Trabajo Práctico 2
Memoria Caché

Jonathan Medina, *Padrón 100052 Slack: U011C1EGWMD*
`jcmedina@fi.uba.ar`

Pablo Inoriza, *Padrón 94986 Slack: U011Q8GB197*
`pinoriza@fi.uba.ar`

Christian Flórez Del Carpio, *Padrón 91011 Slack: U011G8YJ18T*
`chflorez@fi.uba.ar`

1do. Cuatrimestre de 2020
86.37 / 66.20 Organización de Computadoras – Práctica Jueves
Facultad de Ingeniería, Universidad de Buenos Aires

1. Introducción

Se realizo en lenguaje C, una implementación de una memoria caché asociativa por conjunto de cuatro vías, de 4KB de capacidad, bloques de 128 bytes, política de reemplazo LRU, y política de escritura WT/¬WA. El espacio de direcciones es de 16 bits, y además se simulo una memoria principal con tamaño de 64KB. Los bloques de cache cuentan con su meta data, es decir: tag, bit V. Para implementar el LRU se optó con utilizar un contador por conjunto.

2. Diseño e Implementación

El diseño que se opto para simular una memoria caché que tiene las siguientes características:

- asociativa por conjuntos de cuatro vías
- política de reemplazo LRTU
- política de escritura WT/¬WA

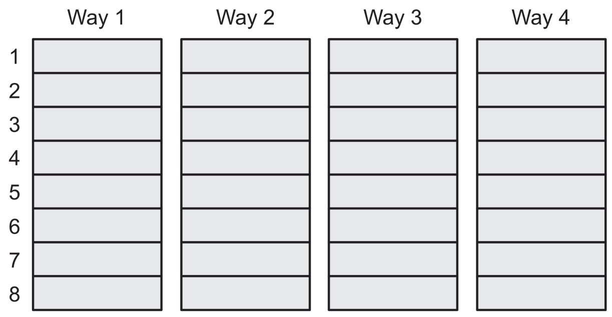

Figura 1: Cache asociativa por conjuntos

Para poder lograr toda está estructura de caché codeada en lenguaje C, implementamos estructuras de tipo struct.

Se creo la **caché** de tipo struct con los siguientes atributos:

- amount access
- amount misses
- conjuntos

Cada **conjunto** es de tipo struct con los siguientes atributos:

- bloquesCache

Donde bloquesCache es un vector de tipo **Struct bloqueCache**. de tamaño 4 (porqué la cache es de asociatividad grado 4).

El tipo struct bloqueCache tiene los siguientes atributos:

- V
- tag
- datos
- ultimamenteUsado

Donde el atributo **datos** es de tipo **bloqueDeMemoria**, el cual es un vector estatico de tipo **unsigned char** de un tamaño fijo de 128 posiciones. El atributo ultimamenteUsado se define para simular la política de reemplazo tipo LRU, el cual va incrementando su valor a medida que es usado, quedando el valor mas alto, el mas reciente utilizado esto es logrado gracias a que a nivel conjunto tenemos un contador que es incrementado cada vez que se accede a un conjunto de la cache, por lo tanto de esta manera contabalizamos las veces que fueron accedidos, y podemos reemplazar el bloque que fue accedido mas antiguamente.

Tambien se definio la memoria principal como un struct el cual tiene como atributo un vector de tipo bloqueDeMemoria con un tamaño fijo de 512 posiciones.

3. Proceso de Compilación

Para la compilación se cuenta con un makefile cuyos comandos son los siguientes:

- **make** (compila el proyecto)
- **make run** (corre la aplicación con cada uno de los archivos de prueba e imprime sus respectivos resultados. Es necesario tener compilado antes el proyecto para ejecutar este comando)

4. Portabilidad

Dado que se utilizó el lenguaje de programación C, sin hacer uso de funciones específicas de sistemas operativos, o de bibliotecas comerciales, los programas pueden ser compilados en varios de ellos. De todas formas, el Makefile presentado fue hecho particularmente para sistemas de tipo UNIX.

Los casos de prueba que seran presentados fueron llevados acabo en Ubuntu-Linux en una arquitectura Intel x86 satisfactoriamente

5. Casos de Prueba

5.1. Prueba con archivo con comando con mas parametros de lo permitido

```
$ cat prueba_rota_1.mem
W 0, 255, 1
MR

$ ./tp2 prueba_rota_1.mem
Error en el procesamiento del archivo


$ cat prueba_rota_3.mem
FLUSH 0

$ ./tp2 prueba_rota_3.mem
Error en el procesamiento del archivo
```

5.2. Prueba con archivo con comandos incorrectos

```
$ cat prueba_rota_2.mem
BROKEN 0, 255

$ ./tp2 prueba_rota_2.mem
```

```
Error en el procesamiento del archivo
```

5.3. Prueba con archivo con parametros incorrectos

```
$ cat prueba_rota_4.mem
R a31072
R 4096
R 8192
R 4096
R 0
R 4096
MR

$ ./tp2 prueba_rota_4.mem
Error en el procesamiento del archivo
```

5.4. Prueba con archivo con parametros vacios

```
$ cat prueba_rota_5.mem
W , 123

$ ./tp2 prueba_rota_5.mem
Error en el procesamiento del archivo
```

5.5. Prueba con archivo con comando con menos parametros de lo permitido

```
$ cat prueba_rota_6.mem
W 0

$ ./tp2 prueba_rota_6.mem
Error en el procesamiento del archivo
```

5.6. Prueba con archivo inexistente

```
$ cat prueba_archivo_no_existe.mem
cat: prueba_archivo_no_existe.mem: No such file or directory

$ ./tp2 prueba_archivo_no_existe.mem
Error archivo inexistente
```

5.7. Prueba con archivo vacio

```
$ echo "" > prueba_archivo_vacio.mem

$ cat prueba_archivo_vacio.mem

$ ./tp2 prueba_archivo_vacio.mem
Error en el procesamiento del archivo
```

5.8. Prueba correr el programa sin parametros

```
$ ./tp2
You need to send an argument for the name of the file
```

5.9. Prueba de ejecucion de archivo con valgrind: Catedra

Prueba 1

```
$ cat prueba1.mem
W 0, 255
W 1024, 254
W 2048, 248
W 4096, 096
W 8192, 192
R 0
R 1024
R 2048
R 8192
MR

$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./tp2 prueba1.mem
==18661== Memcheck, a memory error detector
==18661== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==18661== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==18661== Command: ./tp2 prueba1.mem
==18661==

Writing byte 255 with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't write to cache 0 tag, 0 set, 0 offset

Writing byte 254 with: 1 tag, 0 set, 0 offset. Address: 1024
Couldn't write to cache 1 tag, 0 set, 0 offset

Writing byte 248 with: 2 tag, 0 set, 0 offset. Address: 2048
Couldn't write to cache 2 tag, 0 set, 0 offset

Writing byte 96 with: 4 tag, 0 set, 0 offset. Address: 4096
Couldn't write to cache 4 tag, 0 set, 0 offset

Writing byte 192 with: 8 tag, 0 set, 0 offset. Address: 8192
Couldn't write to cache 8 tag, 0 set, 0 offset

Reading byte with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 0 to cache with way: 3, set: 0
Read from cache the byte: 255

Reading byte with: 1 tag, 0 set, 0 offset. Address: 1024
Couldn't find address in cache
Oldest for set: 0 way: 2
Taking blocknum 8 to cache with way: 2, set: 0
Read from cache the byte: 254

Reading byte with: 2 tag, 0 set, 0 offset. Address: 2048
Couldn't find address in cache
Oldest for set: 0 way: 1
Taking blocknum 16 to cache with way: 1, set: 0
Read from cache the byte: 248
```

```
Reading byte with: 8 tag, 0 set, 0 offset. Address: 8192
Couldn't find address in cache
Oldest for set: 0 way: 0
Taking blocknum 64 to cache with way: 0, set: 0
Read from cache the byte: 192
Missrate: 1.000000
==18661==
==18661== HEAP SUMMARY:
==18661==     in use at exit: 0 bytes in 0 blocks
==18661==   total heap usage: 556 allocs, 556 frees, 72,382 bytes allocated
==18661==
==18661== All heap blocks were freed -- no leaks are possible
==18661==
==18661== For counts of detected and suppressed errors, rerun with: -v
==18661== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Prueba 2

```
$ cat prueba2.mem
R 0
R 127
W 128, 10
R 128
W 128, 20
R 128
MR


$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./tp2 prueba2.mem
==18849== Memcheck, a memory error detector
==18849== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==18849== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==18849== Command: ./tp2 prueba2.mem
==18849==

Reading byte with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 0 to cache with way: 3, set: 0
Read from cache the byte: 0

Reading byte with: 0 tag, 0 set, 127 offset. Address: 127
Read from cache the byte: 0

Writing byte 10 with: 0 tag, 1 set, 0 offset. Address: 128
Couldn't write to cache 0 tag, 1 set, 0 offset

Reading byte with: 0 tag, 1 set, 0 offset. Address: 128
Couldn't find address in cache
Oldest for set: 1 way: 3
Taking blocknum 1 to cache with way: 3, set: 1
Read from cache the byte: 10

Writing byte 20 with: 0 tag, 1 set, 0 offset. Address: 128
Hit writing cache 0 tag, 1 set, 0 offset

Reading byte with: 0 tag, 1 set, 0 offset. Address: 128
Read from cache the byte: 20
Missrate: 0.500000
==18849==
==18849== HEAP SUMMARY:
==18849==     in use at exit: 0 bytes in 0 blocks
==18849==   total heap usage: 556 allocs, 556 frees, 72,342 bytes allocated
==18849==
==18849== All heap blocks were freed -- no leaks are possible
==18849==
==18849== For counts of detected and suppressed errors, rerun with: -v
==18849== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Prueba 3

```
$ cat prueba3.mem
W 128, 1
W 129, 2
W 130, 3
W 131, 4
R 1152
R 2176
R 3200
R 4224
R 128
R 129
R 130
R 131
MR


$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./tp2 prueba3.mem
==18943== Memcheck, a memory error detector
==18943== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==18943== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==18943== Command: ./tp2 prueba3.mem
==18943==

Writing byte 1 with: 0 tag, 1 set, 0 offset. Address: 128
Couldn't write to cache 0 tag, 1 set, 0 offset

Writing byte 2 with: 0 tag, 1 set, 1 offset. Address: 129
Couldn't write to cache 0 tag, 1 set, 1 offset

Writing byte 3 with: 0 tag, 1 set, 2 offset. Address: 130
Couldn't write to cache 0 tag, 1 set, 2 offset

Writing byte 4 with: 0 tag, 1 set, 3 offset. Address: 131
Couldn't write to cache 0 tag, 1 set, 3 offset

Reading byte with: 1 tag, 1 set, 0 offset. Address: 1152
Couldn't find address in cache
Oldest for set: 1 way: 3
Taking blocknum 9 to cache with way: 3, set: 1
Read from cache the byte: 0

Reading byte with: 2 tag, 1 set, 0 offset. Address: 2176
Couldn't find address in cache
Oldest for set: 1 way: 2
Taking blocknum 17 to cache with way: 2, set: 1
Read from cache the byte: 0

Reading byte with: 3 tag, 1 set, 0 offset. Address: 3200
Couldn't find address in cache
Oldest for set: 1 way: 1
```

```
Taking blocknum 25 to cache with way: 1, set: 1
Read from cache the byte: 0

Reading byte with: 4 tag, 1 set, 0 offset. Address: 4224
Couldn't find address in cache
Oldest for set: 1 way: 0
Taking blocknum 33 to cache with way: 0, set: 1
Read from cache the byte: 0

Reading byte with: 0 tag, 1 set, 0 offset. Address: 128
Couldn't find address in cache
Oldest for set: 1 way: 3
Taking blocknum 1 to cache with way: 3, set: 1
Read from cache the byte: 1

Reading byte with: 0 tag, 1 set, 1 offset. Address: 129
Read from cache the byte: 2

Reading byte with: 0 tag, 1 set, 2 offset. Address: 130
Read from cache the byte: 3

Reading byte with: 0 tag, 1 set, 3 offset. Address: 131
Read from cache the byte: 4
Missrate: 0.750000
==18943==
==18943== HEAP SUMMARY:
==18943==     in use at exit: 0 bytes in 0 blocks
==18943==   total heap usage: 556 allocs, 556 frees, 72,388 bytes allocated
==18943==
==18943== All heap blocks were freed -- no leaks are possible
==18943==
==18943== For counts of detected and suppressed errors, rerun with: -v
==18943== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Prueba 4

```
$ cat prueba4.mem
W 0, 255
W 1, 2
W 2, 3
W 3, 4
W 4, 5
R 0
R 1
R 2
R 3
R 4
R 4096
R 8192
R 2048
R 1024
R 0
R 1
R 2
R 3
R 4
MR


$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./tp2 prueba4.mem
==19018== Memcheck, a memory error detector
```

```
==19018== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==19018== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==19018== Command: ./tp2 prueba4.mem
==19018==

Writing byte 255 with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't write to cache 0 tag, 0 set, 0 offset

Writing byte 2 with: 0 tag, 0 set, 1 offset. Address: 1
Couldn't write to cache 0 tag, 0 set, 1 offset

Writing byte 3 with: 0 tag, 0 set, 2 offset. Address: 2
Couldn't write to cache 0 tag, 0 set, 2 offset

Writing byte 4 with: 0 tag, 0 set, 3 offset. Address: 3
Couldn't write to cache 0 tag, 0 set, 3 offset

Writing byte 5 with: 0 tag, 0 set, 4 offset. Address: 4
Couldn't write to cache 0 tag, 0 set, 4 offset

Reading byte with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 0 to cache with way: 3, set: 0
Read from cache the byte: 255

Reading byte with: 0 tag, 0 set, 1 offset. Address: 1
Read from cache the byte: 2

Reading byte with: 0 tag, 0 set, 2 offset. Address: 2
Read from cache the byte: 3

Reading byte with: 0 tag, 0 set, 3 offset. Address: 3
Read from cache the byte: 4

Reading byte with: 0 tag, 0 set, 4 offset. Address: 4
Read from cache the byte: 5

Reading byte with: 4 tag, 0 set, 0 offset. Address: 4096
Couldn't find address in cache
Oldest for set: 0 way: 2
Taking blocknum 32 to cache with way: 2, set: 0
Read from cache the byte: 0

Reading byte with: 8 tag, 0 set, 0 offset. Address: 8192
Couldn't find address in cache
Oldest for set: 0 way: 1
Taking blocknum 64 to cache with way: 1, set: 0
Read from cache the byte: 0

Reading byte with: 2 tag, 0 set, 0 offset. Address: 2048
Couldn't find address in cache
Oldest for set: 0 way: 0
Taking blocknum 16 to cache with way: 0, set: 0
Read from cache the byte: 0

Reading byte with: 1 tag, 0 set, 0 offset. Address: 1024
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 8 to cache with way: 3, set: 0
Read from cache the byte: 0
```

```
Reading byte with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't find address in cache
Oldest for set: 0 way: 2
Taking blocknum 0 to cache with way: 2, set: 0
Read from cache the byte: 255

Reading byte with: 0 tag, 0 set, 1 offset. Address: 1
Read from cache the byte: 2

Reading byte with: 0 tag, 0 set, 2 offset. Address: 2
Read from cache the byte: 3

Reading byte with: 0 tag, 0 set, 3 offset. Address: 3
Read from cache the byte: 4

Reading byte with: 0 tag, 0 set, 4 offset. Address: 4
Read from cache the byte: 5
Missrate: 0.578947
==19018==
==19018== HEAP SUMMARY:
==19018==     in use at exit: 0 bytes in 0 blocks
==19018==   total heap usage: 556 allocs, 556 frees, 72,405 bytes allocated
==19018==
==19018== All heap blocks were freed -- no leaks are possible
==19018==
==19018== For counts of detected and suppressed errors, rerun with: -v
==19018== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Prueba 5

```
$ cat prueba5.mem
R 131072
R 4096
R 8192
R 4096
R 0
R 4096
MR

$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./tp2 prueba5.mem
==19293== Memcheck, a memory error detector
==19293== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==19293== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==19293== Command: ./tp2 prueba5.mem
==19293==

Reading byte with: 0 tag, 0 set, 0 offset. Address: 131072
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 0 to cache with way: 3, set: 0
Read from cache the byte: 0

Reading byte with: 4 tag, 0 set, 0 offset. Address: 4096
Couldn't find address in cache
Oldest for set: 0 way: 2
Taking blocknum 32 to cache with way: 2, set: 0
Read from cache the byte: 0

Reading byte with: 8 tag, 0 set, 0 offset. Address: 8192
```

```
Couldn't find address in cache
Oldest for set: 0 way: 1
Taking blocknum 64 to cache with way: 1, set: 0
Read from cache the byte: 0

Reading byte with: 4 tag, 0 set, 0 offset. Address: 4096
Read from cache the byte: 0

Reading byte with: 0 tag, 0 set, 0 offset. Address: 0
Read from cache the byte: 0

Reading byte with: 4 tag, 0 set, 0 offset. Address: 4096
Read from cache the byte: 0
Missrate: 0.500000
==19293==
==19293== HEAP SUMMARY:
==19293==     in use at exit: 0 bytes in 0 blocks
==19293==   total heap usage: 556 allocs, 556 frees, 72,341 bytes allocated
==19293==
==19293== All heap blocks were freed -- no leaks are possible
==19293==
==19293== For counts of detected and suppressed errors, rerun with: -v
==19293== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Prueba 6

```
$ cat prueba6.mem
W 0, 1
W 1024, 2
W 2048, 3
W 4096, 4
W 8192, 256
R 0
R 1024
R 2048
R 4096
R 8192
MR

$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./tp2 prueba6.mem
==19380== Memcheck, a memory error detector
==19380== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==19380== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==19380== Command: ./tp2 prueba6.mem
==19380==

Writing byte 1 with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't write to cache 0 tag, 0 set, 0 offset

Writing byte 2 with: 1 tag, 0 set, 0 offset. Address: 1024
Couldn't write to cache 1 tag, 0 set, 0 offset

Writing byte 3 with: 2 tag, 0 set, 0 offset. Address: 2048
Couldn't write to cache 2 tag, 0 set, 0 offset

Writing byte 4 with: 4 tag, 0 set, 0 offset. Address: 4096
Couldn't write to cache 4 tag, 0 set, 0 offset

Writing byte 0 with: 8 tag, 0 set, 0 offset. Address: 8192
Couldn't write to cache 8 tag, 0 set, 0 offset
```

```
Reading byte with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 0 to cache with way: 3, set: 0
Read from cache the byte: 1

Reading byte with: 1 tag, 0 set, 0 offset. Address: 1024
Couldn't find address in cache
Oldest for set: 0 way: 2
Taking blocknum 8 to cache with way: 2, set: 0
Read from cache the byte: 2

Reading byte with: 2 tag, 0 set, 0 offset. Address: 2048
Couldn't find address in cache
Oldest for set: 0 way: 1
Taking blocknum 16 to cache with way: 1, set: 0
Read from cache the byte: 3

Reading byte with: 4 tag, 0 set, 0 offset. Address: 4096
Couldn't find address in cache
Oldest for set: 0 way: 0
Taking blocknum 32 to cache with way: 0, set: 0
Read from cache the byte: 4

Reading byte with: 8 tag, 0 set, 0 offset. Address: 8192
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 64 to cache with way: 3, set: 0
Read from cache the byte: 0
Missrate: 1.000000
==19380==
==19380== HEAP SUMMARY:
==19380==     in use at exit: 0 bytes in 0 blocks
==19380==   total heap usage: 556 allocs, 556 frees, 72,381 bytes allocated
==19380==
==19380== All heap blocks were freed -- no leaks are possible
==19380==
==19380== For counts of detected and suppressed errors, rerun with: -v
==19380== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

5.10. Pruebas con valgrind: Alumno

Se realizo algunas pruebas para testear que funcionaba correctamente nuestro programa

Prueba 7

```
$ cat prueba7.mem
W 0, 5
W 1, 90
R 0
R 1
MR

$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./tp2 prueba7.mem
==19483== Memcheck, a memory error detector
==19483== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==19483== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==19483== Command: ./tp2 prueba7.mem
==19483==
Writing byte 5 with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't write to cache 0 tag, 0 set, 0 offset

Writing byte 90 with: 0 tag, 0 set, 1 offset. Address: 1
Couldn't write to cache 0 tag, 0 set, 1 offset

Reading byte with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 0 to cache with way: 3, set: 0
Read from cache the byte: 5

Reading byte with: 0 tag, 0 set, 1 offset. Address: 1
Read from cache the byte: 90
Missrate: 0.750000
==19483==
==19483== HEAP SUMMARY:
==19483==     in use at exit: 0 bytes in 0 blocks
==19483==   total heap usage: 556 allocs, 556 frees, 72,323 bytes allocated
==19483==
==19483== All heap blocks were freed -- no leaks are possible
==19483==
==19483== For counts of detected and suppressed errors, rerun with: -v
==19483== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Prueba 8

```
$ cat prueba8.mem
R 0
R 1
MR

$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./tp2 prueba8.mem
==19549== Memcheck, a memory error detector
==19549== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==19549== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==19549== Command: ./tp2 prueba8.mem
==19549==

Reading byte with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 0 to cache with way: 3, set: 0
Read from cache the byte: 0

Reading byte with: 0 tag, 0 set, 1 offset. Address: 1
Read from cache the byte: 0
Missrate: 0.500000
==19549==
==19549== HEAP SUMMARY:
==19549==     in use at exit: 0 bytes in 0 blocks
==19549==   total heap usage: 556 allocs, 556 frees, 72,308 bytes allocated
==19549==
==19549== All heap blocks were freed -- no leaks are possible
==19549==
==19549== For counts of detected and suppressed errors, rerun with: -v
==19549== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Prueba 9

```
$ cat prueba9.mem
R 0
R 1024
R 2048
R 3072
R 1024
R 3072
R 4096
R 5120
R 3072
R 0
R 1024
MR

$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./tp2 prueba9.mem
==19630== Memcheck, a memory error detector
==19630== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==19630== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==19630== Command: ./tp2 prueba9.mem
==19630==

Reading byte with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 0 to cache with way: 3, set: 0
Read from cache the byte: 0

Reading byte with: 1 tag, 0 set, 0 offset. Address: 1024
Couldn't find address in cache
Oldest for set: 0 way: 2
Taking blocknum 8 to cache with way: 2, set: 0
Read from cache the byte: 0

Reading byte with: 2 tag, 0 set, 0 offset. Address: 2048
Couldn't find address in cache
Oldest for set: 0 way: 1
Taking blocknum 16 to cache with way: 1, set: 0
Read from cache the byte: 0

Reading byte with: 3 tag, 0 set, 0 offset. Address: 3072
Couldn't find address in cache
Oldest for set: 0 way: 0
Taking blocknum 24 to cache with way: 0, set: 0
Read from cache the byte: 0

Reading byte with: 1 tag, 0 set, 0 offset. Address: 1024
Read from cache the byte: 0

Reading byte with: 3 tag, 0 set, 0 offset. Address: 3072
Read from cache the byte: 0

Reading byte with: 4 tag, 0 set, 0 offset. Address: 4096
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 32 to cache with way: 3, set: 0
Read from cache the byte: 0

Reading byte with: 5 tag, 0 set, 0 offset. Address: 5120
Couldn't find address in cache
```

```
Oldest for set: 0 way: 1
Taking blocknum 40 to cache with way: 1, set: 0
Read from cache the byte: 0

Reading byte with: 3 tag, 0 set, 0 offset. Address: 3072
Read from cache the byte: 0

Reading byte with: 0 tag, 0 set, 0 offset. Address: 0
Couldn't find address in cache
Oldest for set: 0 way: 2
Taking blocknum 0 to cache with way: 2, set: 0
Read from cache the byte: 0

Reading byte with: 1 tag, 0 set, 0 offset. Address: 1024
Couldn't find address in cache
Oldest for set: 0 way: 3
Taking blocknum 8 to cache with way: 3, set: 0
Read from cache the byte: 0
Missrate: 0.727273
==19630==
==19630== HEAP SUMMARY:
==19630==     in use at exit: 0 bytes in 0 blocks
==19630==   total heap usage: 556 allocs, 556 frees, 72,371 bytes allocated
==19630==
==19630== All heap blocks were freed -- no leaks are possible
==19630==
==19630== For counts of detected and suppressed errors, rerun with: -v
==19630== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

6. Conclusión

Aprendimos como funciona una caché asociativa por conjunto de k vías, con pol'iticas de reemplazo LRU y políticas de escritura WT/¬WA. Y también la interacción con la memoria principal y como la cache va guardando los datos a medida que son requeridos, y como los va reemplanzando si es necesario, pudiendo implementarla en un lenguaje de bajo nivel, como es C. Además, dado que tuvimos problemas con valgrind pudimos aprender como es que C aloca la memoria, y evalua expresiones con punteros.

Referencias

[1] B. Kernighan, D.Ritchie, *The C Programming Languaje.* Person Prentice Hall

[2] B. Kernighan, R. Pike, *The Unix Programming Environment.* Prentice Hall

[3] D. Patterson, J. Hennessy, *Computer Organization and Design.*

[4] Valgrind http://valgrind.org/

A. Código fuente

El proyecto se encuentra en el siguiente repo de github: `https://github.com/jomedina96/orgaTP1`