CHAPTER 7-1

Memory

Collage of arts, media and Technology

Recall the basic

- Memory stores instruction and data.
- Definitions:
  - **1 byte** = 8 bits
  - **1 word**: a unit of transfer between main memory and registers, usually size of register, *32 bits or 4 bytes*.
  - 1 KB (kilo-bytes) = 2^{10} bytes; 1 MB (mega-bytes) = 2^{20} bytes; 1 GB (giga-bytes) = 2^{30} bytes; 1TB (tera-bytes) = 2^{40} bytes.

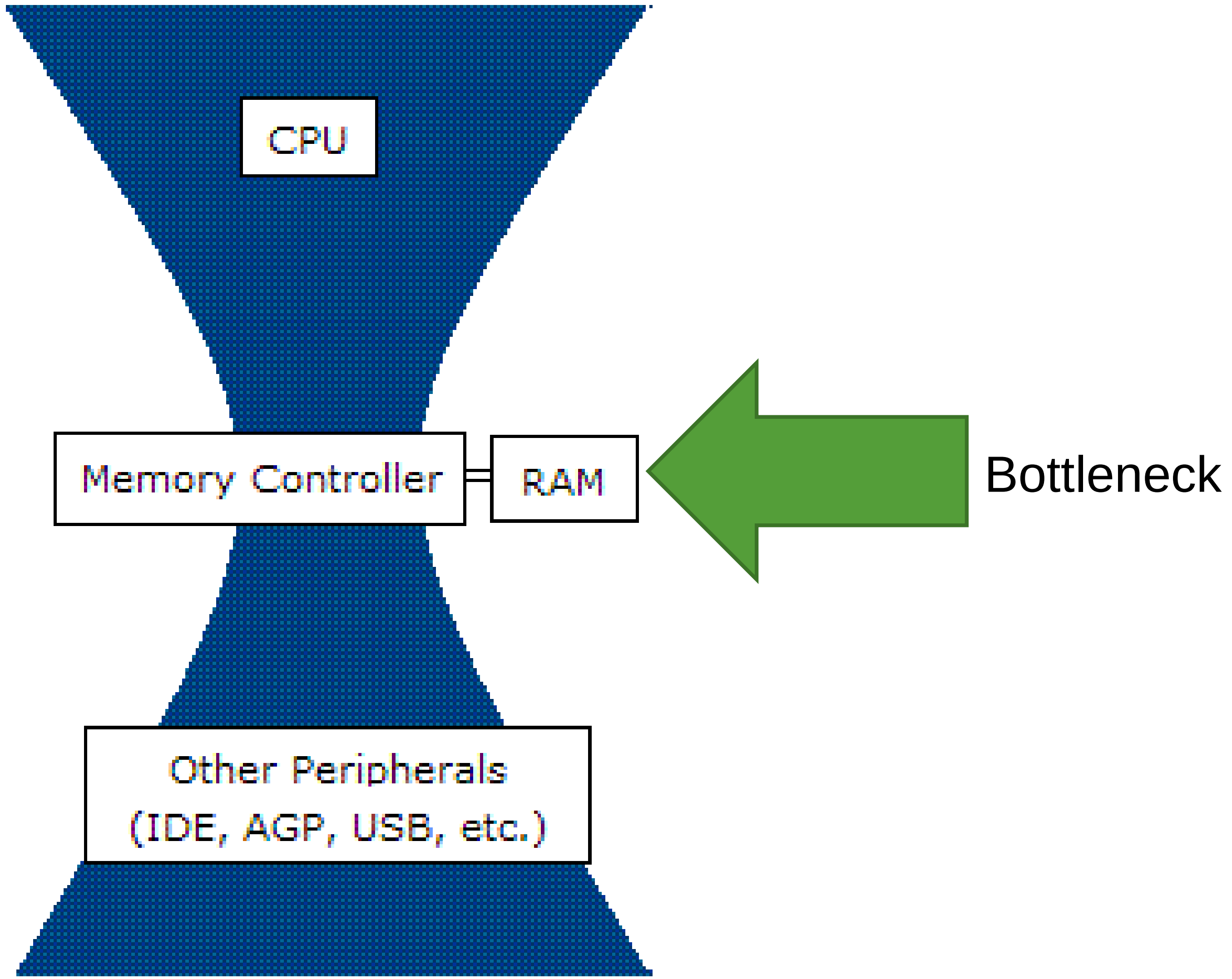
0	1	0	1	1	1	1	0

Requirement of Memory

- Programmers wish that the memory would be :
  - Fast access
  - Large capacity (the more memory, the better)
  - Economical cost
  - Non-volatile
- However, the wish cannot be fulfilled using 1 kind of memory.
  - Faster memory comes with a higher price.
  - Speed is slowed down when the capacity is increased.

Memory Price Example

Memory Technology	Normal Access Time	Price (Bath / G)
DDR3	50 – 70 ns	222.5
Magnetic Disk	5 – 20 ms	2.29


CPU
Memory Controller
RAM
Other Peripherals
(IDE, AGP, USB, etc.)
Bottleneck

- Not all accumulated information is needed by the CPU at the same time.
- There some unneeded information.
- Meaning
  - You do **NOT NEED EVERY** information all the time.


Principle of Locality

- The dream of the programmer can come true by "**exploiting**" the Principle of Locality.
- Memory references within a certain process tends to be closed to each other.
- There are 2 different type of locality.
  - Temporal locality
    - If an information is used, it is likely to be used again soon.
  - Spatial locality
    - If an information is used, the information whose location is nearby is likely to be used soon.

Example of Principle of Locality

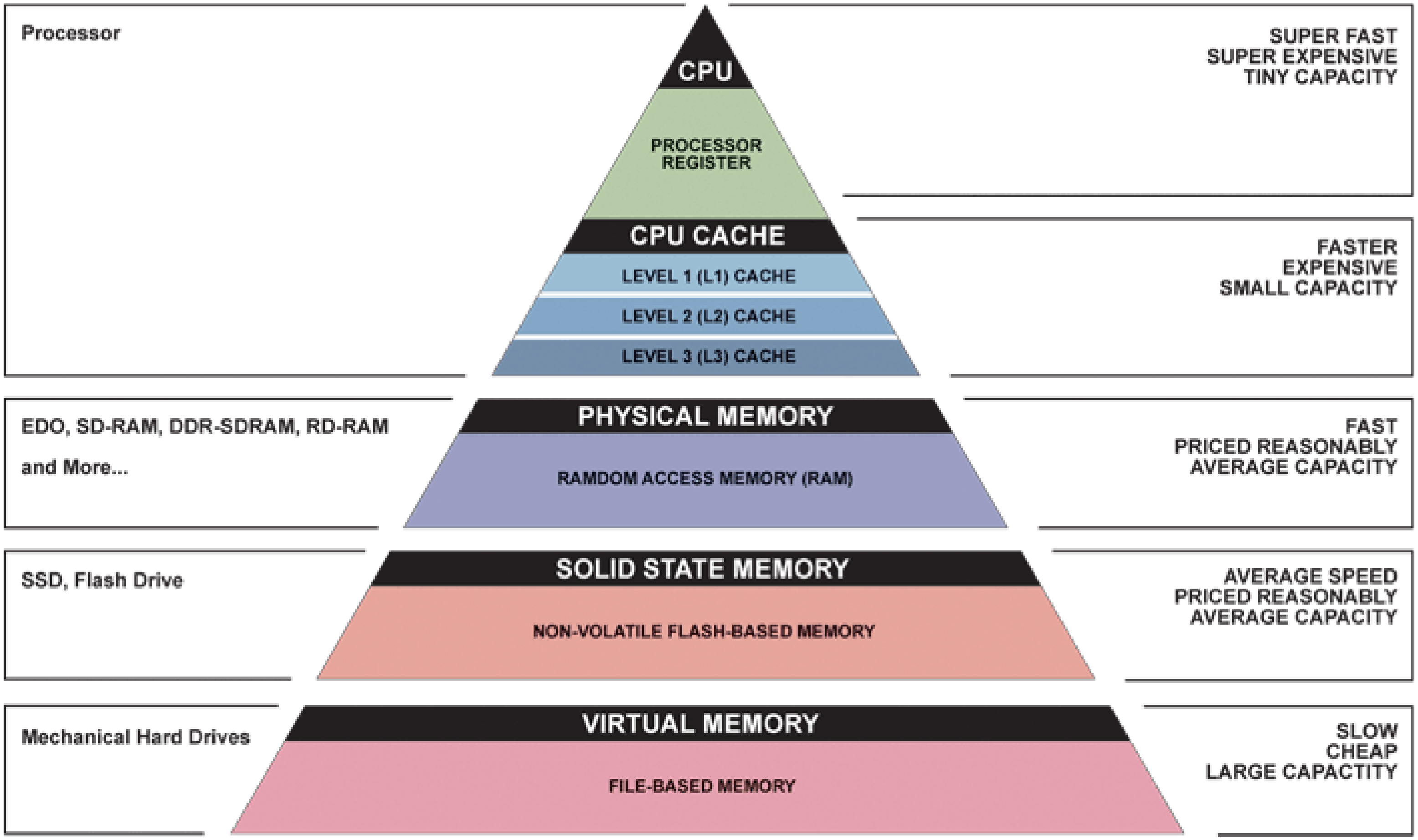
```
int a =0;
int b = 100;
array c[];
for(i = 0; i < b ; i++){
   a += c[i];
}
return a;
```

The all of the slot of `array` is called.
[Spatial Locality]

The `a` is repeatedly called.
[Temporal Locality]

Core concept of Memory Hierarchy

- Combine several types of memory.
  - Mix the cheap slow memory, with the expensive fast memory
- Apply the temporal locality
  - If the needed information is in the slower memory, moves to faster one.
  - If there is unused information is in the faster memory, moves to slower one.
- Apply the spatial locality
  - When move the information from the slower memory, move the nearby information as well



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

Cache

- A small, fast memory used to store data and instructions that is recently accesses.
  - Normally, the cache is a subset of data in the slower device.

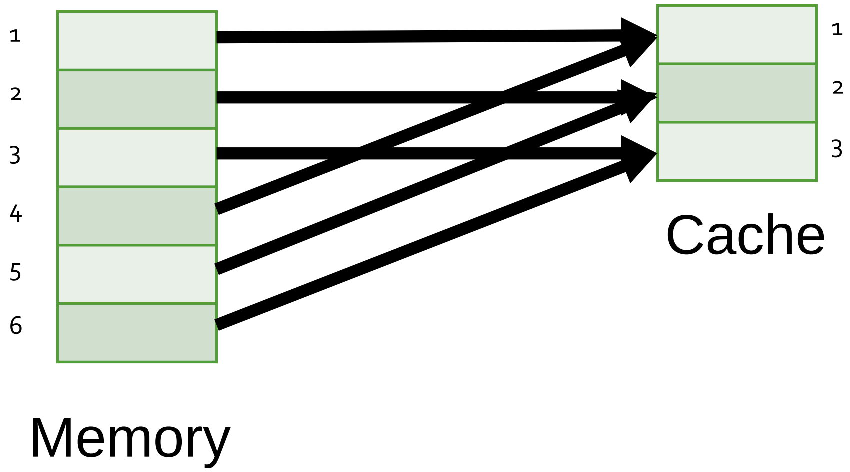
- The smaller faster device at higher level works as a cache for the information from the slower, larger device at the lower level.

Library example

- The library is bigger, slower device.
- The book is the information.
  - The book is existed in the library.
- The desk is a smaller, faster device where you transfer the needed book from the library to.
  - There can be no book on the desk that does not come from library.
- The books from the desk can be easier to find than finding from the library.

Basics of Cache

- The data is **directed mapped** with the address in memory.



Basics of Cache

- The formula to calculate the memory address and the cache location :

Memory address *mod* number of block in cache

- Ex: There 8 words in cache from 000 to 111.

Memory Address	Associated Cache Address
00001 (1)	001 (1 mod 8 = 1)
00101 (5)	101 (5 mod 8 = 5)
01001 (9)	001 (9 mod 8 = 1)
01101 (13)	101 (13 mod 8 = 5)

Issues with direct mapping

- How can you tell if the item cached in slot 101 came from 00101, 01101?
  - Answer: you store the remaining bit for differentiating.
  - **Tags**
- How can you tell if there's any useful item there at all?
  - Answer: you store the special bit that indicate the validity of information
  - Default value is 0. The value will change to 1 when the information is valid
  - **Valid bit**

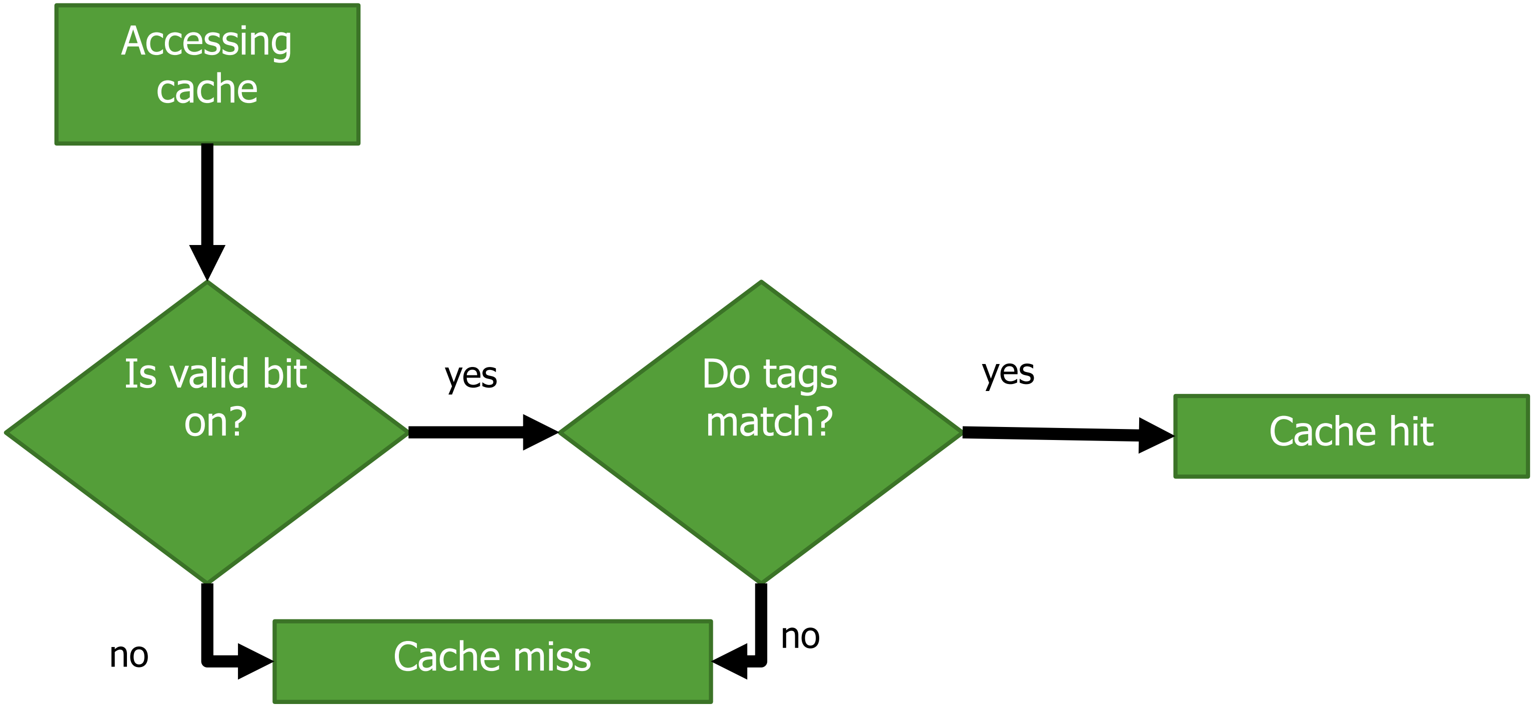
Example of of Tag and Valid Bit

Address Index	Valid bit	tag	data
001	1	01	0101
010	0	00	01011
101	1	01	0000

Cache Terminology

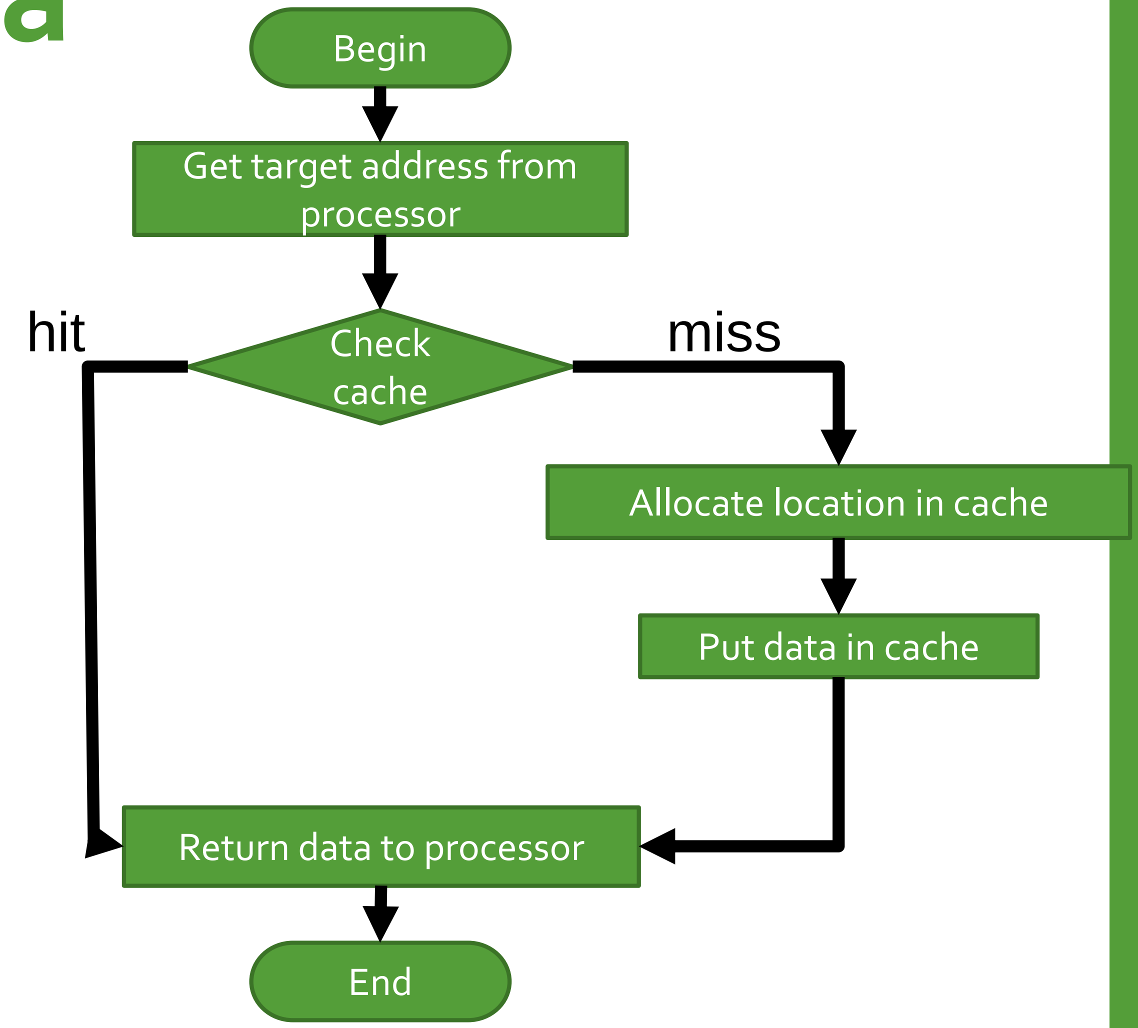
- Cache Hit
  - The request information is existed in the cache.
- Cache Miss
  - The request information is **not** existed in the cache.
  - When the cache miss takes place, the information will be transferred from main memory into cache.

Cache checking process



Accessing a Cache

- This is the flowchart on how to access the data in the cache.



Example

Target memory
10110
00100
00100
10110

Target memory : 10110

index	Valid bit	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	0		
111	0		

Target memory
10110
00100
00100
10110

CACHE MISS : no data at 110

Target memory : 10110

index	Valid bit	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	**1**	**10**	**MEMORY(10110)**
111	0		

Get data from memory!

Target memory : 00100

index	Valid bit	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	1	10	MEMORY(10110)
111	0		

CACHE MISS : no data at 100

Target memory
10110
00100
00100
10110

Target memory : 00100

index	Valid bit	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	**1**	**00**	**MEMORY(00100)**
101	0		
110	1	10	MEMORY(10110)
111	0		

Get data from memory!

Target memory : 00100

index	Valid bit	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	**1**	**00**	**MEMORY(00100)**
101	0		
110	1	10	MEMORY(10110)
111	0		

Target memory
10110
00100
00100
10110

CACHE HIT : the data exist!

Target memory : 00100

index	Valid bit	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	**1**	**00**	**MEMORY(00100)**
101	0		
110	1	10	MEMORY(10110)
111	0		

Target memory
10110
00100
00100
10110

CACHE HIT : the data exist!

Handling Writes

- Different from accessing the cache.
- When you try to write the new data to memory, you will write the data to the cache first.
  - So, at this point, the data in cache and the data in the main memory is different.
  - We call *inconsistent*.

Handling Writes: Solution

- We call this method ***walk-through scheme***.
- Just write the data to both memory and cache.
  - This can be applied when the information is in the cache.
- When the data is not in the cache (write miss),
  - We first have to fetch the information from memory to cache.
  - Then, write the new value to cache.
  - And, write the new value to memory using full address.