Math Review: Counting and Addressing

ICS332

Operating Systems

Henri Casanova (henric@hawaii.edu)

Disclaimer

- The content in these slides will be obvious to many of you
  - This is a good thing!
- But when I teach this course, this material often causes problems
  - Although it’s not technically OS material
- And we need it to be solid for this second part of the semester
  - And for the rest of your life!
- So I am presenting is here now, so that students who have difficulties with this have plenty of time to practice before it becomes critical for this course

Units of Storage

- The smallest unit of information is the bit
  - Anybody knows why it’s called a bit?
- The basic unit of memory is a byte
  - 1 Byte = 8 bits
  - 1 KiB = 2^{10} Byte = 1,024 bytes
  - 1 MiB = 2^{10} KiB = 2^{20} bytes (1 Million) (mega)
  - 1 GiB = 2^{10} MiB = 2^{30} bytes (1 Billion) (giga)
  - 1 TiB = 2^{10} GiB = 2^{40} bytes (1 Trillion) (tera)
  - 1 PiB = 2^{10} TiB = 2^{50} bytes (1,000 Trillion) (peta)
  - 1 EiB = 2^{10} PiB = 2^{60} bytes (1 Million Trillion) (exa)

- Often the “i” above is missing, which is not great
- **1GB = 10^9 bytes, but 1GiB = 2^{30} bytes!**

- **You have to know the units and order above!!!**

Exponents, Logarithms

- We'll use Exponents:
  - $\alpha^x \cdot \alpha^y = \alpha^{x+y}$
  - $\alpha^{-x} = 1 / \alpha^x$
  - $\alpha^x / \alpha^y = \alpha^{x-y}$
- But we'll do only powers of 2:
  - $2^x \cdot 2^y = 2^{x+y}$
  - $2^{-x} = 1 / 2^x$
  - $2^x / 2^y = 2^{x-y}$
- We'll use Logs:
  - $\log_\alpha \alpha^n = n$
- But only for base 2:
  - $\log_2 2^n = n$
  - Not to be confused with the natural logarithm, ln, which is really $\log_e$ ($\ln e^x = x$), and which you've seen in Calculus courses
  - In computer science: $\log_2$ is often just written as log, especially when we deal with asymptotic computational complexities (e.g., $O(\log_2 n) \sim O(\log_{10} n)$)
- I am going to assume the above is solid for everyone, but if it's not, you know what you have to do…

Counting Bytes

- When studying operating systems, we often need to count “chunks” of bytes in some memory space
- Example #1: how many 1MiB chunks are there in a 8MiB file?
  - easy: 8
- Example #2: how many 4KiB chunks are there in a 8GiB file?
  - not so easy perhaps?
- The way to do this: use powers of 2
  - We want results in powers of 2 anyway because numbers are typically too large to just write them out in decimal conveniently
  - All our stuff will be in powers of 2, therefore none of our integer divisions will have remainders

Examples

- How many groups of 12 parking spots are there in 252-spot parking lot?
  - answer: 252 / 12 = **21**   (remainder = 0 in this case)
- How many groups of x thingies are there in a set of y thingies?
  - answer: **y / x** (plus perhaps a remainder)
- How many 2 KiB chunks are there in 1 GiB?
  - 1 GiB = 2^{30} bytes
  - 2 KiB = $2 \times 2^{10} = 2^{11}$ bytes
  - answer: $2^{30} / 2^{11}$ = **2^{19} chunks**
- How many 8 KiB chunks are there in 128 MiB?
  - 128 MiB = $2^{7} \times 2^{20} = 2^{27}$ bytes
  - 8 KiB = $2^{3} \times 2^{10} = 2^{13}$ bytes
  - answer: $2^{27} / 2^{13}$ = **2^{14} chunks**

Addressing

- We often partition thingies into chunks
  - Partition a pie into slices
  - Partition a computer's memory into bytes
  - Partition a file into "blocks"
  - Partition an address-space into "pages"
  - Partition a disk into "sectors"
- After partitioning we need to address the chunks
- Addressing means: "refer to something using a name/number"

- We already know what addresses are:
  - Each byte in RAM is addressed by a number (called "the address")
  - An address is stored in binary form in the computer (like all numbers)
  - We can then use these addresses, for instance in instructions ("store value 00110011 at address 1101001")

How Many Address Bits?

- **Key question:** what is the range of addresses that we need to address all chunks (uniquely)?
- We also want the smallest range not to waste address bits by having large addresses that are not used
  - For saving on storage (we store addresses as data to do indirection)
- Example:
  - I have 7 dogs
  - I want to “address” them via binary addresses
  - I should use 3 address bits: 000, 001, 010, 011, 100, 101, 110
  - With 3 bits I can address $2^3 = 8$ dogs, so we’re “wasting” one slot
  - With 2 bits I can address only $2^2 = 4$ dogs (00, 01, 10, 11), so that’s not enough
  - I don’t want to use 4-bit addresses because when I need to store dog names as data, then I’d be wasting 1 bit of storage per house
    - i.e., all addresses would have the same leftmost bit, so that leftmost bit contains zero information

How Many Address Bits?

- If one has 2^n thingies, then one uses n-bit addresses to address the thingies
  - fewer, and you can’t address them all
  - more, and you’re wasting address bits
- Conversely, if on has n thingies, then one needs $\lceil \log_2 n \rceil$-bit addresses
  - Example with 7 houses: $\log_2 7 \sim 2.8074$, therefore we should use $\lceil \log_2 7 \rceil = 3$ bits
- In this course we’ll almost always have a number of thingies that’s a power of 2
  - After all we “build” the system and choose what we use
  - And as you can see above in red, powers of 2 are convenient when using binary addresses

Some More Discrete Math

- Say you have a parking lot that consists of a long row of N parking spots, **numbered 0 to N − 1**
- We structure this long row into blocks of n parking spots (assume n divides N)
- Here are two simple discrete math "results":

1. The x-th spot in the parking lot ($0 \leq x < N$) is the (x mod b)-th spot in the ($\lfloor x/b \rfloor$)-th block

2. The y-th spot in the z-th block is the ($z \times b + y$)-th spot in the parking lot

- Let's see this on an example…

Parking Lot Example

- Say we have a parking lot with 3000 spots, and we structure them in blocks of 100 spots
- What is the index of spot 2212 in its block?
  - 2212 mod 100 = 12
- In what block is spot 2212?
  - 2212 / 100 = 22 (integer division!)
- What is the global index of spot 5 in block 20?
  - $20 \times 100 + 5 = 2005$
    (because the first block is block 0)

The End

- You must be absolutely comfortable with all this since we'll soon be doing counting/addressing all the time and I will skip the intermediate steps

- Besides, being a computer scientist implies that you can count and address things, and that you're not fazed by powers of 2!
  - Sometimes a first interview question is: what's 2 to the 8? 😳

- Let's look at practice slides…