COMPUTER ARCHITECTURE
2020 - 2021

TD n°4 - Correction

Exercise 1:

We consider a 2 MB main memory where each byte is separately addressable (→ address = "number of bytes from 0"):

1. Calculate the address, in octal, of the sixth element of an array whose first element's address is 77_8, and whose elements are all 16 bits long.
   Each element is stored in 2 bytes, so the 6th element is at the starting address + 10 bytes, which gives $77_8 + 12_8 = 111_8$ (or $77_8 = 63_{10} + 10_{10} = 73_{10} = 111_8$).

2. Calculate, in decimal, the number of bytes before the address 77_8.
   $77_8 = 63_{10}$, so before this address we have the addresses 0 to 62 available, which gives 63 bytes.

3. Calculate the size of this memory by expressing it in 16-bit words, then in 32-bit words.
   Memory size is 2 MB or 1 M 16-bit words or 512 32-bit words.

Exercise 2:

If the address register of a memory has 32 bits, calculate:

1. The number of addressable words if 1 word = 1 byte
   2^{32} 1-byte words = 4 G.

2. The highest possible address for these 1-byte words
   Highest address = $2^{32} - 1$ = 4 294 967 296 – 1 = 4 294 967 295.

3. The number of addressable words if 1 word = 32 bits
   2^{32} 32-bit words = 4 G.

4. The highest possible address for these 32-bit words
   The highest address is always = $2^{32} - 1 = 4\ 294\ 967\ 295$.

The number of words is the same regardless of the word size, which is normal because it only depends on the number of bits in the address. On the other hand, the size of the memory is different in the two cases, it is 2^{32} x 8 bits in the first case, and 2^{32} x 32 bits in the second one.

Exercise 3:

A RAM memory, connected to the processor, has a cycle time of 20 ns for the first access and 10 ns for the next three accesses which are accelerated. Each access extracts 8 bytes.

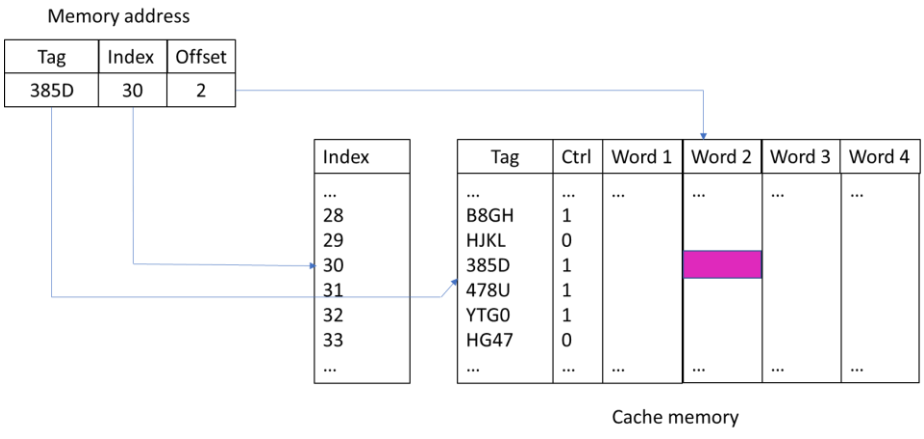
What is the bandwidth of the information transfer between the memory and the processor?

8 bytes are retrieved in 20 ns and then another 3 × 8 Bytes in the next 30 ns. So 32 bytes are transferred every 50 ns.

The bandwidth is thus: $32 / (50 * 10^{-9}) =$ **640 MB/s**

Exercise 4:

As a reminder, a word in a cache memory is accessed via an address that indicates its belonging block (label), its place in the block (offset) and the place of this block in the cache (index) as shown in the following figure.



A processor has a direct-match cache memory with four entries and the following features:

- Memory addresses are 16 bits long.
- Each memory word is 32 bits long.
- The main memory is byte addressable.
- Each cache entry contains a block of four words.

1. What is the capacity of the main memory in KB and K words?
   The capacity of the main memory is equal to 2^{16} bytes, i.e., 64 KB, or 16 K words.

2. What is the length of the label?
   A block consists of 4 words = 16 bytes, which assumes that 4 bits are needed to identify the offset in an address.
   There are 4 entries in the cache, which assumes that 2 bits will be used to identify the index in an address.

   Tag size = Address size - Offset size - Index size
   = 16 – 4 – 2 = **10 bits**

3. What is the real size of the cache?
   The real size of the cache is equal to 4 × (1 + 10 + 128) = **556 bits**.

Exercise 5:

A cache has lines of 8 bytes. The access time to the main memory during a line transfer is 50 ns for the first byte and 5 ns per subsequent byte for the rest of the line.

1. What is the average time to retrieve a line if the cache is **write-through**? If 30% of the cache lines are modified, what is the average retrieval time of a line if the cache is **write-back**?
   It takes 50 ns to retrieve a line for the first byte and then 5 ns for each of the remaining seven bytes, for a total of **85 ns**. This is the time for a **write-through** cache.

   However, in the case of a **write-back** cache, the time taken to write back a modified line must be included, i.e., 85 ns again, in 30% of cases. The total time is then 85 + 0.3 × 85 or **110.5 ns**.

2. If, on average, each line is modified three times, how much time does each cache spend writing the data back to main memory when the line is replaced? At how many rewrites per line is the **write-back** cache more interesting?
   For a **write-through** cache, each rewrite only concerns one byte and therefore uses 50 ns. There are three rewrites, so an average time of **150 ns** per modified line.

   A **write-back** cache only rewrites the data once, when the line is replaced, but must rewrite the entire line; this takes **85 ns**.

   This 85 ns time is the same regardless of the number of changes to the line in a **write-back** cache. In contrast, a **write-through** cache uses 50 ns per modification to change the modified byte in main memory.

   If there are, on average, more than (85/50) = 1.7 changes per modified line, the **write-back** cache is more advantageous.