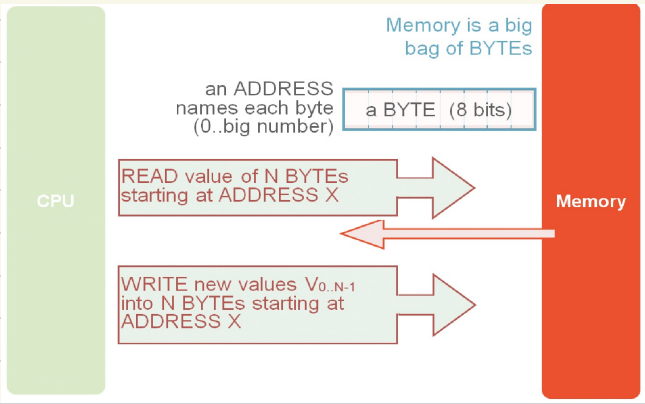
# Lec 1. 2 Jan 10 Wed

A simple computing machine

- Memory
  - bits ← data
  - Program Instruction & state
- CPU
  - read instruction / data from memory
  - Compute and write back to memory

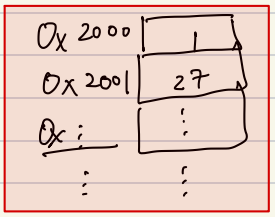
If 32-bit computer: need 4 bytes for one byte. Not efficient !!

- We don't have to store addresses. Address → data is taken for granted.
- Pointers are: variable = an address




Memory

- Naming
  - address in 1 byte = 8 bits, e.g., $\square\square\square\square\square\square\square\square_2$
    - * Every byte in memory ⇔ unique address
    - * Some machines 32-bit address
      - {64-bit … etc "64ĩ"




$2^{10} \approx 1000$

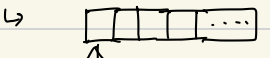
$2^{20} \approx 1\,000\,000$

$2^{30} \approx 1\,000\,000\,000$

4 GB address ? $2^{32}$ possible for 32-bit address

$2^{32}$ = 4GB

- Access
  - There can be overflow: things too big.
    - ↳ Hence, chunks: contiguous, power-of-two size



addr. for the chunk

usually no need for "termination" character because: int, LongInt, Long, Double, Float all perscribed:

↘ But strings: NOT

Integer Data Types by Size

| # bytes | # bits | C | Java | Assembly |
|---|---|---|---|---|
| 1 | 8 | char | byte | b byte |
| 2 | 16 | short | short | w word |
| 4 | 32 | int | int | l long |
| 8 | 64 | long | long | q quad |

We will use only 32-bit integers