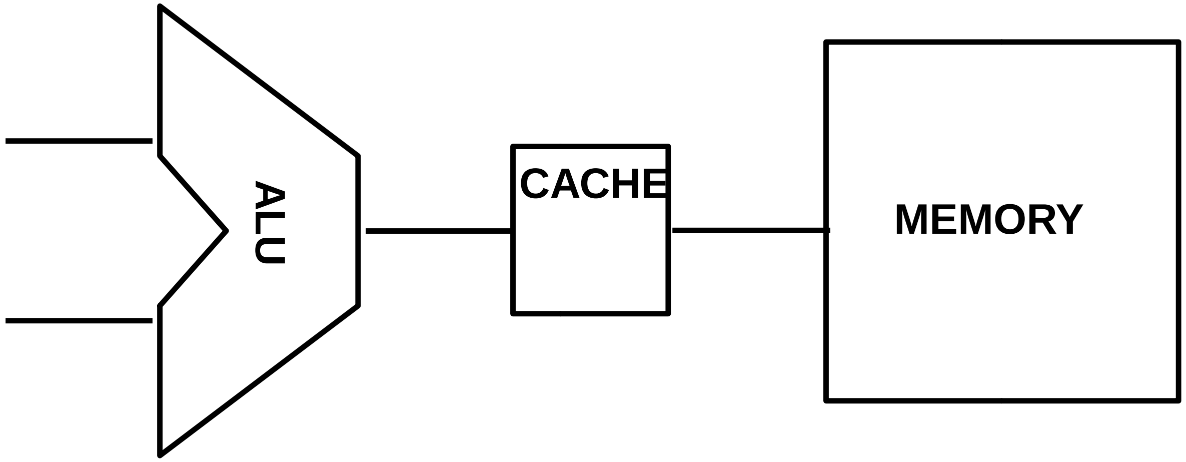
CHAPTER 6-4

Memory

By Pattama Longani
Collage of arts, media and Technology

Books in a library



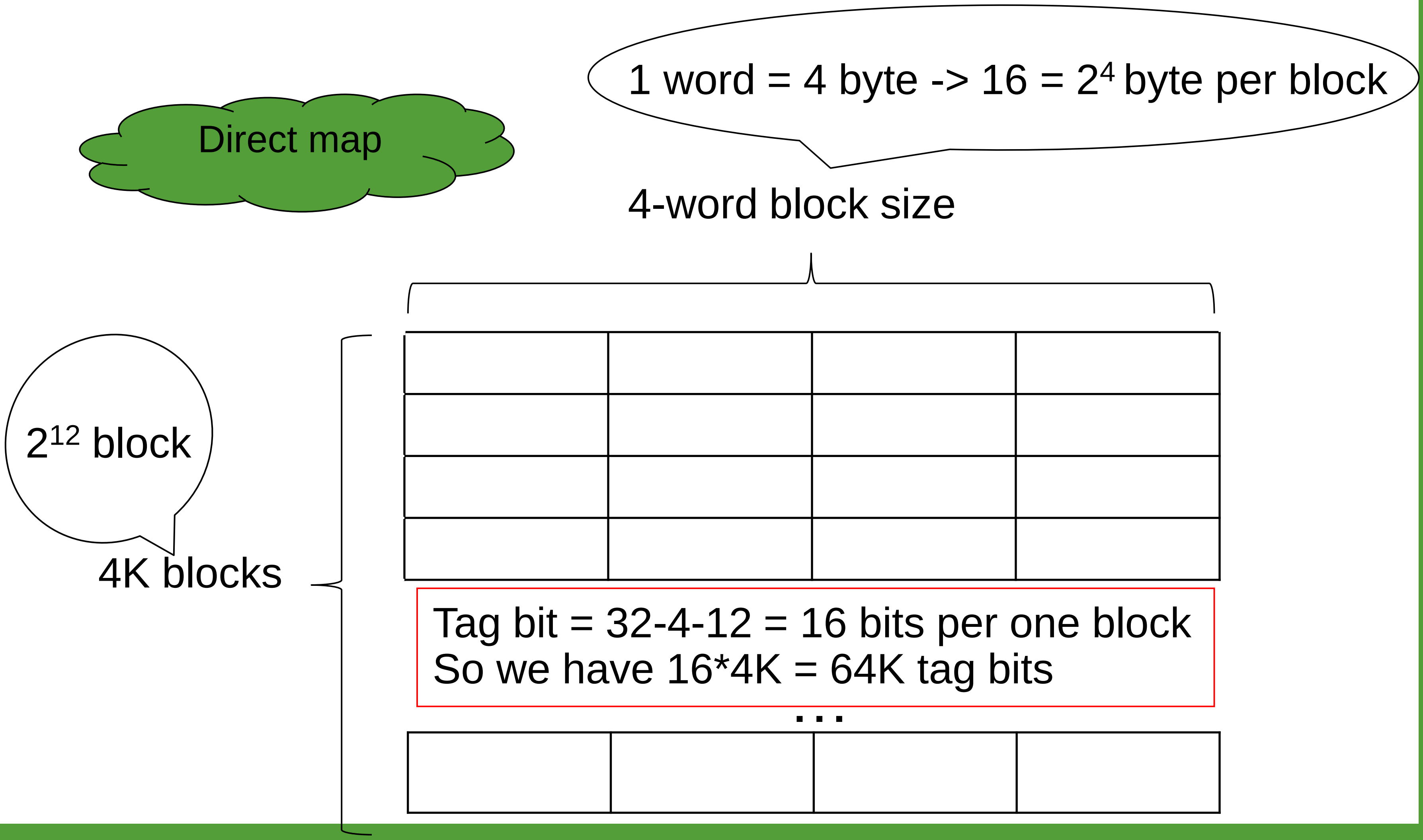
http://www.gettyimages.com/detail/photo/college-students-studying-at-table-royalty-free-image/485207369
http://commons.wikimedia.org/wiki/File:Bookshelf.jpg

Size of Tags vs. Set Associativity

- Ex:
- Assuming a cache of 4K blocks, a 4-word block size, and a 32-bit address, find the total number of sets and the total number of tag bits for caches that are direct mapped, two-way and four-way set associative, and fully associative.
- Note: Increasing associativity requires more comparators and more tag bits per cache block.

Size of Tags vs. Set Associativity

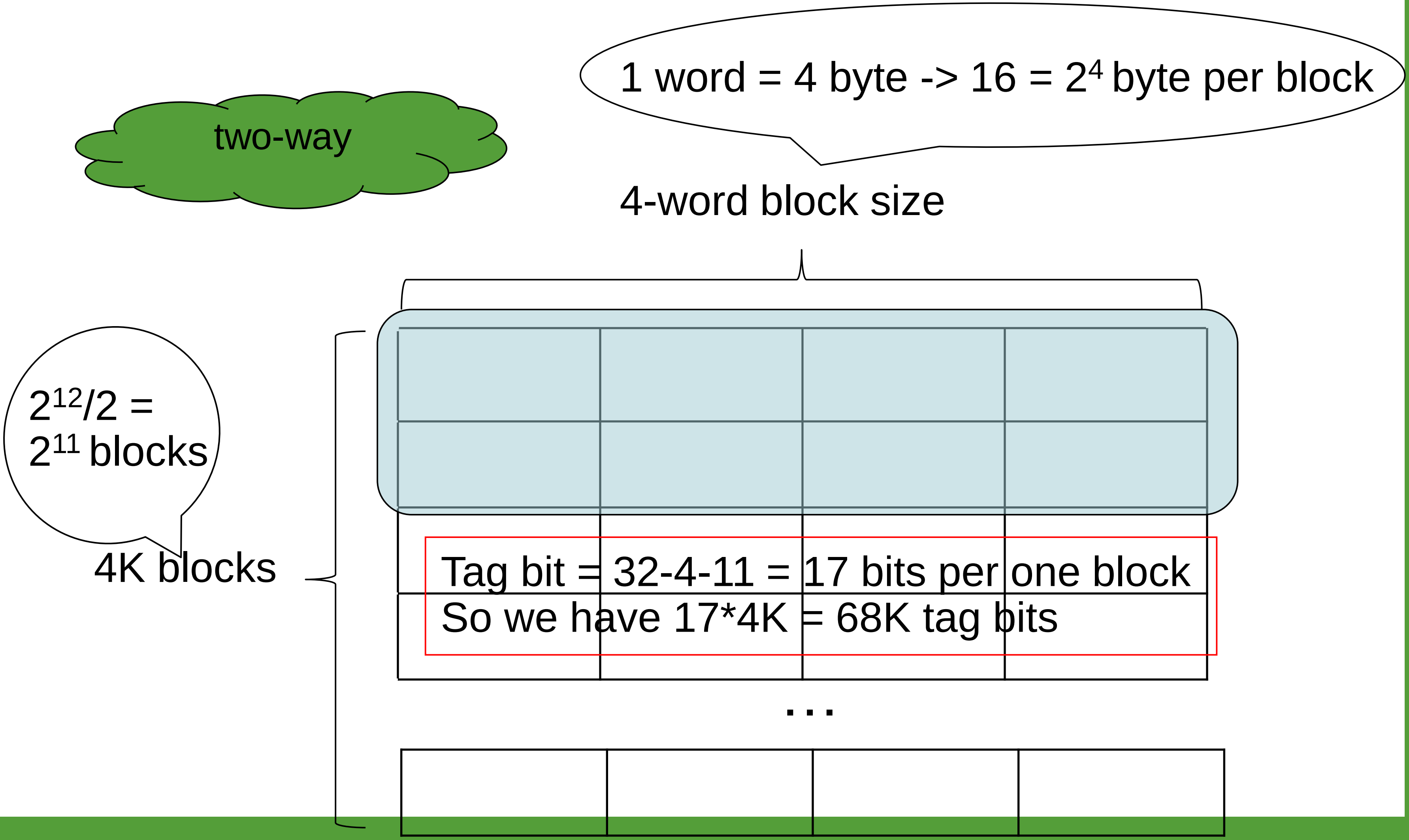
- 4K blocks, a 4-word block size, and a 32-bit address



Tag bit = 32-4-12 = 16 bits per one block
So we have 16*4K = 64K tag bits

Size of Tags vs. Set Associativity

- 4K blocks, a 4-word block size, and a 32-bit address

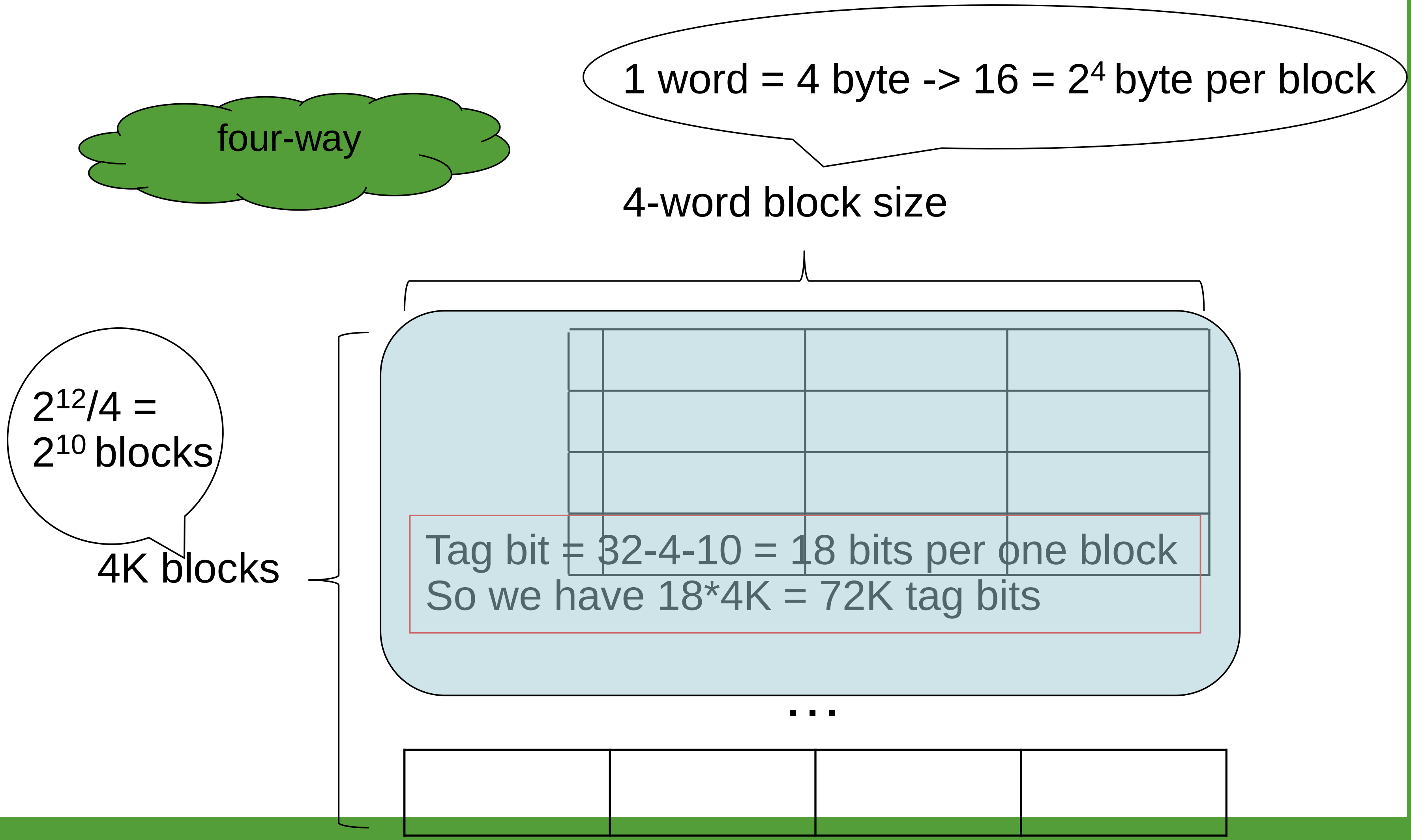


two-way
1 word = 4 byte -> 16 = 2^4 byte per block
4-word block size
$2^{12}/2$ = 2^{11} blocks
4K blocks
…

Tag bit = 32-4-11 = 17 bits per one block
So we have 17*4K = 68K tag bits

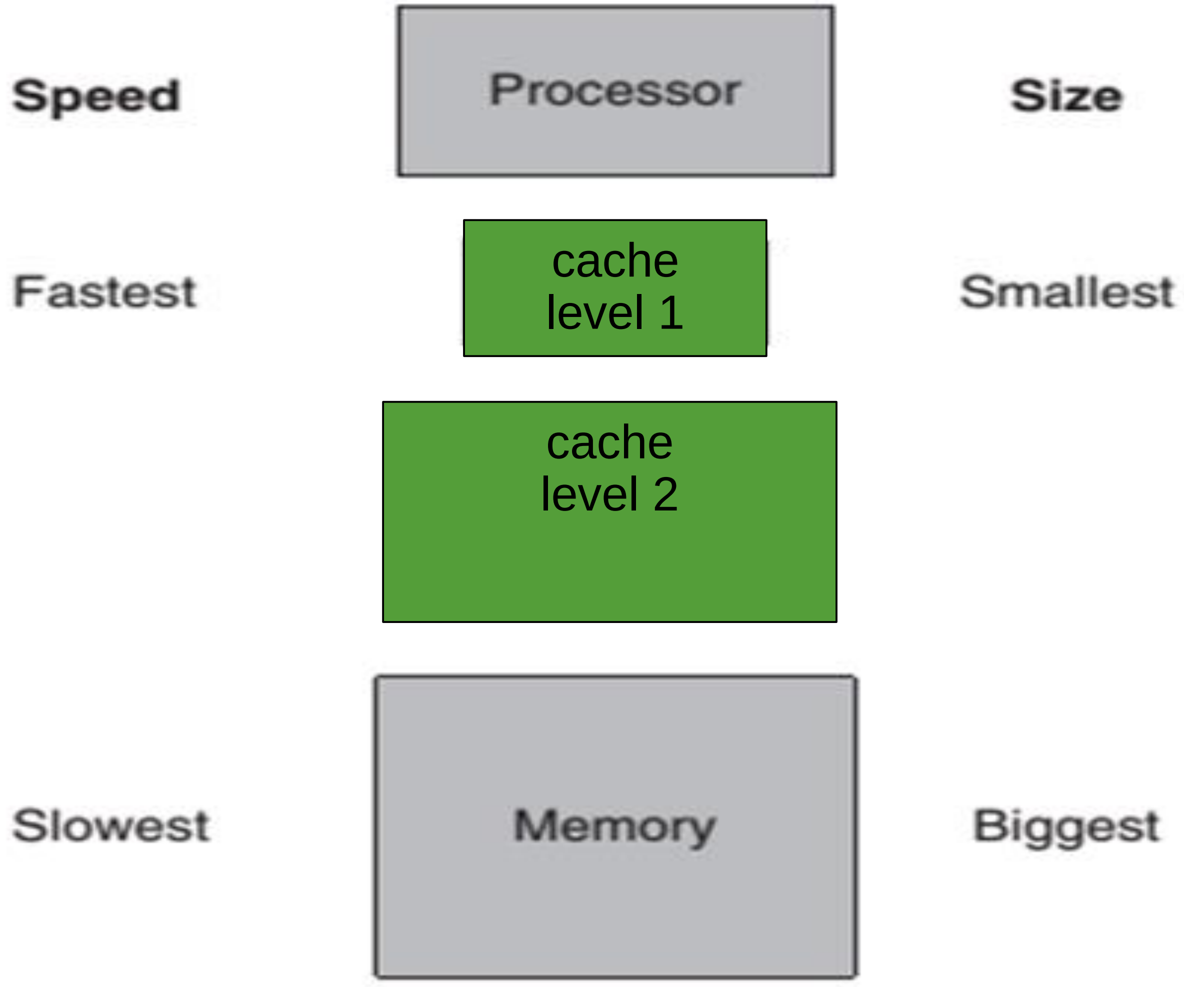
Size of Tags vs. Set Associativity

- 4K blocks, a 4-word block size, and a 32-bit address



four-way
1 word = 4 byte -> 16 = 2^4 byte per block
4-word block size
$2^{12}/4 = 2^{10}$ blocks
4K blocks
Tag bit = 32-4-10 = 18 bits per one block
So we have 18*4K = 72K tag bits
…

Multilevel Caches



memory hierarchy

- Suppose we have a processor with a base CPI of 1.0, assuming all references hit in the primary cache, and a clock rate of 4 GHz. Assume a main memory access time of 100 ns, including all the miss handling. Suppose the miss rate per instruction at the primary cache is 2%. How much faster will the processor be if we add a secondary cache that has a 5 ns access time for either a hit or a miss and is large enough to reduce the miss rate to main memory to 0.5%?

memory hierarchy

- Base CPI = 1
- Main memo access time = 100 ns
- Clock rate = 4GHz ->1/4 = 0.25 clock cycle time

0.25 ns -> 1 clock cycle

100 ns -> (1/0.25)*100 = 400 clock cycle

memory hierarchy

The miss penalty to main memory is

$$\frac{100\ \text{ns}}{0.25\ \frac{\text{ns}}{\text{clock cycle}}} = 400\ \text{clock cycles}$$

$$\text{Total CPI} = \text{Base CPI} + \text{Memory-stall cycles per instruction}$$

For the processor with one level of caching,

$$\text{Total CPI} = 1.0 + \text{Memory-stall cycles per instruction} = 1.0 + 2\% \times 400 = 9$$

memory hierarchy

two levels of caching

$$\frac{5\text{ ns}}{0.25\,\dfrac{\text{ns}}{\text{clock cycle}}} = 20\text{ clock cycles}$$

$$\begin{aligned}\text{Total CPI} &= 1 + \text{Primary stalls per instruction}\\ &\quad + \text{Secondary stalls per instruction}\\ &= 1 + 2\% \times 20 + 0.5\% \times 400 = 1 + 0.4 + 2.0 = 3.4\end{aligned}$$

Thus, the processor with the secondary cache is faster by

$$\frac{9.0}{3.4} = 2.6$$

Memory

- **volatile memory :** storage, such as DRAM, that retains data only if it is receiving power.
- **nonvolatile memory :** a form of memory, such as magnetic disk, that retains data even in the absence of a power source and that is **used to store programs between runs.**

Memory

- **main memory** also called **primary memory.** Memory used to **hold programs while they are running**
- **secondary memory** Nonvolatile memory used to **store programs and data between runs;** typically consists of magnetic disks also call hard disk, flash memory in today's computers.

Virtual Memory

- **virtual memory** : a technique that uses main memory as a “cache” for secondary storage.
- two major motivations for virtual memory:
  - to allow efficient and safe sharing of memory among *multiple programs*
  - to remove the programming burdens of a small, limited amount of main memory.

efficient and safe sharing of memory

- multiple programs to share the same memory
  - Protect the programs from each other -> ensuring that a program can only read and write the portions of main memory that have been assigned to it.

efficient and safe sharing of memory

- Main memory need contain only the active portions of the many programs
- We cannot know which programs will share the memory with other programs when we compile them.
- the memory change dynamically while the sharing programs are running.

efficient and safe sharing of memory

- we would like to compile each program into its own ***address space***—a separate range of memory locations accessible only to this program.
- Virtual memory translate a program's address space to **physical addresses**.
  - This translation process enforces **protection** of a program's address space from other programs.

remove the programming burdens

- Formerly, if a program became too large for memory, it was up to the programmer to make it fit.
- Programmers divided programs into pieces and then identified the pieces that were mutually exclusive, cannot occur at the same time.

remove the programming burdens

- These *overlays* were loaded or unloaded under user program control during execution, with the programmer ensuring that the program never tried to access an overlay
- that was not loaded and that the overlays loaded never exceeded the total size of
- the memory

remove the programming burdens

- These *overlays* were loaded or unloaded under user program control during execution, with the programmer ensuring that the program never tried to access an overlay that was not loaded and that the overlays loaded never exceeded the total size of the memory
- both code and data. Calls between procedures in different modules would lead to overlaying of one module with another.

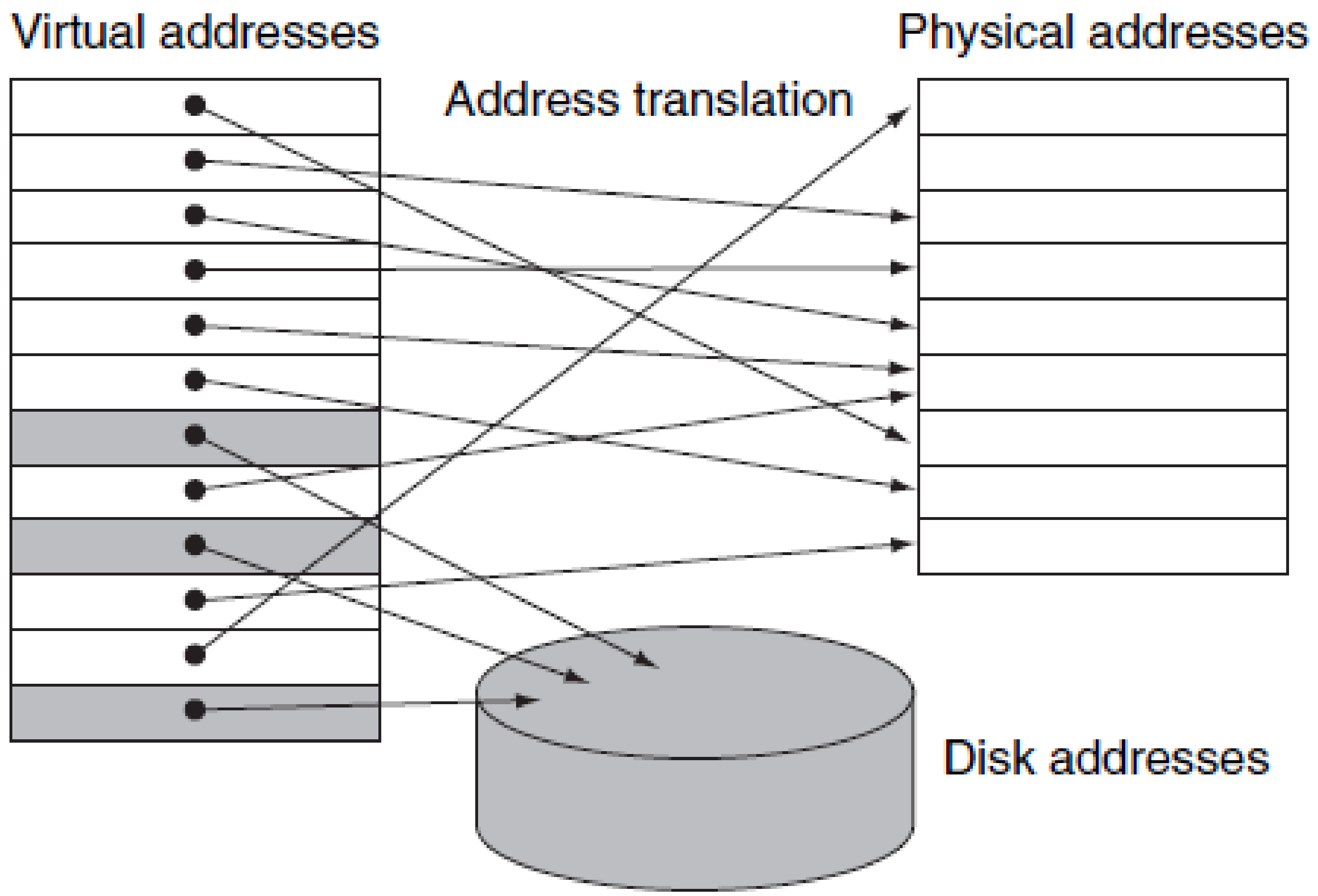
remove the programming burdens

- Virtual memory, which was invented to relieve programmers of this difficulty
  - automatically manages the two levels of the memory hierarchy represented by main memory (DRAM) and secondary storage (magnetic disk).

Virtual Memory

- **page** : a virtual memory block.
- **page fault** : and a virtual memory miss
- **virtual address :** address of the virtual memory, the processor produces, which is translated by a combination of hardware and software to a *physical address*, which in turn can be used to access main memory
- **address translation** (address mapping) The process by which a virtual address is mapped to an address used to access memory.

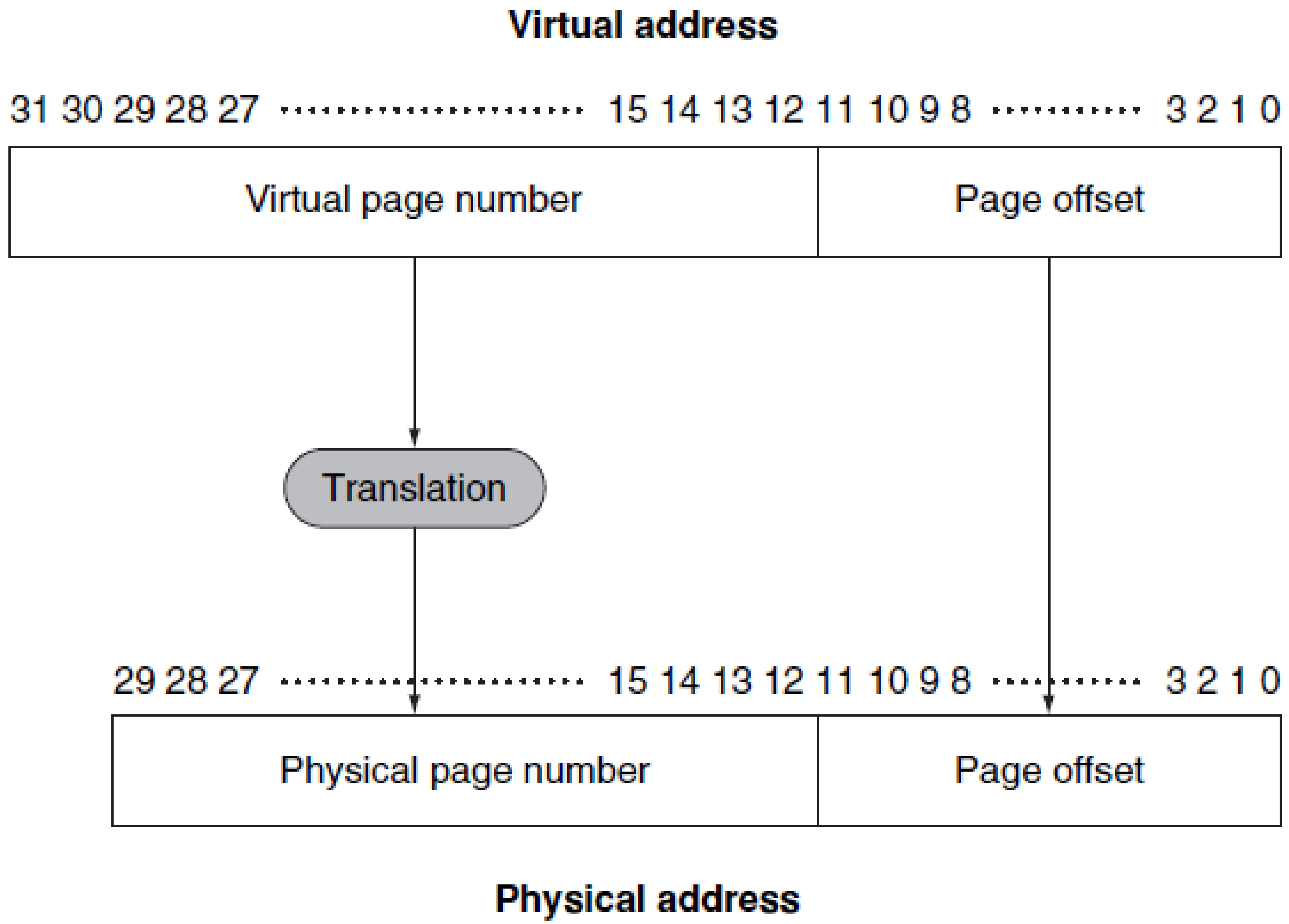
Virtual Memory



Virtual Memory

- we can think of a virtual address as the title of a book and a physical address as the location of that book in the library
- In the pass, fully associative, there are **relocation** maps the virtual addresses used by a program to different physical addresses before the addresses are used to access memory.
- all virtual memory systems in use today relocate the program as a set of fixed-size blocks(pages), thereby eliminating the need to find a contiguous block of memory to allocate to a program

Virtual Memory



designing virtual memory systems:

- Pages should be large enough to try to amortize the high access time.
- Organizations that reduce the page fault rate are attractive.
- Page faults can be handled in software because the overhead will be small compared to the disk access time.
- **Write-through** will not work for virtual memory, since writes take too long.
- Instead, virtual memory systems use **write-back.**