4

Cache memory

Objectives

The cache memory is one of the key elements of a computer involving performance improvement. Exercises in this section review the internal operation of the cache memory, focusing on placement, replacement and write strategies, as well as corresponding issues.

Some exercises contain a figure describing the state of the cache memory in a given time instant. Thus, very small cache memories are used in these exercises in comparison to the cache size in real computers. However, the concepts reviewed in these exercises are independent of the memory capacity.

It is assumed that all the items stored in the cache memories represented in these figures are expresses in hexadecimal to simplify the notation, although the 'h' suffix is not written. The rest of the elements of the cache memory are expressed in binary, such as the valid bit or the tag bits.

In all the exercises, unless otherwise indicated, it must be assumed that a cache miss when performing a writing operation provokes reading the corresponding block from the main memory to the cache.[1]

Exercise 30.

Usually, in modern computers the cache memory is implemented inside the integrated circuit of the processor. This technique notably enhances the access time of the CPU to the cache memory, and thus, the performance of the computer is improved.

The manufacturing process of both the processor and the cache memory requires transistors integrated inside the chip. As manufacturing technology improves, more transistors can be integrated inside this chip, so caches featuring larger capacities can be designed. However, regardless of the manufacturing technology, implementing a basic cell, the element which stores 1 bit of information, requires 6 transistors approximately.

In this exercise the Intel© Core i7-3770T processor, with a manufacturing technology of $22\,\text{nm}$ and Ivy Bridge microarchitecture, will be used. This processor costs about 300 Euro and integrates about 1400 millions of transistors in four cores, integrated peripherals and three cache levels:

- 4×64 KiB L1-cache.

[1] In this scenario some real CPUs write directly in the main memory.

- 4×256 KiB L2-cache.
- 8 MiB L3-cache.

❑ **30.1** What is the percentage of the transistors in this CPU used to implement the cache memory cells?

$$\frac{\left(4\times64\times2^{10}+4\times256\times2^{10}+8\times2^{20}\right)\times8\times6}{1400\times10^{6}} \times 100 \equiv 33.25\%$$

❑ **30.2** Assuming that the cost of the cache memory is proportional to the number of transistors, what is, approximately, the cost of the cache of this processor?

$300 \times 0.3325 = 99.75$ `Euro`

Exercise 31.

One computer features an address space of 4 MiB and a cache memory of 64 KiB. The memory implements the byte addressing technique, that is, each byte of the memory can be accessed individually. Furthermore, the cache and the main memory block consists of 16 bytes.

❑ **31.1** How many cache blocks does this computer contain?

`64 Ki/16 = 4 Ki blocks`

❑ **31.2** How many blocks does the main memory contain?

`4 Mi/16 = 256 Ki blocks`

❑ **31.3** What is the size, expressed in bits, of the memory addresses used by this computer?

`22 bits`

Each memory address is divided into several fields depending on the placement strategy implemented by the cache memory. In the following cases, write down the name of the fields in which each memory address is divided, as well as the number of bits used per field.

❑ **31.4** Placement strategy: fully associative.

```
18 most significant bits: tag
4 least significant bits: offset
```

❑ **31.5** Placement strategy: direct-mapped.

```
6 most significant bits: tag
12 intermediate bits: block number
4 least significant bits: offset
```

❑ **31.6** Placement strategy: set associative, 4 ways.

```
8 most significant bits: tag
10 intermediate bits: set number
4 least significant bits: offset
```

Exercise 32.

A cache memory will be designed for the CPU of the CT. The size of the address space of this computer is 64 Ki words, 16-bit wide each. The cache memory is designed with a set associative placement strategy. The cache block consists of 8 words, and the tag field in each memory address will be 6-bit wide. Furthermore, it is also known that if the cache memory of this computer used a direct-mapped placement strategy, the tag field would be 4-bit wide.

❑ **32.1** How many sets does the cache memory contain?

$2^{16-6-3} = 128$

❑ **32.2** How many blocks does the cache memory contain?

$2^{16-4-3} = 512$

❑ **32.3** What is the size, expressed in bytes, of the cache memory?

$512 \times 8 \times 2 = 8\,\texttt{KiB}$

Exercise 33.

The following features of a computer are known:

- Address space of 1 MiB implementing byte addressing.
- Unified cache memory implementing direct-mapped placement with 4 KiB capacity and a block size of 32 bytes.
- Memory access time:
  - Cache memory: 4 ns
  - Main memory: 20 ns
  - In a cache miss, the only access time considered is the main memory time.

❑ **33.1** What is the width of each memory address generated by the CPU of this computer?

20 bits

❑ **33.2** What fields, and how many bits each, are each memory address divided in?

```
8 most significant bits: tag
7 intermediate bits: block number
5 least significant bits: offset
```

❑ **33.3** Assuming the fact that the cache memory is empty, the CPU accesses consecutively to the following memory addresses: A5234h, A8230h, A823Fh, and FF01Ah. What memory addresses provoke cache misses? And why?

```
A5234h, the corresponding cache block is empty
A8230h, the cached block does not match, and must be replaced
FF01Ah, the corresponding cache block is empty
```

❑ **33.4** How long does the CPU take to perform the memory accesses listed in the previous question?

$1 \text{ hit} \times 4\,\text{ns} + 3 \text{ miss} \times 32 \times 20\,\text{ns} = 1924\,\text{ns}$

❑ **33.5** Assuming that the cache memory is empty again, what is the hit rate obtained when executing the following snippet?

```
1 for (i = 0; i < 1024; i++)
2 {
3     v[i] = 0;
4 }
```

It is also known that v is an integer array, in which each integer number is stored in four consecutive memory locations, stored in memory from the memory address 2A530h. It is also known that the constants and the loop index are stored in registers of the CPU, that is, no memory accesses are required.

1024 items × 4 bytes / 32 bytes per block = 128 blocks
=> 128 misses. $\frac{4096-128}{4096} \times 100 = 96.9$ %

❑ **33.6** Assuming that the cache memory is empty again, what is the hit rate obtained when executing the following snippet?

```
1 for (i = 0; i < 100; i++)
2 {
3     a[i] = 0;
4     b[i] = i;
5 }
```

It is also known that both a and b are integer arrays. Elements of a are consecutively stored in memory from the memory address 10220h; elements of b are consecutively stored in memory from 00230h. It is also known that the constants and the loop index are stored in registers of the CPU, that is, no memory accesses are required.

0%, every memory access provoke a cache miss, and thus, the blocks are always being replaced

#	Addr	R/W	#	Addr	R/W
1	15C0h	R	**9**	208Fh	W
2	15C1h	R	**10**	2090h	R
3	15C2h	R	**11**	15D9h	R
4	15C9h	R	**12**	405Ch	W
5	2080h	W	**13**	15D6h	R
6	2081h	W	**14**	15D7h	R
7	15CAh	R	**15**	6579h	R
8	15D8h	R	**16**	657Ah	R

Table 4.1: Memory access pattern of the exercise #34.

❑ **33.7** How can the above hit rate be improved?

```
Using a set associative placement strategy with two or more
sets, or a fully associative one.
```

Exercise 34.

The following features of a computer are known:

- 16-bit wide address bus.
- 16-bit wide memory word.
- Cache memory with 64 bytes, organized in blocks of 8 words and fully associative placement strategy.
- LRU replacement strategy.
- *Write-back* writing strategy.

In this computer a program with the memory access pattern shown in table 4.1 will be executed. In this table, `Addr` expresses the accessed memory address and `R/W` expresses whether it is a reading (`R`) or a writing (`W`) access.

It must be assumed that the cache memory is empty when the execution of the program starts.

❑ **34.1** How many blocks does the cache memory contain?

```
4
```

❑ **34.2** What memory addresses provoke a cache miss when executing the program in the above memory configuration?

```
15C0h, 15C9h, 2080h, 15D8h, 208Fh, 2090h, 405Ch, 15D6h, and
6579h.
```

❑ **34.3** What is the hit rate of the program?

43.75%, 7 hits out 16 memory accesses.

❑ **34.4** What main memory block, or blocks, are replaced when executing the program? Write down its/their number(s) in hexadecimal. You may answer NONE if no blocks are replaced.

2B8h, 410h, 2B9h, 411h, and 412h.

❑ **34.5** What main memory block, or blocks, are updated when executing the program? Write down its/their number(s) in hexadecimal. You may answer NONE if no blocks are updated.

410h and 411h.

Exercise 35.

One program executes a loop which accesses 10 times to the following memory addresses: A400h, A401h, A402h, 8201h, 8202h, D250h, D251h, and D252h.

The processor in which this program is run has a cache memory with eight words per block, and implements the LRU replacement strategy. Two placement strategies are proposed:

A) Direct-mapped cache.

B) 2-way set associative cache.

❑ **35.1** What is the hit rate using the direct-mapped cache?

$(80 - (3 + 2 \times 9))/80 \Rightarrow 73.75\%$

❑ **35.2** What is the hit rate using the 2-way set associative cache

$(80 - 3)/80 \Rightarrow 96.25\%$

Exercise 36.

The initial state of a cache memory using direct-mapped placement strategy is shown in figure 4.1.

❑ **36.1** Which writing strategy is being used?

Write-back, since the cache has a dirty bit.

	Valid bit	Dirty bit	Tag	Offset 7	6	5	4	3	2	1	0
0	1	0	011011	A212	B6A1	4519	180A	671B	12C5	2F34	CCA4
1	1	1	100111	1223	9034	BD34	F545	56A6	1C56	09A6	B711
2	1	0	100101	B92A	3508	1212	48A3	341B	C540	6702	4878
3	0	0	110101	39F5	E42D	D5F6	A318	3567	2DF5	D3A1	68DD
4	1	0	001011	13A2	9723	22A2	BDA4	E645	2F12	B034	0816
5	1	1	010101	3278	6790	01A1	0000	56B2	0000	0089	0000
6	0	0	111110	56A4	1171	0317	2000	09D1	ABA2	F23D	EC04
7	0	0	110110	72C0	02A0	09F1	4A74	3E22	BA80	0000	BA41
8	1	0	111000	A242	B619	4517	1802	6761	C500	2F78	C00C
9	1	0	101101	1002	9056	B23D	4C05	A6D1	1C23	0679	1100
A	1	0	000101	A42A	3615	12BA	0248	C01B	409F	A402	0048
B	0	0	110100	D139	232D	F6D1	1856	A435	2C0D	D9F3	6842
C	1	1	010011	6113	9027	1722	B42D	02E6	172F	B000	C008
D	1	1	101101	3223	2367	0001	00C0	0056	0000	0000	9F00
E	0	0	110001	5026	11BA	A403	20D1	09BA	AB56	F61D	E79C
F	1	1	101110	7422	0056	4209	4A42	BA32	89F0	0000	7941

Figure 4.1: Initial state of the cache memory in exercise #36.

❑ **36.2** What is the size of the cache memory, expressed in bytes?

$2^4 \times 2^3 \times 2 = 256$ bytes

❑ **36.3** How many main memory blocks are pending to be updated?

5

❑ **36.4** What is the main memory block with lowest memory address which is cached? Answer in hexadecimal; you may answer UNDEFINED if it cannot be identified.

000101 1010 = 05Ah

❑ **36.5** How many different main memory blocks can be placed, not simultaneously, in the cache block $\#2$?

2^6 = 64

❑ **36.6** Could an LRU replacement strategy be used in this cache memory? Justify your answer.

Using direct-mapped placement strategy, there is no need to use a replacement strategy since each main memory block is assigned to the same cache block.

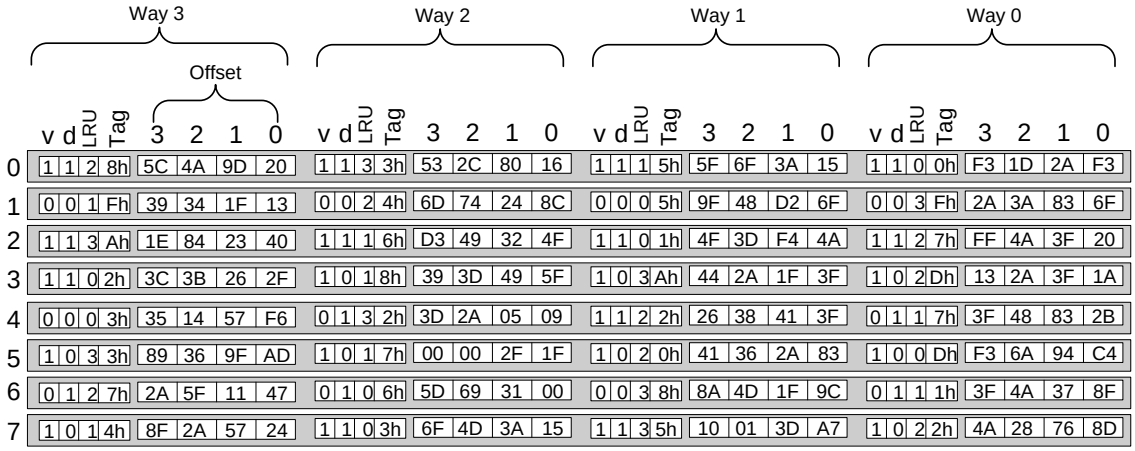

Figure 4.2: Initial state of the cache memory in exercise #37.

❑ **36.7** When the cache memory reads the `012Ah` memory address, what happens, a hit or a miss? What cache block is being accessed? Will any main memory block be replaced? If so, which one? Answer in hexadecimal.

```
Miss
Block #5
The #155 main memory block is replaced
```

❑ **36.8** If a DMA-capable peripheral would access the main memory to read the `177Fh` memory address, what value would it read? What would happen in the cache memory?

```
7422h
It would be necessary to update the main memory with the cache
block #F, and the dirty bit of this block would be reseted.
```

❑ **36.9** If a DMA-capable peripheral would access the main memory to read the `018Dh` memory address, what value would it read? What would happen in the cache memory?

```
Nothing would happen since this memory address is not cached.
```

Exercise 37.

Figure 4.2 shows the initial state of a unified cache memory. Each cache block has a valid bit, v, a dirty bit, d, two LRU bits, and a 4-bit wide tag. The block with a lowest LRU value is the least recently accessed.

❑ **37.1** How many main memory blocks are stored in the cache memory?

```
21
```

❑ **37.2** What value does the cache memory provide when the CPU tries to read the `053h` memory address? Answer MISS if a cache miss is generated?

```
26h
```

❑ **37.3** How many blocks are coherent between the cache and the main memory?

```
12 blocks
```

❑ **37.4** What is the highest main memory address that could be cached in the way $\#1$ of the set $\#6$?

```
1FBh
```

❑ **37.5** Assuming a *write allocate* strategy, if the CPU writes in the `044h` memory address, what is the state of the valid, dirty and LRU bits of the corresponding cache line?

```
v = 1; d =1; LRU = 3
```

❑ **37.6** Write down a memory address that could be generated by the CPU for the main memory block `1F` to be updated in the main memory?

```
Anyone cached in the set #7; address pattern: (x xxx1 11xx)
```

❑ **37.7** The interface of a peripheral with DMA ability accesses three different memory locations. Write down the bits of the cache line that result modified as well as the final values for each of the accesses. The memory addresses accessed are:

- Write `101h`.
- Read `1A2h`
- Read `0BCh`

```
Write 101h: Cached and dirty bit set. Thus, the block must be
updated → Its valid bit is reset after writing the main memory.

Read 1A2h: Not cached; nothing happens in the cache memory.

Read 0BCh: Cached and dirty bit set. Thus, the block must be
updated → Its dirty bit is reset after writing the main memory.
```

Exercise 38.

Figure 4.3 shows the state of a separated cache memory when running a program. Each line in the data cache has a valid bit, v, a dirty bit, d, and a bit indicating if the block has been accessed, a. The a bit is set when block of the set is accessed and it is reset when the other block of the set is accessed. The replacement strategy is LRU based on the a bit. The code cache does not provide a dirty bit.

Code cache

Set	Way 0 v	a	Tag	7	6	5	4	3	2	1	0	Way 1 v	a	Tag	7	6	5	4	3	2	1	0
0	1	1	011101	28	3D	1A	68	00	12	34	F0	0	0	100011	4D	12	4D	2A	F6	E5	45	6A
1	0	1	010000	B6	57	3F	2D	90	24	90	5F	1	1	110110	C3	02	4F	3A	76	25	2B	3C
2	1	0	101101	3A	3B	3C	3D	12	4D	4A	9C	1	1	100001	61	58	F5	55	3A	22	57	DA
3	1	0	011010	9B	0A	68	35	25	21	6F	2F	1	1	101001	23	00	38	35	9D	A8	A3	67

Data cache

Set	Way 0 v	d	a	Tag	7	6	5	4	3	2	1	0	Way 1 v	d	a	Tag	7	6	5	4	3	2	1	0
0	0	0	0	00001	04	00	80	00	F0	0A	C4	16	1	0	1	10010	A3	67	00	2A	7F	FF	4F	2B
1	1	1	1	00010	00	12	31	36	D6	FF	2A	A6	1	1	0	00101	15	3F	27	62	61	29	3F	E5
2	0	0	0	01110	58	21	49	26	36	90	21	27	1	1	1	00010	C5	54	CC	2A	25	68	90	23
3	1	1	1	10101	AA	A1	B5	74	C3	3A	34	00	1	0	0	10110	46	79	67	A3	00	22	22	57
4	1	0	0	01101	12	3A	90	00	D1	CC	10	05	1	1	1	00110	07	56	00	4C	00	45	11	43
5	0	1	1	11001	21	98	CE	19	B1	00	11	02	1	0	1	10110	C5	3F	23	46	00	31	59	AA
6	1	1	0	11110	20	13	53	0A	FF	19	03	79	1	1	1	00110	08	76	91	E3	A1	87	E1	32
7	1	1	1	00101	EE	09	16	A8	01	54	27	00	0	0	0	10110	00	17	74	B3	92	0C	17	C5

Figure 4.3: Initial state of the cache memory in exercise #38.

❑ **38.1** Which is the writing strategy in this cache memory?

Write-back int data cache; *write-through* in code cache.

❑ **38.2** What is the maximum number of main memory blocks that can be simultaneously cached?

24

❑ **38.3** Taking the information of figure 4.3 into account, what is the maximum number of cache misses that can take place without replacement?

6

❑ **38.4** Taking the information of figure 4.3 into account, how many main memory blocks must be updated in case of being replaced?

8

❑ **38.5** What is the lowest code address that is cached? Answer in hexadecimal.

358h

❑ **38.6** What main memory block, or blocks, are replaced when the CPU reads a double precision floating-point number (64-bit wide) stored in the `070h` memory address?

`F6h`

❑ **38.7** When the CPU writes a data item in the `15Ah` memory address a hit occurs. If a *write allocate* strategy is used, the main memory block must be copied to the cache memory before the writing operation is performed. Write down the cache block in which the main memory block containing the above address will be stored, as well as the new values for the tag, valid, dirty, and access bits of the corresponding cache line.

Data cache, set #3, way #1.
Tag = 00101; v = 1; d = 1; a = 1.

Exercise 39.

Which of the following statements are TRUE? You may answer NONE if you think all of them are false.

A) The associativity level of a set associative cache is increased when the number of sets is decreased and the number of cache blocks remains constant.

B) L1 separated caches improve the performance with regard to unified caches when they are connected to a CPU which does not implement any instruction-level parallelism technique.

C) When a peripheral device with DMA ability reads a main memory location which is cached and the main memory is connected to a cache memory implementing the *write-back* strategy, coherence problems may appear.

D) In real computers, the most commonly used placement strategy is fully associative.

E) Usually, a separated cache provides a hit rate lower that its unified equivalent.

`A, C, and E`

Exercise 40.

Which of the following statements are TRUE? You may answer NONE if you think all of them are false.

A) The memory hierarchy provides the minimum access time when the hit rate in the cache memory is 0%.

B) A cache block is replaced in a reading operation if this operation provokes a cache miss.

C) Usually, the *write-back* strategy provides a higher performance in memory accesses with regard to the *write-through* strategy.

D) When the control is switched from a user task to the operating system, or vice versa, the cache miss rate is usually increased.

E) In multicore processors, all cache levels are usually replicated in all the cores.

`B, C, and D`