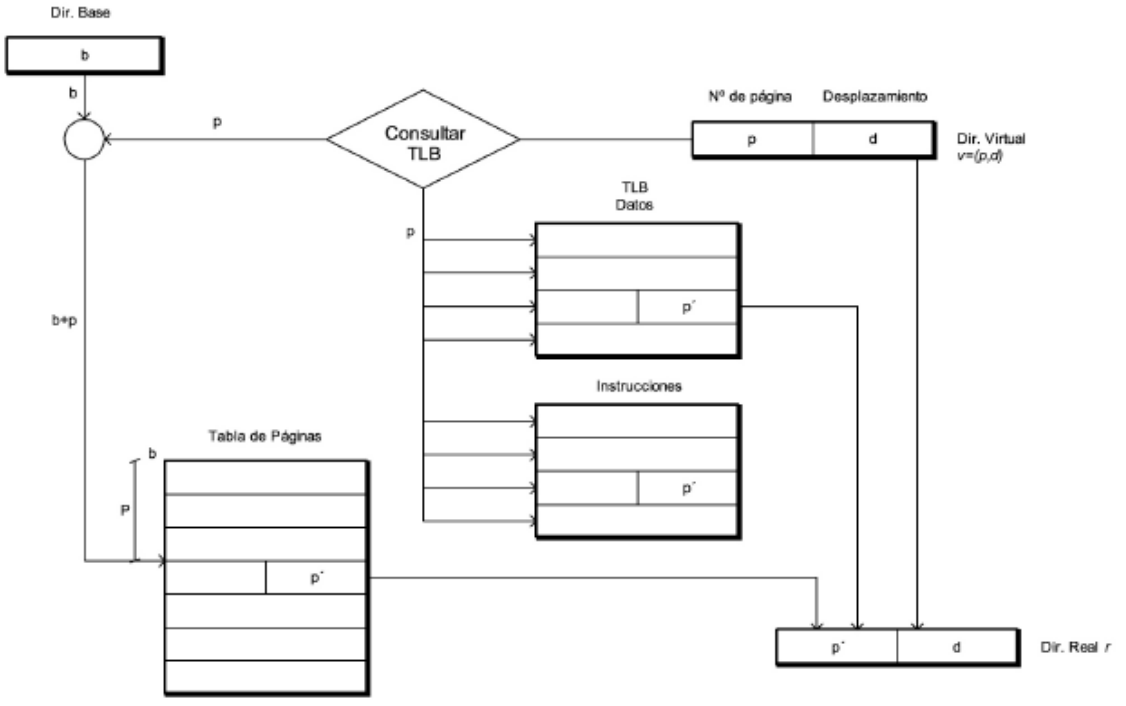


Ilustración 12: Traducción de páginas utilizando TLB datos e instrucciones

- *Opciones de configuración*
  - Activar o desactivar la TLB
  - Número de entradas
  - Separación de TLB para datos y para instrucciones
  - Número de conjuntos asociativos en la TLB
  - Unidades de tiempo que costará el acceso a la TLB
  - Política de reemplazamiento

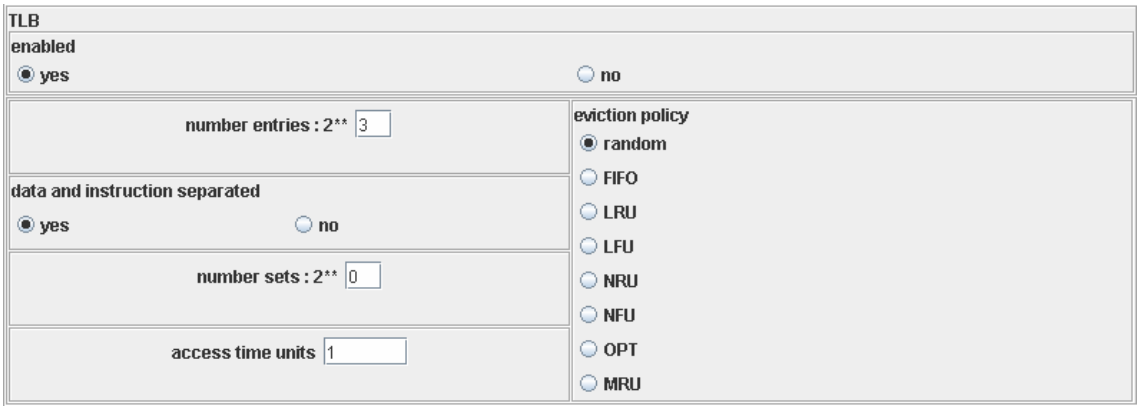


Ilustración 13: Configuración de la TLB

❑ *Tabla de páginas*

- *Conceptos previos*

Una tabla de páginas contiene la información necesaria que relaciona cada página de un proceso con el marco que la contiene. Cada proceso debe tener una tabla de páginas que le indique donde se encuentran sus páginas en cada momento. Esta forma de organizar la tabla de páginas, es denominada *Mapeado Directo*. En contraposición, el *Mapeado inverso*, la tabla de página relaciona cada marco de página con una página de un proceso, por tanto, el tamaño de la tabla es proporcional al de la memoria. Esta tabla es construida como una tabla hash, en el que las casillas son todos los marcos de página de la memoria principal. Esto se usa para reducir el tamaño de la tabla de páginas, ya que no contiene todas las páginas de los procesos, sino de las páginas que están en memoria.

Para ahorrar espacio en memoria podemos crear, usando mapeado directo, una tabla que se divida en varios niveles, llamada *Tablas Multinivel*. Cada nivel es un trozo de tabla, cuyas entradas apunta a otras tablas que pertenecen al nivel inferior. Sólo necesitaríamos tener la tabla de nivel superior cargada en memoria, y las otras tablas

de nivel inferior en disco y llamarlas bajo demanda según las necesitemos. Una tabla multinivel tiene la siguiente forma:

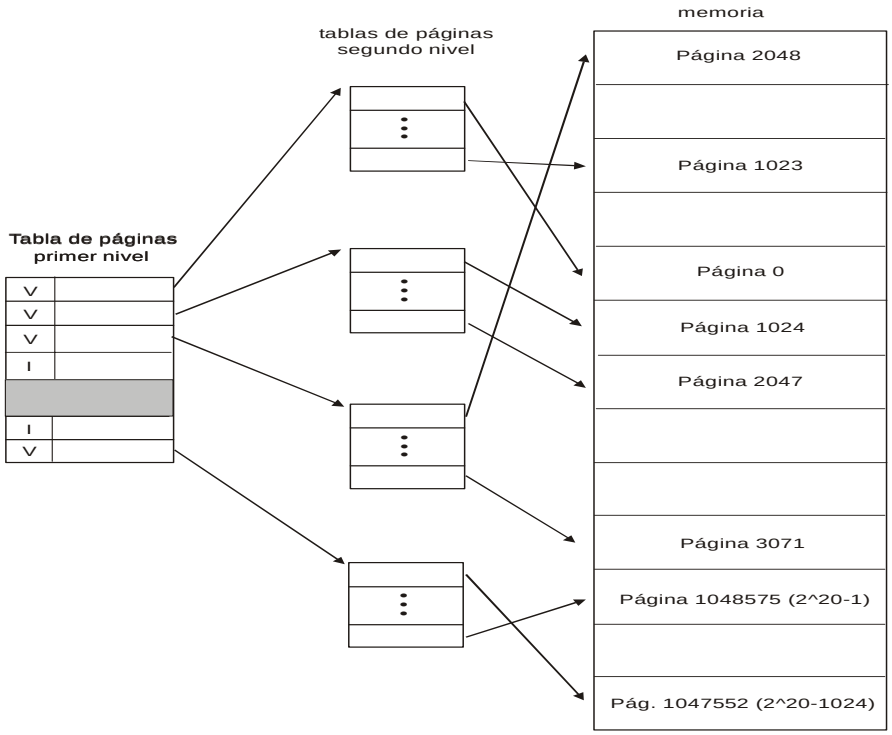


Ilustración 14: Tablas de página multinivel

A la hora de buscar una página en una tabla de páginas, podemos recorrerlos de dos modos: desde los niveles inferiores hacia los superiores (*bottom-up*) o desde los niveles superiores hacia los inferiores (*top-down*).

- *Opciones de configuración*
  - Tipo de mapeado: directo o inverso
  - Número de niveles para mapeado directo. Con longitudes configurables.
  - Método de búsqueda para mapeado directo: top-down o bottom-up
  - Número de páginas virtuales que pueden referenciar a la vez a la misma página física (para mapeado inverso).

Ilustración 15: Configuración de la tabla de páginas

2.1.3. Tercer apartado

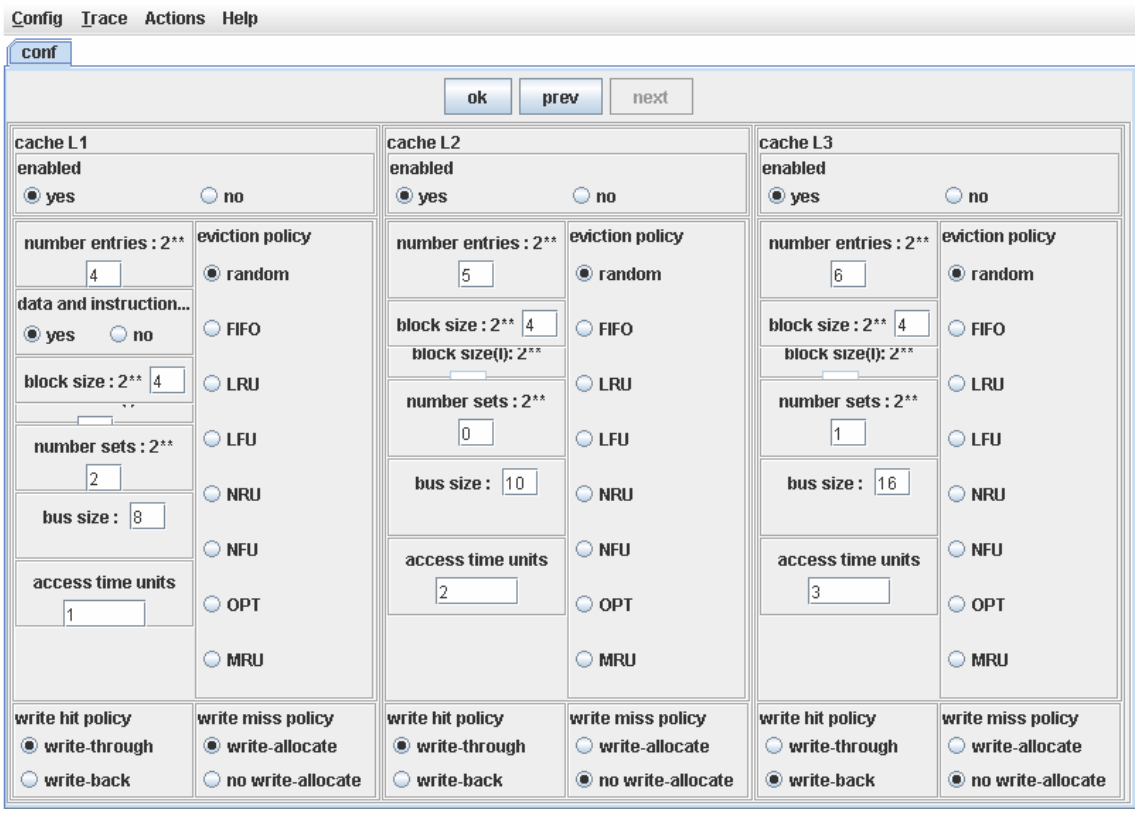


Ilustración 16: Tercer apartado de configuración

- *Memorias Cachés (L1, L2, L3)*

  - *Conceptos previos*

Como el acceso a la memoria principal es lento, se incluye una jerarquía de acceso rápido que se intercala entre la CPU y la memoria principal, llamada memorias caché. La organización de una memoria caché determina la posición en que un nueva página puede ser colocada dentro de la memoria caché. Así se distinguen tres tipos de estrategias de colocación:

  - Mapeado directo: Cada página de memoria principal tiene un solo lugar donde ser colocado en memoria caché.

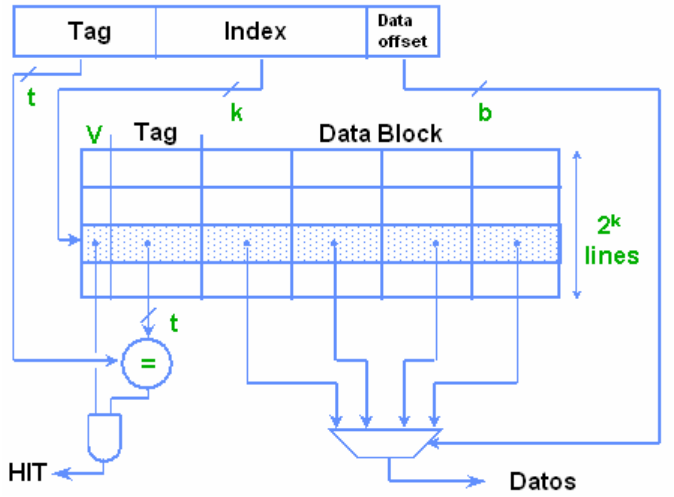


Ilustración 17: Mapeado Directo

- Memoria completamente asociativa: Un página de memoria principal puede ser colocado en cualquier lugar de la memoria caché.

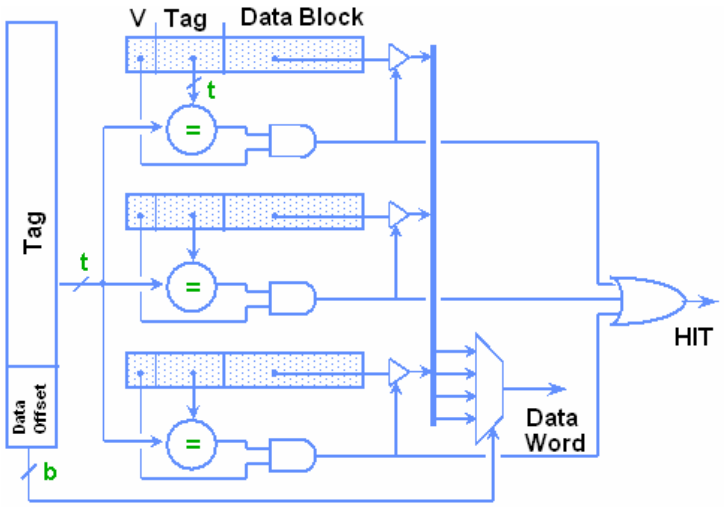


Ilustración 18: Caché completamente asociativa

- Memoria asociativa por conjuntos: A una página de memoria principal le corresponde un conjunto de marcos en memoria caché, de manera que puede ser colocado en cualquier marco dentro del conjunto.

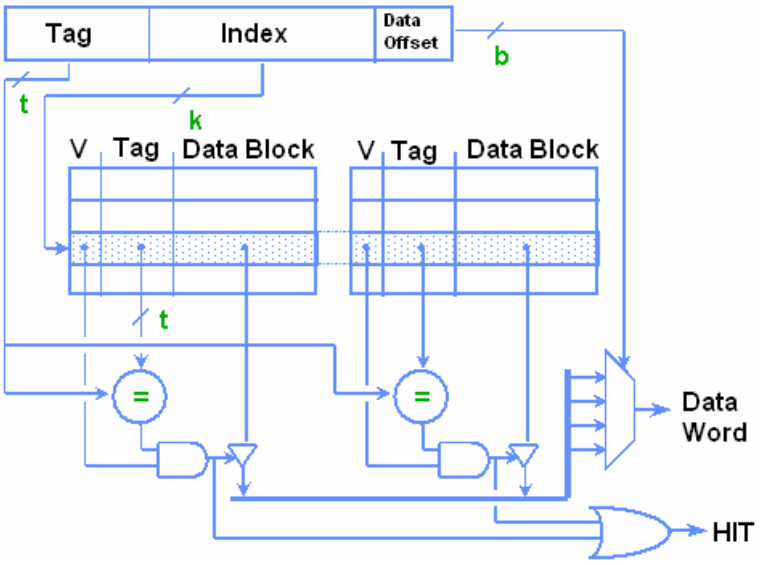


Ilustración 19: Caché asociativa por conjuntos

Mneme implementa hasta 3 niveles de memoria caché. Los niveles inferiores de caché, que estarían más cerca de la CPU, son más rápidos y de menor tamaño que las memorias caché superiores, que son algo más lentas pero de mayor tamaño. Una jerarquía de memorias que incluyera memoria caché de 3 niveles quedaría como la siguiente:

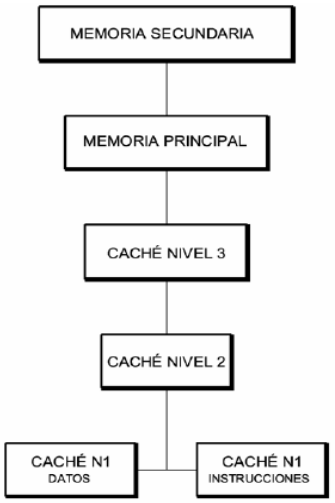


Ilustración 20: Jerarquía de memoria con 3 niveles de caché

Según se desciende en la jerarquía, el tamaño de las memorias disminuye pero su velocidad de acceso aumenta, siendo así las memorias de caché de nivel 1 las más rápidas y las que más cercas están de la CPU.

Una vez que la CPU ha ejecutado una instrucción, y en el caso de que se escriben o modifiquen algunos datos, debe actualizarse el nuevo valor de esos datos. Para ello se diferencian dos casos: cuando la posición de memoria sobre la que vamos a escribir se encuentra en la caché (acierto) o cuando no (fallo). Frente a los aciertos podemos distinguir dos maneras de actualizar los datos:

- *Write through* (escritura inmediata): Todas las operaciones pertinentes para la actualización de los datos, se realiza tanto en la memoria principal como en la memoria caché.
- *Copy Back* (postescritura): las actualizaciones sólo se realizan en la memoria caché. Cuando se realiza un reemplazo de una página, los datos son actualizados en la memoria principal.

Si existe fallo en la memoria caché, las maneras de actualizar los datos son:

- Write allocate (asignación en escritura): el bloque de la página donde se encuentran los datos, se carga en memoria caché cuando ocurre el fallo y luego es tratado con un write through o copy back.
- No write allocate (no asignación en escritura): los datos que pertenecen a la página se modifica en memoria principal sin ser cargados a memoria caché.

- *Opciones de configuración*

Por cada uno de los niveles de caché se pueden establecer los siguientes parámetros:

- Activar o desactivar la caché
- Número de entradas
- Tamaño del bloque
- Número de conjuntos
- Tamaño del bus
- Unidades de tiempo que costará el acceso a la caché
- Políticas ante un acierto *(write-hit)* y un fallo *(write-miss)*

Para la caché L1 puede además especificarse:

- Separación para datos e instrucciones

2.2. Ficheros de configuración

La configuración se almacena en ficheros XML. Si se incluyen en el fichero campos que no aparecen en el asistente de configuración serán ignorados.

Si algún parámetro no está definido en el fichero de configuración tomará su valor por defecto.

```
<config>

    <virtualAddressNBits>32</virtualAddressNBits>
    <numberProcessesNBits>2</numberProcessesNBits>
    <diskATU>10</diskATU>

    <pageTable direct="true">
        <offsetLengths>
            <length>10</length>
            <length>10</length>
        </offsetLengths>
        <searchMethod>topdown</searchMethod>
        <tlbConfig>
            <numberEntriesNBits>3</numberEntriesNBits>
            <evictionPolicy>RANDOM</evictionPolicy>
            <dataInstrSeparated>true</dataInstrSeparated>
            <accessTimeUnits>1</accessTimeUnits>
        </tlbConfig>
    </pageTable>

    <pageAgingConfig>
        <pageAgingIncrease>2</pageAgingIncrease>
        <memRefToRun>5</memRefToRun>
    </pageAgingConfig>
```

Ilustración 21: Fragmento de un fichero de configuración

3. Simulación

3.1. Ficheros de traza

El simulador admite la carga de ficheros de traza (*.trd*) que contienen las secuencias de instrucciones que se desean simular.

```
 1 003d49b0 MEMREAD  9952
 2 116f49a0 MEMWRITE 50
 3 22ba3c0 MEMREAD 5097
 4 311ba3c0 MEMWRITE 50
 5 442ba3b8 MEMREAD  2642
 6 22ba3c0 MEMREAD 5097
 7 0c3d49b0 MEMREAD  9952
 8 442ba3b8 MEMREAD  2642
 9 311ba3c0 MEMWRITE 50
10 116f49a0 MEMWRITE 50
```

Ilustración 22: Fichero de traza "test_opt.trd"

3.1.1. Formato

Cada línea de un fichero de traza debe seguir el siguiente formato:

Dirección virtual	***Tipo de instrucción***	***Tiempo***

- *Dirección virtual*: contiene una dirección virtual en hexadecimal de N bits que representa la dirección generada por el programa en ejecución
- *Tipo de instrucción*:
  - FETCH: búsqueda de instrucciones
  - MEMREAD: lectura de memoria (datos)
  - MEMWRITE: escritura en memoria (datos)
- *Tiempo*: actualmente el simulador Mneme **no** utiliza este parámetro

3.2. Interfaz de simulación

Una vez cargada la configuración para iniciar la simulación habrá que realizar los siguientes pasos:

1. Hacer clic sobre el botón 

del asistente de configuración
2. Entrar en el modo simulación

   Actions → Start
3. Cargar un fichero de traza (*sjlocal/files/trace/*.trd*)

   Trace → Load local trace file

   NOTA: En la parte superior del diálogo de selección de fichero de traza ha de introducirse un valor en el campo *tUnits* (e.g. 1)

Llegados a este punto se activarán las pestañas que corresponden a la simulación.

Ilustración 23: Pestañas del simulador

3.2.1. Memoria principal y cachés

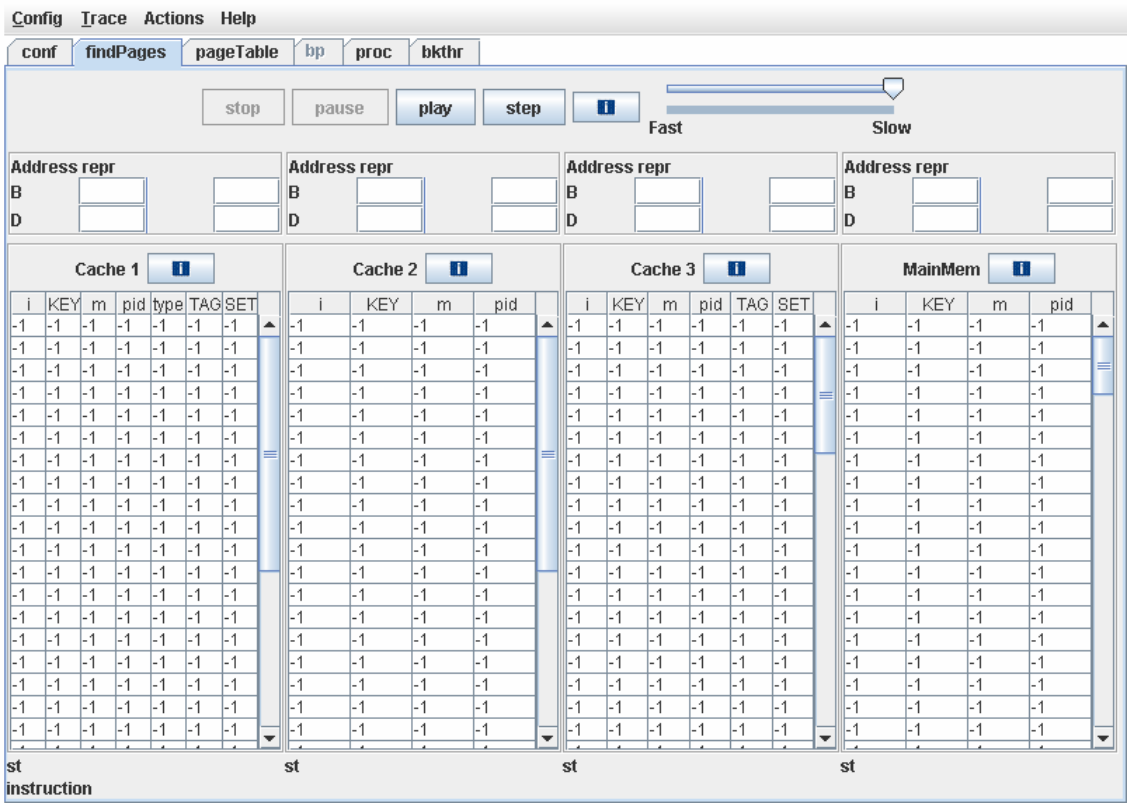


Ilustración 24: Memoria principal y cachés

❑ *Control de la simulación*

Mediante cuatro botones y una barra de desplazamiento se puede controlar:

- *Step*: simulación paso a paso
- *Play*: simulación completa a la velocidad que indique la barra de desplazamiento
- *Pause*: pausa la simulación
- *Stop*: interrumpe la simulación

Ilustración 25: Botones para controlar la simulación

El botón  que se encuentra a la derecha de  muestra información referente a la memoria virtual, número máximo de procesos y tiempo de acceso a disco.

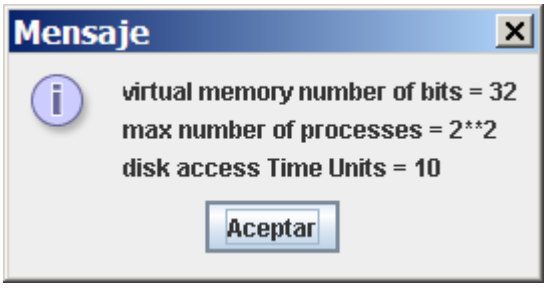


Ilustración 26: Diálogo información de los botones de simulación

En la parte inferior izquierda del panel se muestra el último tipo de instrucción que ha terminado de ejecutarse (*FETCH, MEMREAD, MEMWRITE*).

❑ *Memoria principal*

La primera parte en que se divide la visión de la memoria principal es la representación de las direcciones (*Address repr*).

En ella se muestra la última dirección consultada dividiéndola en dos partes, la de la derecha son los bits que referencian el bloque y la de la izquierda los bits restantes. Se representa tanto en binario (B) como decimal (D).

Ilustración 27: Representación de las direcciones

La segunda parte describe el contenido de la memoria principal utilizando una tabla con los siguientes campos:

- *i:* número de la entrada
- *KEY:* número de página
- *m:* bit de modificación. (1 página modificada, 0 no modificado)
- *pid:* número del proceso que referencia a esa página

Existen además una serie de campos que aparecerán dependiendo de la política de reemplazamiento utilizada. Esto es válido tanto para la memoria principal como para las cachés:

- *ni:* sólo con el algoritmo OPT (Óptimo), número de instrucción del proceso actual cuando el bloque o página es almacenado en la caché.
- *Timestamp:* marca de tiempo utilizada en el algoritmo FIFO (First In First Out)
- *notUsed:* contador de veces que se han referenciado otros bloques de la caché, para los algoritmos MRU (Most Recently Used) y LRU (Least Recently Used)
- *used:* contador de veces que el bloque ha sido referenciado, para los algoritmos LFU (Least Frecuently Used) y NFU (Not Frecuently Used).
- *r:* para la política NRU (Not Recently Used) es 1 si el bloque es leído, 0 en otro caso. Para la política NFU (Not Frecuently Used) es 1 si el bloque es referenciado, 0 en otro caso.

Cuando se ejecuta el algoritmo de reemplazamiento se tendrán en cuenta los siguientes factores para elegir el bloque o página a ser sustituido:

- Si las instrucciones y datos están separados

- Si la memoria es asociativa por conjuntos se buscará el bloque/página a ser reemplazado en el mismo conjunto en el que caiga el nuevo bloque/página a almacenar.
- Si la política de asignación de memoria es local se tendrán en cuenta sólo las páginas de memoria del mismo proceso.

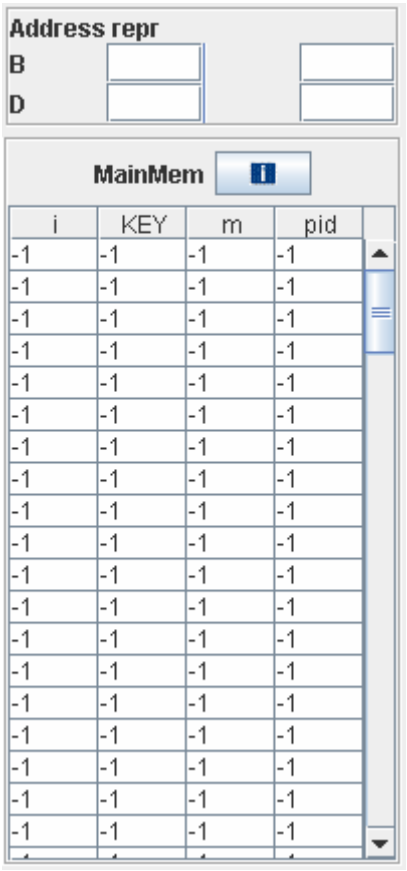


Ilustración 28: Tabla de memoria principal

El botón  que se encuentra a la derecha de la palabra *MainMem* presenta un resumen de las opciones configuradas para la memoria principal.

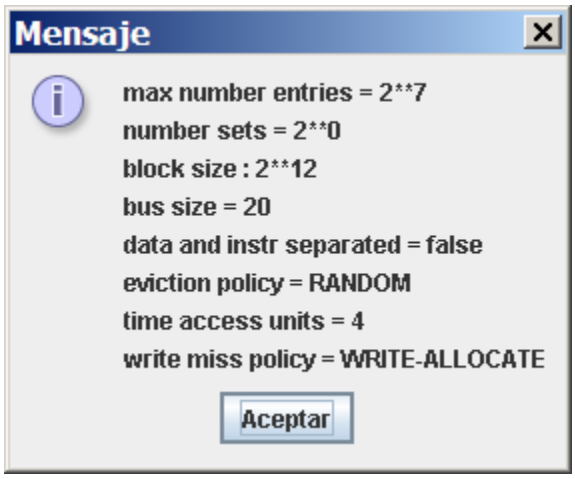


Ilustración 29: Resumen opciones configuración para memoria principal

Mneme puede presentar además una serie de estadísticas relacionadas con la traza que se ha ejecutado haciendo clic sobre el texto *"st"* que se encuentra en la parte inferior izquierda de la tabla que representa la memoria.

Ilustración 30: Estadísticas para una de las memorias

- *Memorias cachés (L1, L2, L3)*

La visión que ofrece Mneme para las memorias cachés es análoga a la de la memoria principal teniendo en cuenta que los campos de las tablas pueden diferir según la configuración de cada nivel de caché:

- *i:* número de entrada en la caché
- *KEY:* dirección de la caché
- *m:* bit de modificación. (1 bloque modificado, 0 no modificado)
- *pid:* número del proceso que referencia al bloque

Los siguientes campos aparecerán en función del tipo de caché:

- *type:* sólo aparece cuando en la caché se diferencian datos e instrucciones (L1)

- *TAG, SET:* las direcciones se descomponen en TAG|SET|OFFSET.
  Las cachés totalmente asociativas sólo tienen un conjunto.
  Las directamente mapeadas tienen por cada entrada en la caché un conjunto asociativo (SET) diferente.
  La entrada KEY está compuesta por TAG y SET y representa el número de página/bloque real. Este campo no aparece en los modelos de caché totalmente asociativas.

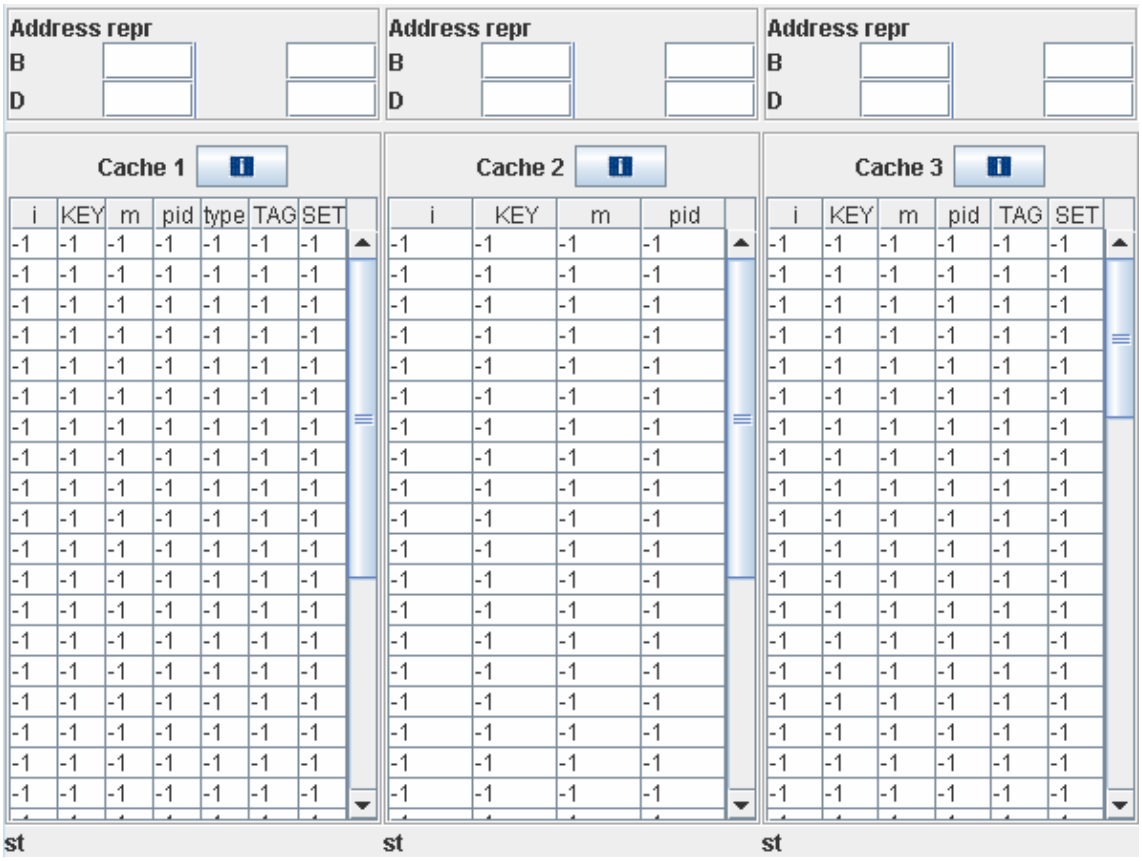


Ilustración 31: Representación de las memorias cachés

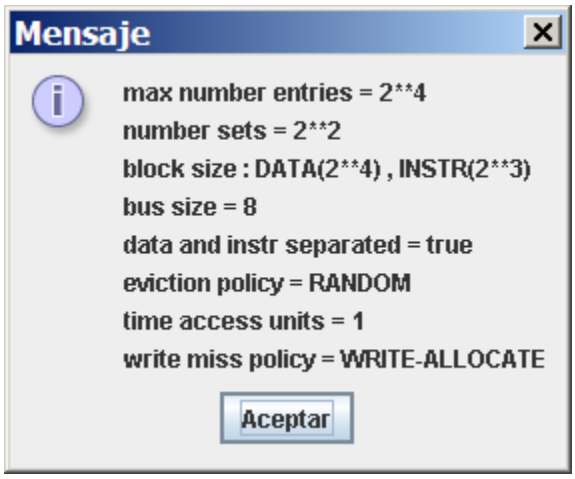


Ilustración 32: Diálogo de información de la caché L1

- *Código de colores*

Tanto para la memoria principal como para las memorias cachés el simulador utiliza un código de colores para representar cada suceso en la tabla.

Cada vez que sucede un evento se selecciona la fila correspondiente en la tabla y se le asocia un código de color de entre los descritos a continuación:

	Instrucción FETCH/MEMREAD y el bloque ya está en la caché
	Instrucción MEMWRITE y el bloque ya está en la caché
	Cuando el bloque todavía no está en la caché
	Antes de reemplazar un bloque
	Antes de eliminar un bloque

Tabla 1: Código de colores utilizados en las tablas de las memorias

3.2.2. Gráfico de simulación

En la pestaña *"bp"* Mneme muestra una representación gráfica de la jerarquía de memoria de la máquina.

Haciendo clic en el botón 

comenzará una animación que recrea las operaciones llevadas a cabo durante el último paso de simulación indicando el

intercambio de información que tiene lugar entre los diferentes niveles de memoria y la *CPU*.

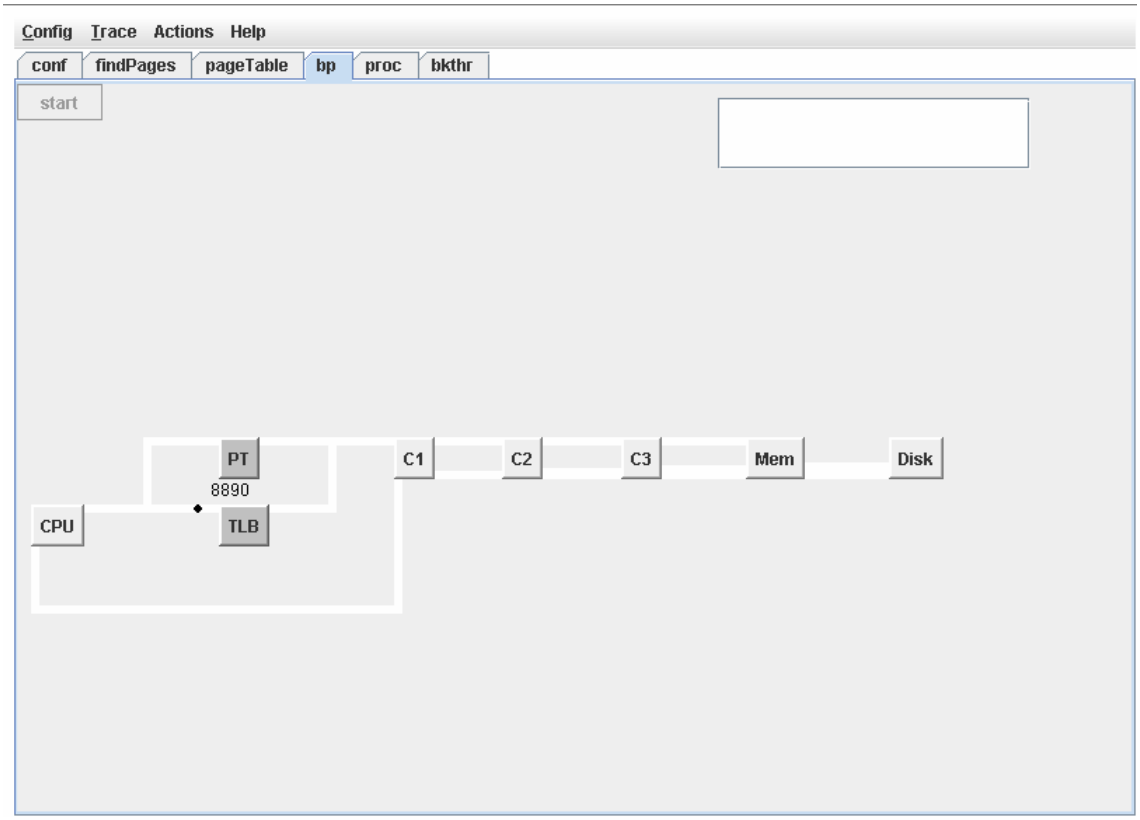


Ilustración 33: Panel de gráfico de simulación

3.2.3. Tabla de páginas

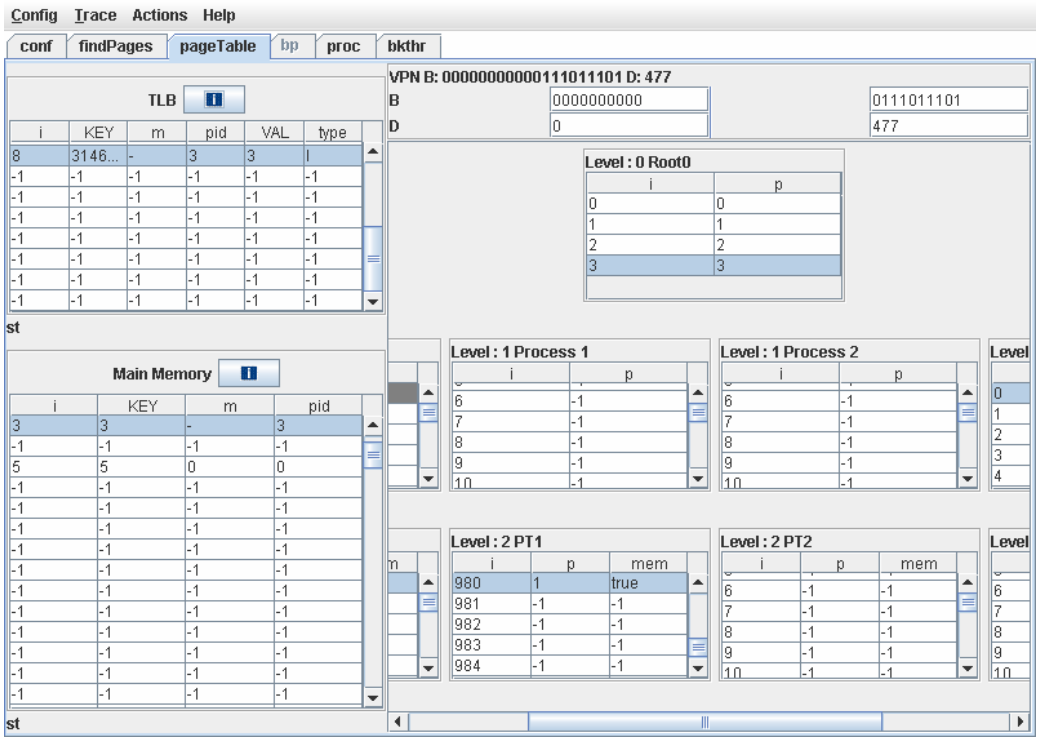


Ilustración 34: Panel de tabla de páginas utilizando mapeado directo de 2 niveles

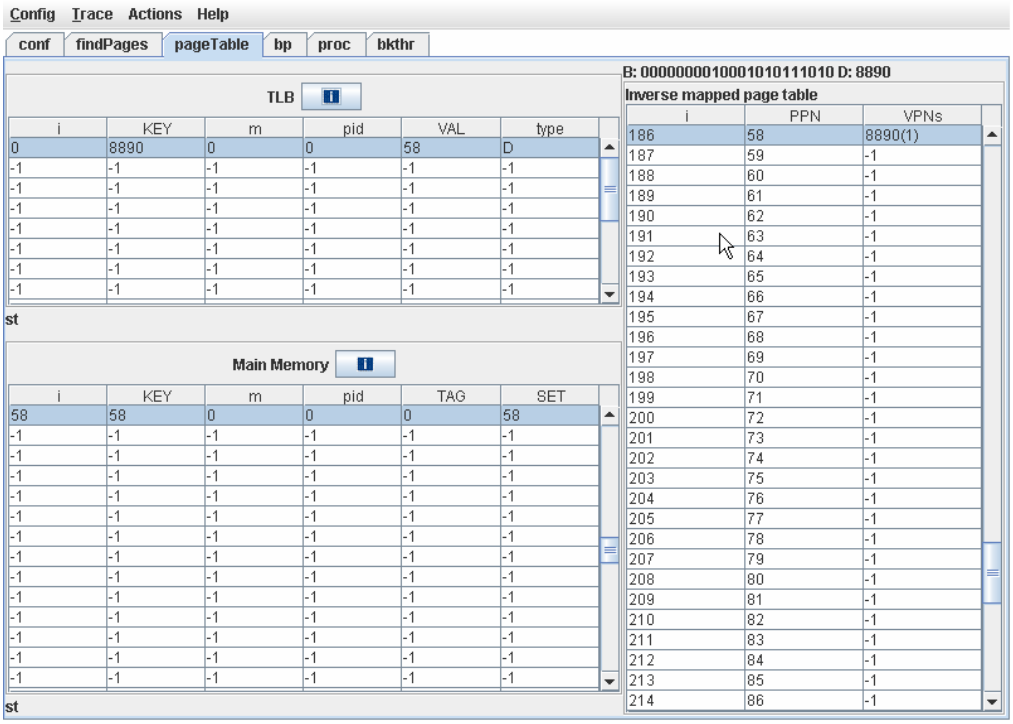


Ilustración 35: Panel de tabla de páginas utlizando mapeado inverso

- *Dirección virtual*

En la zona superior derecha del panel se muestra la dirección virtual generada por el proceso en ejecución.

Ilustración 36: Dirección virtual dividida para mapeado directo de 2 niveles (10, 10)

En el caso de utilizar mapeado directo se mostrará la dirección virtual (VPN) dividida en tantas partes como niveles se hayan configurado.

El número de bits de la VPN que se le asigne a cada nivel puede ser configurado respetando que la suma de todos estos bits sea igual al número de páginas virtuales (*Virtual Memory Size / Page Size*).

Ilustración 37: Dirección virtual para mapeado inverso

La dirección se muestra tanto en binario (B) como en decimal (D).

- *Tablas de páginas*

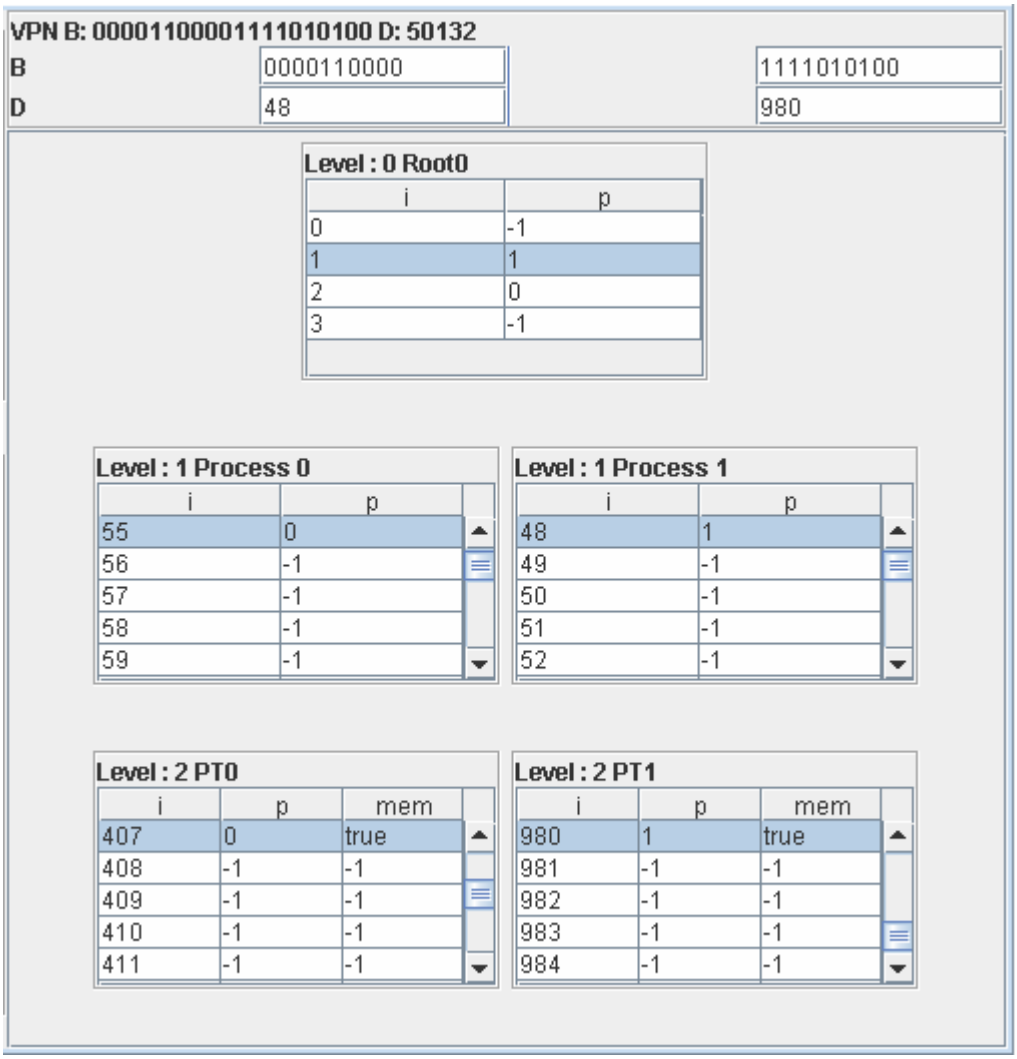
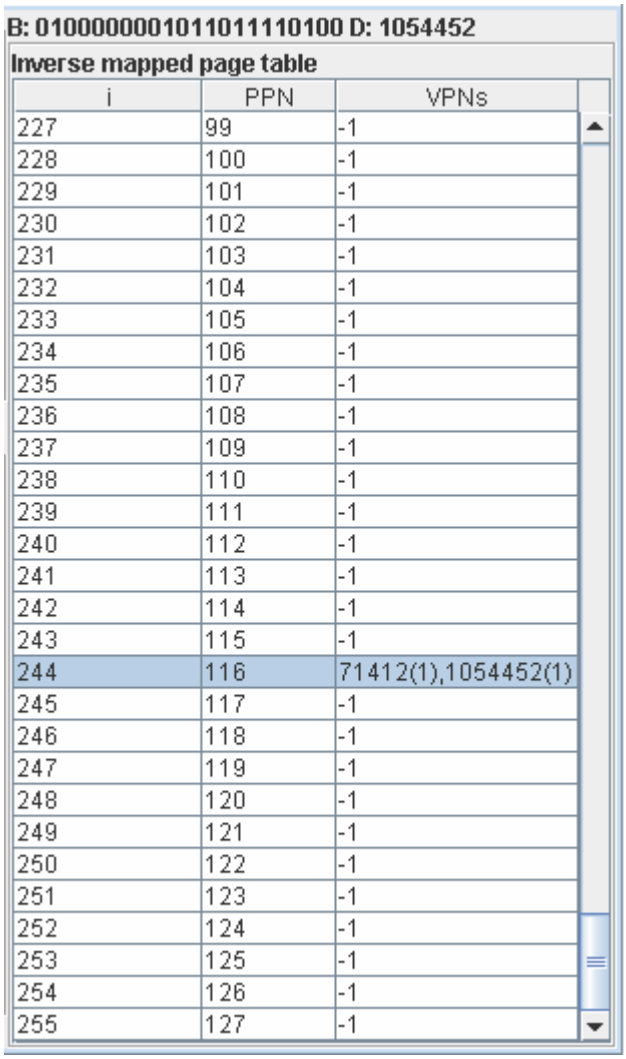


Ilustración 38: Tabla de páginas con mapeado directo de 2 niveles (10, 10)

Cuando se utiliza mapeado directo, el funcionamiento en el ejemplo de la ilustración anterior para la traducción de direcciones es el siguiente:

1. Se divide la dirección virtual en 10 y 10 bits (tantas divisiones como niveles de páginas haya)
2. La tabla de nivel 0 denominada *Root* siempre está presente e indica la tabla de nivel 1 que utiliza el proceso con *PID* indicado en el campo *i*.
3. A continuación se busca la entrada *48* (que corresponde al primer nivel) en la tabla que se obtuvo en el paso anterior (*Level: 1 Process 1*).
4. Se repite el paso previo pero esta vez sobre el siguiente nivel. En el ejemplo se busca la entrada *980* en (*Level: 2 PT 1*). En esta caso se obtiene en el campo *p* la página de la memoria a la que corresponde la *VPN* (página 1).



B: 01000000010110111110100 D: 1054452

Inverse mapped page table

i	PPN	VPNs
227	99	-1
228	100	-1
229	101	-1
230	102	-1
231	103	-1
232	104	-1
233	105	-1
234	106	-1
235	107	-1
236	108	-1
237	109	-1
238	110	-1
239	111	-1
240	112	-1
241	113	-1
242	114	-1
243	115	-1
244	116	71412(1),1054452(1)
245	117	-1
246	118	-1
247	119	-1
248	120	-1
249	121	-1
250	122	-1
251	123	-1
252	124	-1
253	125	-1
254	126	-1
255	127	-1

Ilustración 39: Tabla de páginas inversa

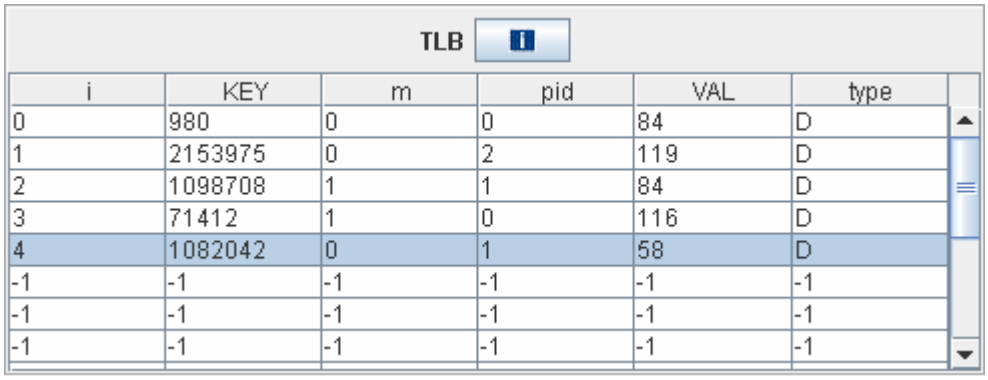
En el caso de que se haya decidido utilizar mapeado inverso, el funcionamiento en el ejemplo de la ilustración anterior para la traducción de direcciones es el siguiente:

1. Se tiene un *hash* que relaciona las páginas de memoria de los procesos con las direcciones virtuales. El número de entradas de la tabla *hash* puede modificarse en el asistente de configuración.
2. En el ejemplo la lista de direcciones virtuales (*71412, 1054452*) corresponden con la página 116 de la memoria.

- *TLB*

La TLB utiliza la misma representación que las memorias cachés tratada en la sección 3.2.1 Memoria principal y cachés.

La única diferencia es que se indexa a través del campo *KEY* utilizando direcciones virtuales y que además proporciona en el campo *VAL* la página de memoria a la que corresponde la dirección virtual.



i	KEY	m	pid	VAL	type
0	980	0	0	84	D
1	2153975	0	2	119	D
2	1098708	1	1	84	D
3	71412	1	0	116	D
4	1082042	0	1	58	D
-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1

Ilustración 40: TLB

- *Memoria principal*

La representación de la memoria principal se trata con detalle en el apartado 3.2.1 Memoria principal y cachés.

3.2.4. Procesos

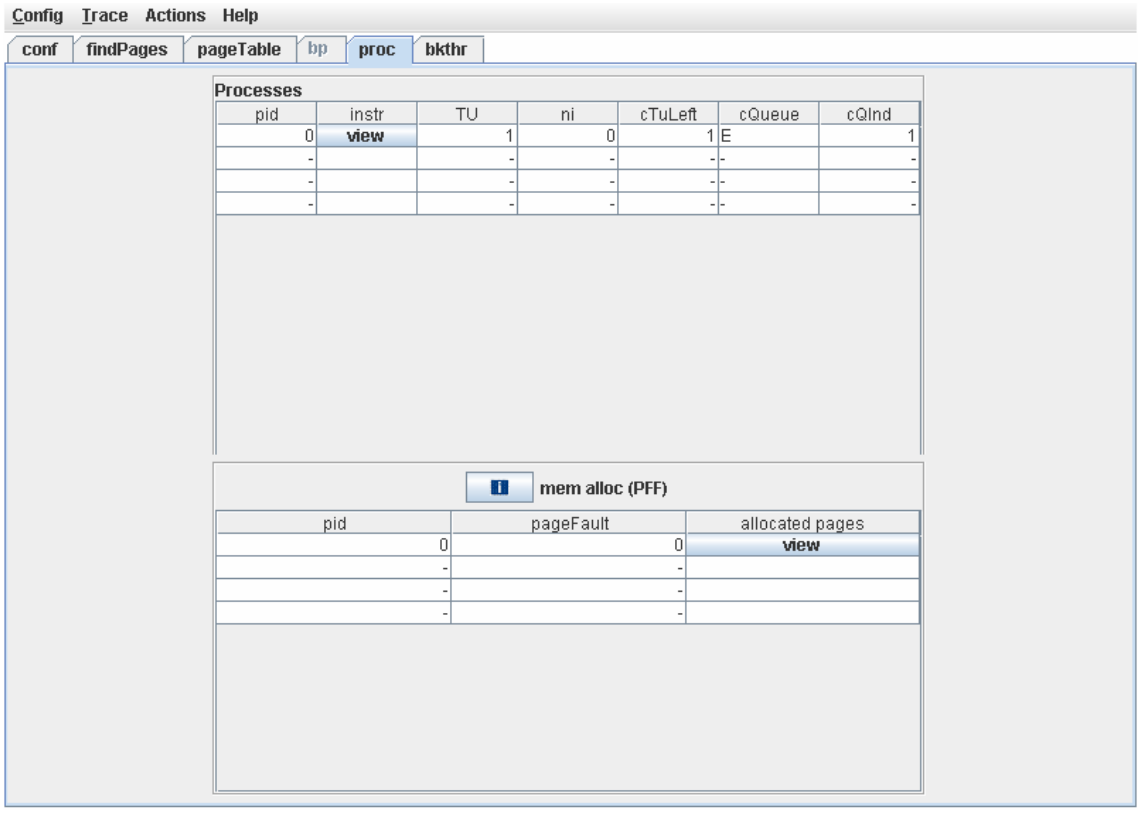


Ilustración 41: Panel de procesos

Mneme soporta la ejecución de múltiples procesos al mismo tiempo. Para ello basta con cargar diferentes ficheros de traza (3.1 Ficheros de traza).

Los procesos pueden estar en dos colas: espera y ejecución. Cuando un proceso pasa a la cola de ejecución permanecerá en este estado tantas unidades de tiempo como indique *tUnits* (que se establece al cargar un fichero de traza) o hasta que provoque un fallo de página.

Cuando un proceso está en ejecución las unidades de tiempo que transcurran serán decrementadas de su campo *tUnits* y también del campo *tUnits* del primer proceso de la cola de espera.

Cuando las *tUnits* del proceso en ejecución lleguen a 0 el proceso pasa a ser el último elemento en la cola de ejecución.

Cuando ocurre un fallo de página, el proceso en ejecución pasa a ser el último en la cola de espera estableciendo sus *tUnits* al número de unidades de tiempo que toma el acceso a disco.

Finalmente, cuando las *tUnits* del primer proceso de la cola de espera lleguen a cero pasará a ser el último elemento de la cola de ejecución.

- *Procesos*

Los procesos cargados pueden verse en la parte superior de este panel donde muestra además una serie de campos con información relevante:

- *pid:* identificador del proceso
- *instr:* al hacer clic sobre view muestra las instrucciones del proceso precedidas por su número.
- *TU:* unidades de tiempo configuradas al cargar el fichero de traza
- *ni:* número de instrucción actual del proceso
- *cQueue:* cola en la que se encuentra el proceso (E para ejecución, W para espera)
- *cQInd:* lugar que ocupa en la cola en la que se encuentra
- *cTuLeft:* unidades de tiempo que tiene el proceso

Processes

pid	instr	TU	ni	cTuLeft	cQueue	cQInd
2	view	5	1	5	E	1
1	view	2	0	4	W	1
0	view	1	0	10	W	2
-		-	-	-	-	-

Ilustración 42: Tabla de procesos

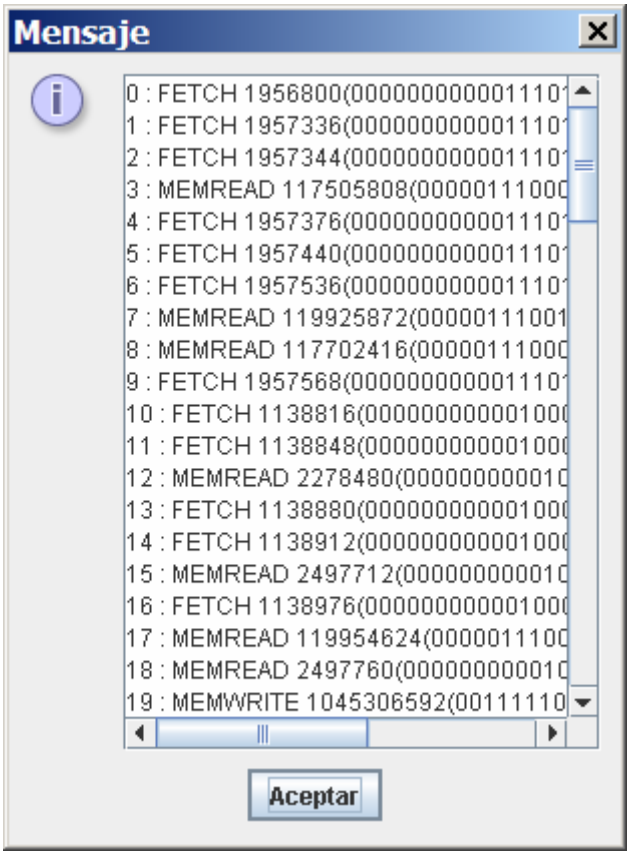


Ilustración 43: Instrucciones de un proceso al hacer clic sobre view

❑ *Alojamiento en memoria de procesos*

La zona inferior del panel de procesos muestra información sobre las páginas que tiene alojadas:

- *pid:* identificador del proceso
- *pageFault:* fallos de página provocados por el proceso
- *allocated pages:* al hacer clic sobre view muestra un diálogo con las páginas que tiene el proceso alojadas en memoria.

Haciendo clic en el botón  junto a *"mem alloc (PFF)"* se muestra un resumen de la configuración de la política de alojamiento de memoria.

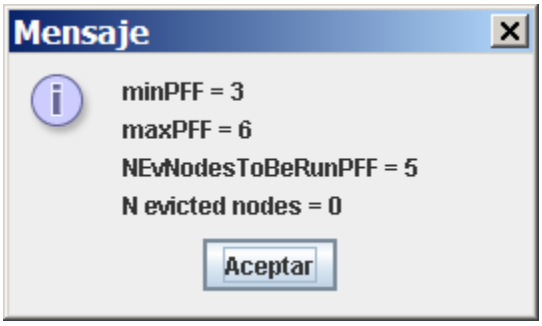


Ilustración 44: Diálogo resumen de la configuración de la política de alojamiento

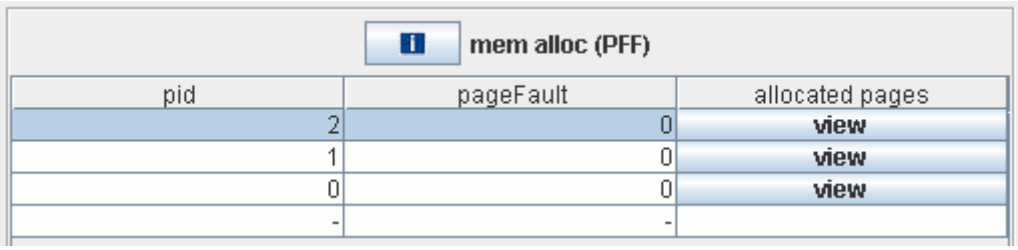


Ilustración 45: Alojamiento de memoria de los procesos

3.2.5. Bkthr

Este pestaña no tiene demasiada utilidad y únicamente muestra unos cuadros de información sobre el estado de la traza. En concreto información sobre el thread de page aging e información sobre las páginas alojadas del proceso.

Ilustración 46: Pestaña Bkthr

Índice de Ilustraciones

Ilustración 1: Barra de herramientas *5*
Ilustración 2: Pestañas del simulador *5*
Ilustración 3: Ventana de ayuda *6*
Ilustración 4: Botones del asistente de configuración *7*
Ilustración 5: Primer apartado de configuración *7*
Ilustración 6: Configuración general *8*
Ilustración 7: Envejecimiento de páginas *8*
Ilustración 8: Configuración de la asignación de memoria *10*
Ilustración 9: Configuración de la memoria principal *11*
Ilustración 10: Segundo apartado de configuración *12*
Ilustración 11: Traducción de direcciones directa con TLB *13*
Ilustración 12: Traducción de páginas utilizando TLB datos e instrucciones *13*
Ilustración 13: Configuración de la TLB *14*
Ilustración 14: Tablas de página multinivel *15*
Ilustración 15: Configuración de la tabla de páginas *16*
Ilustración 16: Tercer apartado de configuración *17*
Ilustración 17: Mapeado Directo *18*
Ilustración 18: Caché completamente asociativa *18*
Ilustración 19: Caché asociativa por conjuntos *19*
Ilustración 20: Jerarquía de memoria con 3 niveles de caché *19*
Ilustración 21: Fragmento de un fichero de configuración *21*
Ilustración 22: Fichero de traza "test_opt.trd" *22*
Ilustración 23: Pestañas del simulador *23*
Ilustración 24: Memoria principal y cachés *23*
Ilustración 25: Botones para controlar la simulación *24*
Ilustración 26: Diálogo información de los botones de simulación *24*
Ilustración 27: Representación de las direcciones *25*
Ilustración 28: Tabla de memoria principal *26*
Ilustración 29: Resumen opciones configuración para memoria principal *27*
Ilustración 30: Estadísticas para una de las memorias *27*
Ilustración 31: Representación de las memorias cachés *28*
Ilustración 32: Diálogo de información de la caché L1 ..... *29*
Ilustración 33: Panel de gráfico de simulación *30*
Ilustración 34: Panel de tabla de páginas utilizando mapeado directo de 2 niveles *31*
Ilustración 35: Panel de tabla de páginas utlizando mapeado inverso *31*

Ilustración 36: Dirección virtual dividida para mapeado directo de 2 niveles (10, 10) *32*
Ilustración 37: Dirección virtual para mapeado inverso *32*
Ilustración 38: Tabla de páginas con mapeado directo de 2 niveles (10, 10) *33*
Ilustración 39: Tabla de páginas inversa *34*
Ilustración 40: TLB *35*
Ilustración 41: Panel de procesos *35*
Ilustración 42: Tabla de procesos *37*
Ilustración 43: Instrucciones de un proceso al hacer clic sobre view *37*
Ilustración 44: Diálogo resumen de la configuración de la política de alojamiento *38*
Ilustración 45: Alojamiento de memoria de los procesos *38*
Ilustración 46: Pestaña Bkthr *38*

Índice de Tablas

Tabla 1: Código de colores utilizados en las tablas de las memorias 29