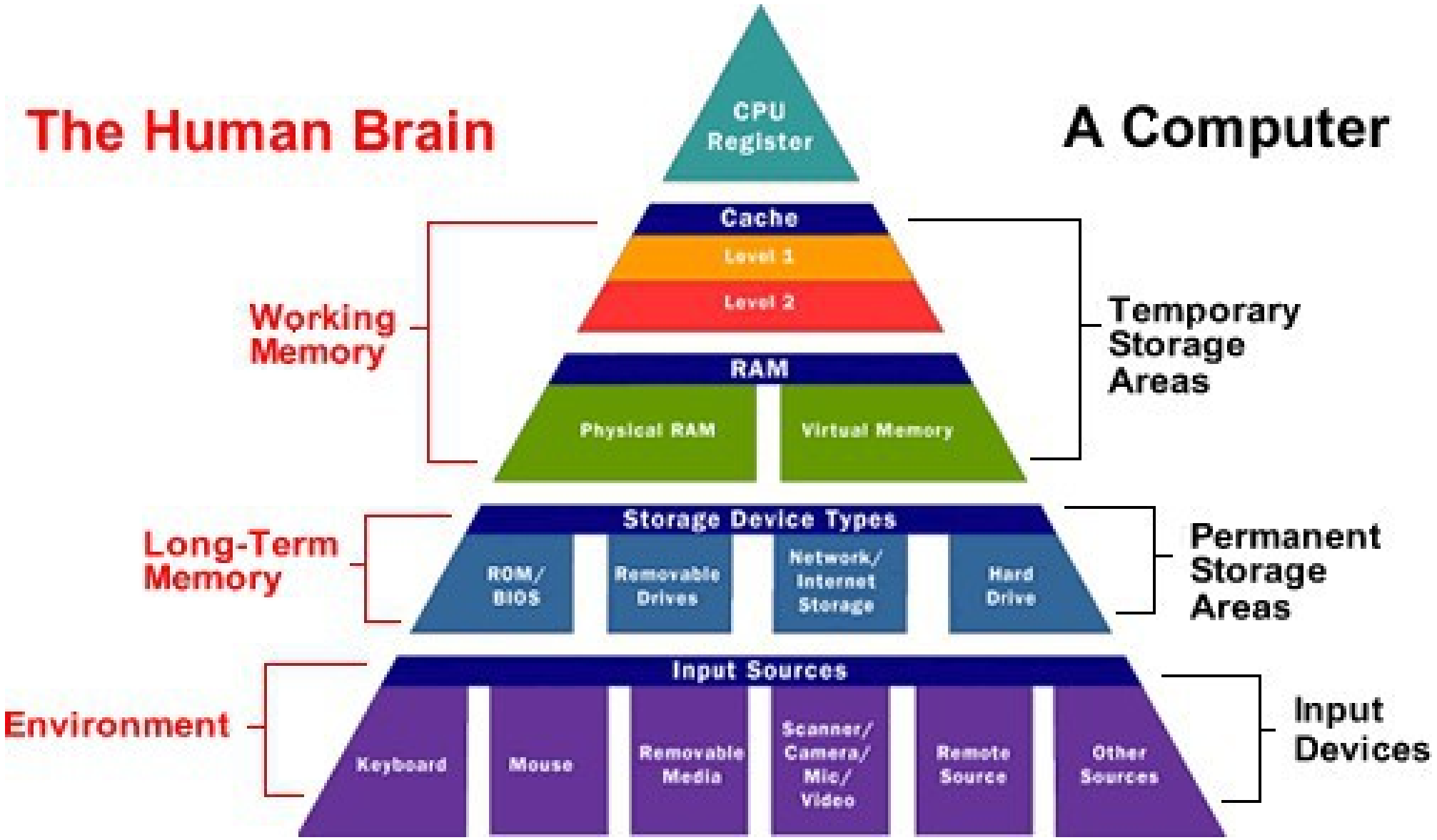
# Cache Memory

Baiqiang Wen
Norman Tang
Chenwei Cao


The Human Brain
A Computer
CPU Register
Cache
Level 1
Level 2
RAM
Physical RAM
Virtual Memory
Working Memory
Temporary Storage Areas
Storage Device Types
ROM/ BIOS
Removable Drives
Network/ Internet Storage
Hard Drive
Long-Term Memory
Permanent Storage Areas
Input Sources
Keyboard
Mouse
Removable Media
Scanner/ Camera/ Mic/ Video
Remote Source
Other Sources
Environment
Input Devices

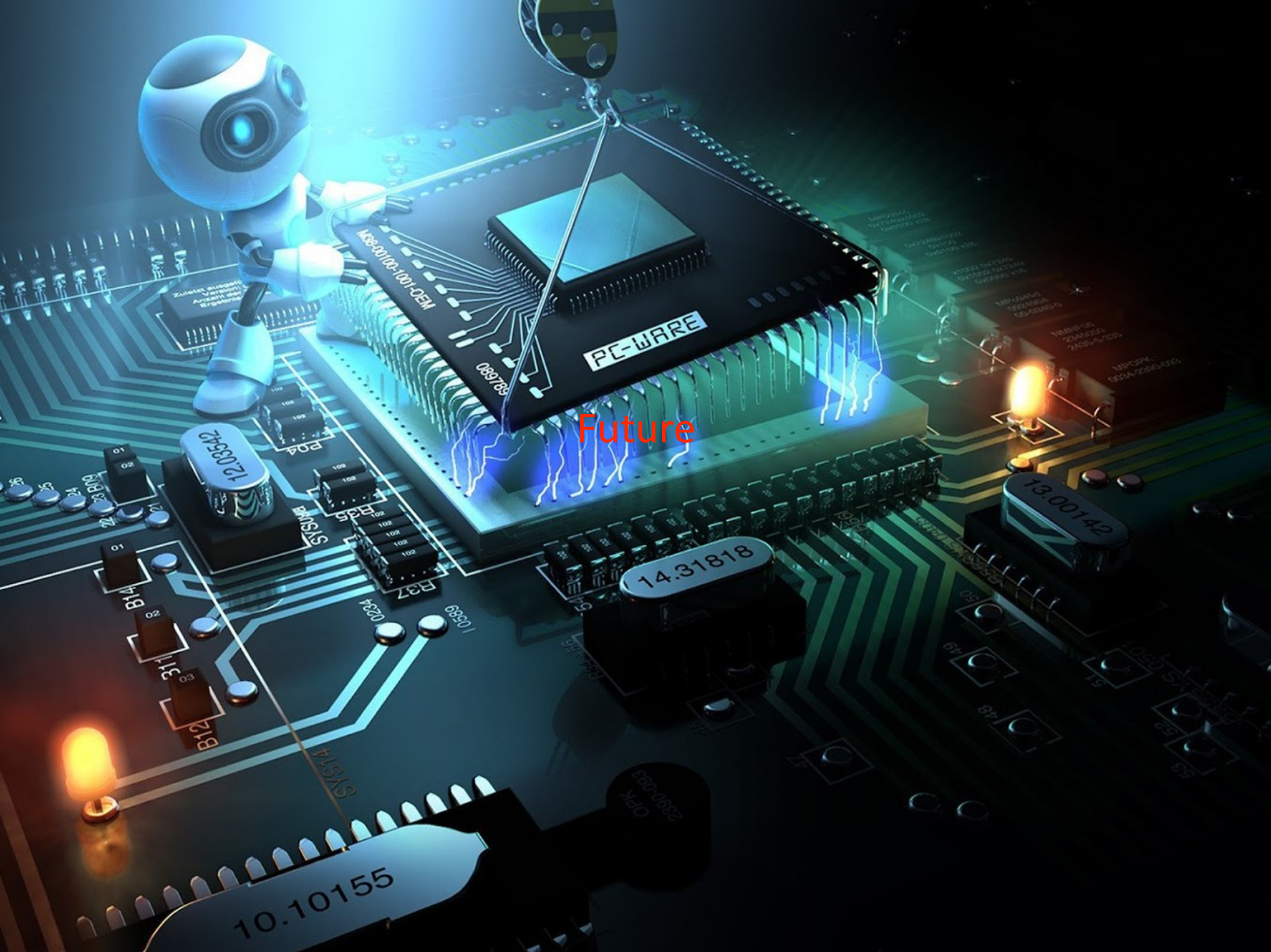
- Cache memory is a small-sized type of volatile computer memory
- Cache is cheaper
- Cache memory is a very high speed memory
- Cache is used to reduce the average time to access data from the main memory


HISTORY

1968 C.J. Conti introduced cache when he describe the difference between the system 360/85 and 360/91

The 68010, released in 1982, has a "loop mode" which can be considered a tiny and special-case instruction cache that accelerates loops that consist of only two instructions. The 68020, released in 1984, replaced that with a typical instruction cache of 256 bytes, being the first 68k series processor to feature true on-chip cache memory.

- The 68030, released in 1987, is basically a 68020 core with an additional 256-byte data cache, a process shrink, and added burst mode for the caches.
- The 68040, released in 1990, has split instruction and data caches of four kilobytes each.
- The 68060, released in 1994, has the following: 8 KB data cache (four-way associative), 8 KB instruction cache (four-way associative), 96-byte FIFO instruction buffer, 256-entry branch cache, and 64-entry address translation cache MMU buffer (four-way associative).


PC-WARE
Future
14.31818
13.00142
12.00542
10.10155

# Cache Performance

hit or miss:

- find memory location in cache
- does not find memory location in cache

hit ratio = hit/(hit + miss)

number of hits/ total accesses

- Early cache designs focused entirely on the  direct cost of cache and RAM and average execution speed.
- More recent cache designs also consider energy efficiency, fault tolerance, and other goals.
- Researchers have also explored use of emerging memory technologies such as eDRAM (embedded DRAM) and NVRAM (non-volatile RAM) for designing caches.

# Multi-ported cache

- A multi-ported cache is a cache which can serve more than one request at a time.
- When accessing a traditional cache we normally use a single memory address, whereas in a multi-ported cache we may request N addresses at a time
- The benefit of this is that a pipelined processor may access memory from different phases in its pipeline. Another benefit is that it allows the concept of super-scalar processors through different cache levels.

# Cache Performance

hit or miss:

- find memory location in cache
- does not find memory location in cache

hit ratio = hit/(hit + miss)

number of hits/ total accesses

# Cache Performance

- Miss rate: fraction of memory references not found in cache
- Miss rate = 1 - hit rate

- Typical miss rates:
  - 3-10% for L1
  - quite small(<1%) for L2

# Cache Performance

- Hit time: time to deliver a line in the cache to the processor includes time to determine whether the line is in the cache.

- Typical times:
  - 1-2 clock cycle for L1
  - 5-20 clock cycle for L2

# Calculation avg Access Time

example: 2 level cache, DRAM

L1 costs 1 cycle to access and has a miss rate of 10%

L2 cache costs 10 cycles to access has a miss rate of 2%

DRAM costs 80 cycles to access(no miss rate)

Then avg memory access time(AMAT) would be:

| | |
|---|---|
| 1+ | always access L1 cache |
| 0.1*10+ | probability miss in L1 cache * time to access L2 |
| 0.1*0.02*80 | probability miss in L1 cache * probability miss in L2 |

cache * time to access DRAM = 2.16 clock cycles

# Cache Mapping

3 models are commonly used

- Direct Mapping
- Associative Mapping
- Set-Associative Mapping

# Direct Mapping

- The simplest kind of mapping.
- Each block in memory has one and only one place where its cache line will be held. If a part of the memory block is occupied when a new one is loaded, we replace the old one.
- The address space consists of two parts, and the cache stores only the tag.
  - Index
  - Tag
- The mapping is done using the equation x = y % n
  - X = block number in cache,
  - y = block number of memory,
  - n = number of blocks in cache
- Direct mapping's performance is good in the best case, but quickly suffers. It is directly proportional to its hit ratio.

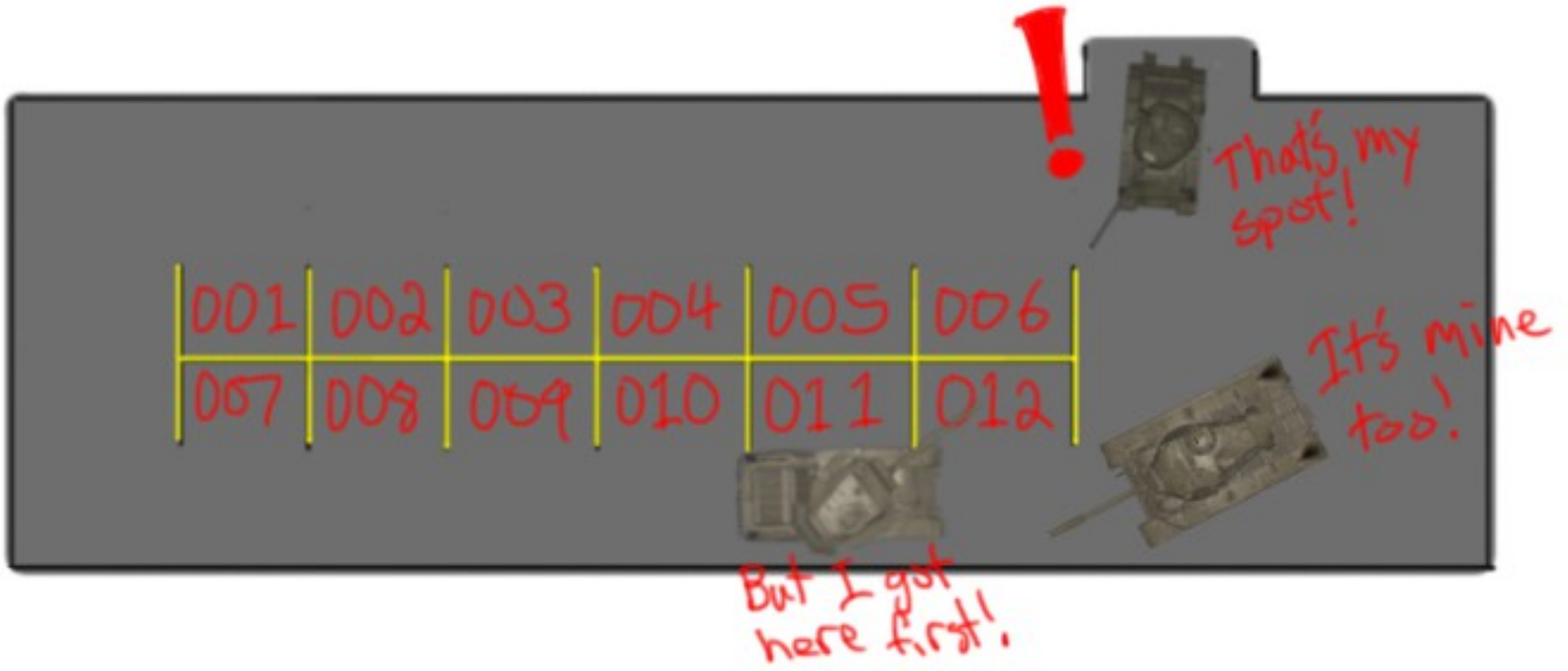
# Fully Associative Mapping

- In some ways the opposite thought of direct mapping. Any memory block can go into any cache line. This means that we need another way of identification.
- This is achieved by using word ID bits to find which in the block is needed. The tag then becomes all of the remaining bits in the line.
- Highly flexible, and generally considered to have the best performance in terms of hit/miss ratio.
    - There is a price to pay for this though:  To check if an address exist in the cache, we must compare all existing entries tags. We also must implement a replacement algorithm, such as Least Recently Used (LRU).
- Ideally used for very small caches, because as cache size rises, checking through the cache takes too much time.

# Set Associative Mapping

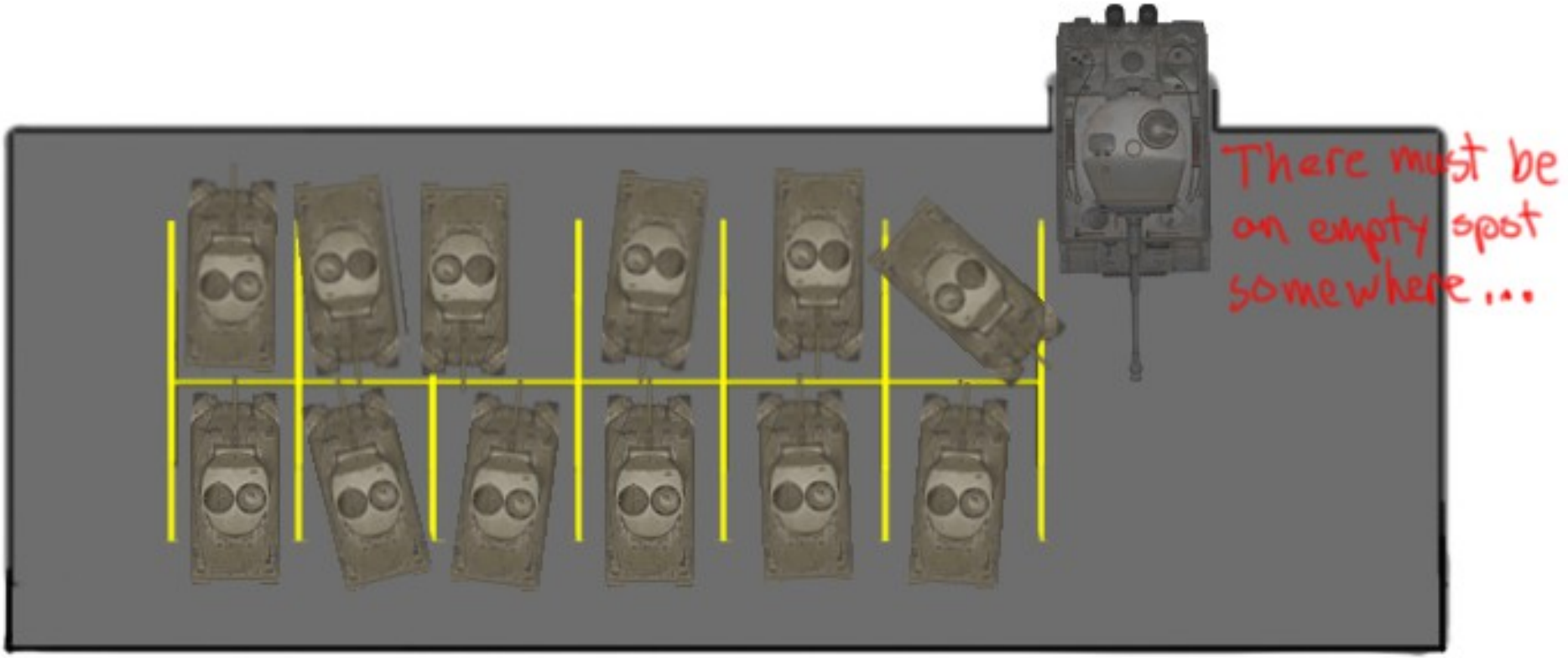
- A mixture between direct and fully associative mapping. The idea is to combine the best of both worlds, the hit ratio of fully associative, and the simple calculations of direct mapping.
- Instead of only allowing only one line that a memory block can map to, it will group a few lines together to form a set. That block can then map to any line in the set.
    - Allows each word in the cache to have two or more words in the memory for the same index address.
- To find an address, first look at the indexes of the set and then search the tag fields of all lines in the set for the address.
- The number of sets can vary, usually from 2, 4, 8, and 16.
    - Past sets of four lines, we start seeing reduced gains.

# Direct Mapping Analogy



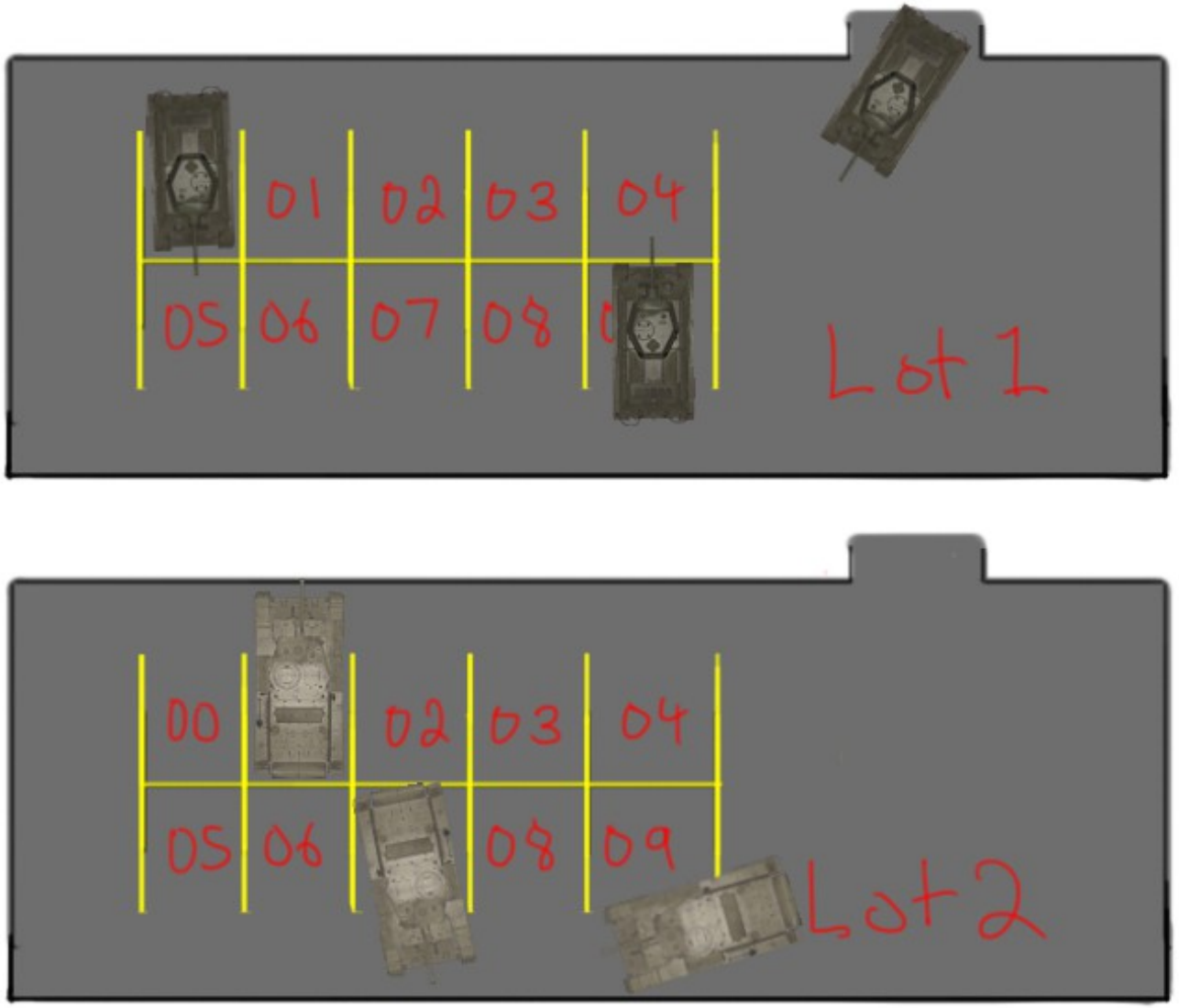

- Given a student ID, you may only park in the space that matches the first 3 numbers of your ID number.
- A potential problem arises when we have lots of students with the same first 3 numbers in their ID. There will be lots of competition for the same space.
- The upside is that you will know right away if your spot is available, and you don't spend much time searching.

# Full Associative Mapping Analogy




- It doesn't matter what your ID is, you can park anywhere you want to!
- But, what if all the spaces are filled?
  - You will end up driving to every single one looking for an empty spot before going through the process of kicking one out.

# Set Associative Mapping Analogy




- Your parking lot is based on the first digit of your student ID number, and you can park anywhere in that lot, but only in that lot.
- This gives you some flexibility about where to park. You don't have to search the entire campus to find a spot, and there's potentially less competition for a spot.