# Memory

- Data and instructions are transferred from memory to CPU via data buses.

## RAM

- Random Access Memory - read-write memory.
  - DRAM - Dynamic
  - SRAM - Static

## DRAM

- Refreshed every few (MS) to avoid data loss.
- leak charge over time.
- Bits are stored in capacitors
- DRAM consist of capacitors.

Benefits

- Low costs
- Small
- Low power consumption
- 1 transistor + 1 capacitor

Used for main memory

## Static RAM SRAM

- A bit is stored by 6 transistors as D flip-flops.

Benefits

- Very fast
- No need to refresh
- Large area
- 6 transistors/cell

Cons

- High power
- High cost

Used for cache

- 2 lines : bit and negated bit line

## ROM

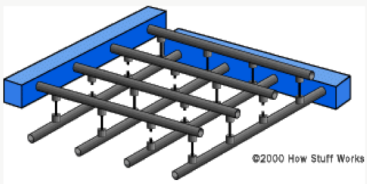
- Non volatile, isn't lost when power off.
- Used where data does not need to change.

5 types of ROM:

- ROM
- PROM
- EPROM
- EEPROM
- Flash

## PROM (Programmable ROM)

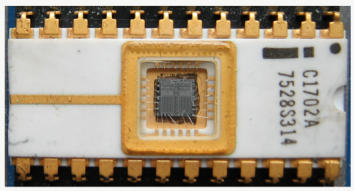
- Not changeable
- gridline structure
- High voltage 'blows' the fuse setting a zero



PROM

## EPROM (Erasable PROM)

- Whole chip can be reset by UV light.




## EEPROM (Electronically EPROM)

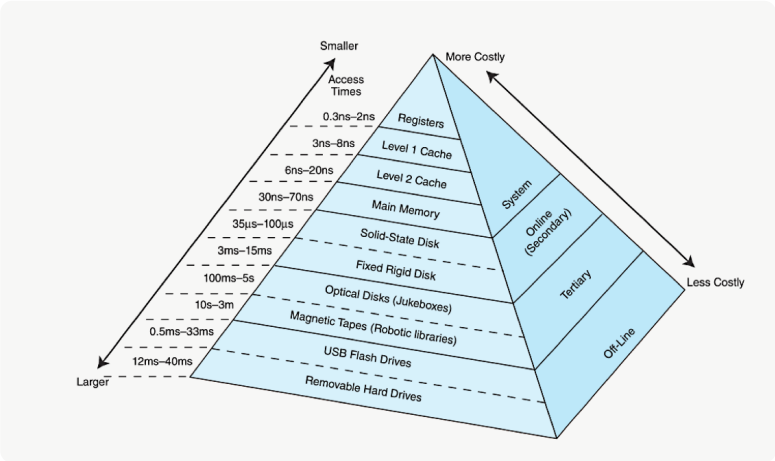
- Changed by electric field.
- Can change portions.
- Special equipment not needed.
- Erase one byte at a time, slow.

# Flash Memory

- Modern EEPROM
- Data stored on floating transistors.
- Nand & Nor are 2 main types.
- USB, Phones

# Memory heirachy

In order to make a device efficient and cost effective.




- Communication is between pairs of levels.
- The unit of information transfer is a block.

1) CPU will fetch from newest memory (hit)
2) If not there, then next memory. (miss)
3) Once, located, then goes to memory.

Miss rate = 1 - hit rate

# Cache

- 'Small, local' storage medium.
- Things which are used often.

# Data access times

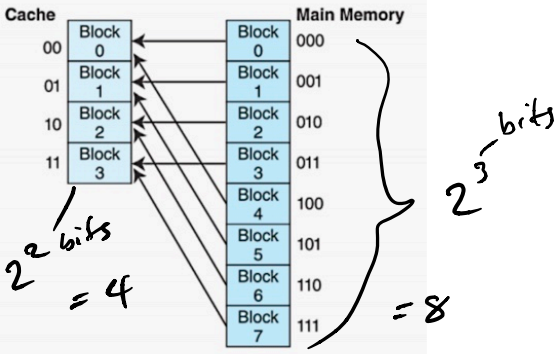
- Read from register: 1cycle (2 ns)
- Read from 1st level (L1) cache: 1 to 2 cycles (2-4 ns), 8K – 64K
- Read from 2nd level (L2) cache: 4 to 5 cycles (8-10 ns), 256K – 512K
- Read from main memory: 8 to 30 cycles (16-60 ns)
- When accessing from L1 or L2 cache we can get data faster

# Cache mapping

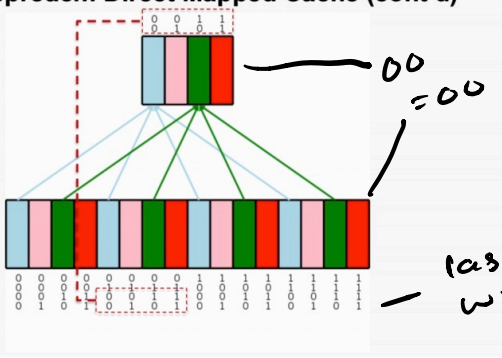
- When the CPU requests data, it first generates a main memory address.
- This address needs to be converted to cache by cache mapping.

## A Simple Approach: Direct Mapped Cache

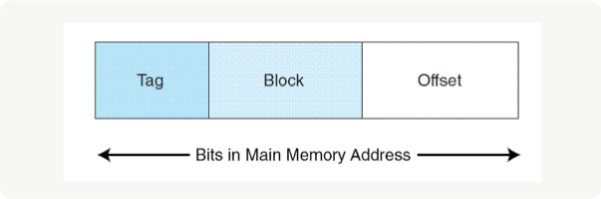
- 8 block locations in main memory
  - Memory block address = 3 bits
- Cache with 4 blocks
  - Cache block address (2 bits)
    - Lower 2 bits of memory block address
    - (main memory address) *modulo* 4 blocks



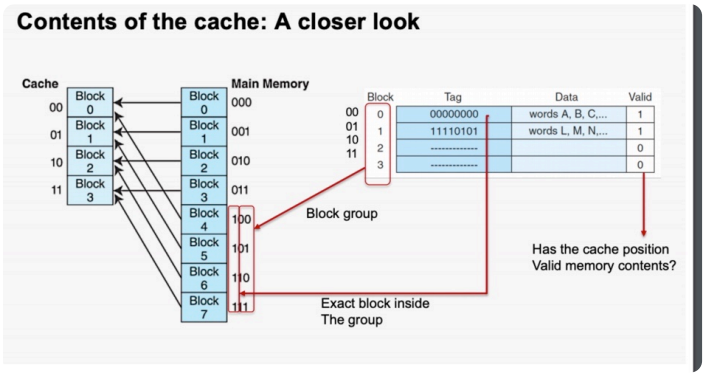

## A Simple Approach: Direct Mapped Cache (cont'd)




last 2 digits map with same address on cache



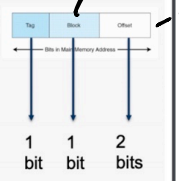

- The **block** field selects a unique group of blocks of cache.
- The **tag** field selects a unique block in the group
- The **offset** identifies an address within the block.




cache 2¹

**Cache and Bits, Example 6.1 in Book**

- First, we need to determine the address format for mapping, see the graph below.
- Each block is 4 bytes and is byte-addressable → **offset** field must contain 2 bits (00, 01, 10, 11)
- There are 2 blocks in cache, so the **block** field must contain 1 bit (0 for 1st block, 1 for 2nd block);
- The **memory (physical) address** requires 4 bits because there are a total of 16 words (4 blocks x 4 words each) and 16=2⁴).




4 bytes
$2^2$ - so 2 bits

Tag bits = Physical address – Cache Address = 4 – (2+1) = 1

$4 - (2+1) = 1$

tag bits = 1

- Tag bits = Most significant bits in the main memory eg if 4 bits 00 11 . 00 is the tag bit. 11 is the cache address.

eg. Main memory = 128 bytes Cache = 32

MM. 128 bytes = $2^7$ = 7 bits

Cache 32 $= 2^5 = 5$ bits

Each block = 4 $= 2^2 = 2$ bits

off set = 2 bits

block = 3 bits

tag bits = 2

$TB = PA_{full} - CA_{cache}$

$TB = 7 - 5$

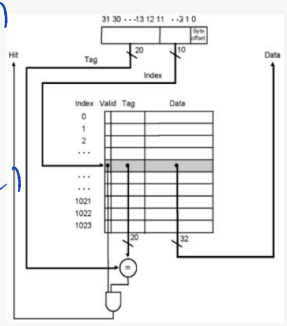
To check if present in cache, you will use the cache address, to check which block in cache, thereafter tag bits.

**Cache Memory Efficiency**

- Memory Efficiency
  - Cache memory size
    - 1024 32 bits = 32K bits
  - Tag memory size
    - 1024 20 bits = 20K bits
  - Valid bit
    - 1024 1 bit = 1K bits
  - Efficiency
    - 32 bits / 53 bits = 60.4%

32+20+1 = 53

32+20+1 = 53




$$EAT = H \times Access_C + (1-H) \times Access_{MM}$$

- For example: $Access_C$=10ns, $Access_{MM}$=200ns, H=0.99
  - EAT = 0.99(10ns) + 0.01(200ns) = 9.9ns + 2ns = 11.9ns.

## Split vs Combined Caches

- **Split caches** for data and instructions (Harvard Architecture)
  - Higher miss rate due to their smaller size
  - Higher bandwidth due to separate data paths
  - Data and instructions can be cached simultaneously
- **Combined caches**
  - Lower miss rate due to their size
  - Lower bandwidth due to sharing of data paths
  - Data and instructions cannot be cached simultaneously