Conceptually, memory is a large array of bytes that can be accessed from your program by specifying a starting address and a byte count.

\[
\text{get}(\text{addr1, k}) \quad \rightarrow \quad \text{[M(addr1), ..., M(addrk)]}
\]

\[
\begin{align*}
\text{addr1} \\
\text{addr2} \\
\vdots \\
\text{addrk}
\end{align*}
\]

Storage Technologies
Concretely, memory is a collection of technologies that store data in multiple places and formats. The memory controller(s) map addresses onto commands to retrieve bytes from these technologies.
Random-Access Memory (RAM)

Key Features

- RAM is packaged as a chip
- The basic storage unit is a cell (one bit per cell)
- Multiple RAM chips form a memory.
Static RAM (SRAM)
- Each cell stores a bit with a 6-transistor circuit.
- Retains value indefinitely, as long as kept powered (volatile).
- Relatively insensitive to disturbances such as electrical noise.
- Faster but more expensive than DRAM.

Dynamic RAM (DRAM)
- Each cell stores a bit with a capacitor and transistor.
- Value must be refreshed every 10–100 ms (volatile).
- Sensitive to disturbances, slower and cheaper than SRAM.
Flash RAM (what’s in your ipod and cell phone)

- Each cell stores 1 or more bits on a “floating-gate” capacitor
- Keeps state even when power is off (non-volatile).
- As cheap as DRAM, but much slower

Note that flash has characteristics of RAM (random access), but also of ROM (non-volatile). It’s often considered a hybrid of both.
## RAM Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Trans. per bit</th>
<th>Access time</th>
<th>Persist?</th>
<th>Sensitive</th>
<th>Cost</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>6</td>
<td>1X</td>
<td>No</td>
<td>No</td>
<td>100X</td>
<td>cache memory</td>
</tr>
<tr>
<td>DRAM</td>
<td>1</td>
<td>10X</td>
<td>No</td>
<td>Yes</td>
<td>1X</td>
<td>main memory</td>
</tr>
<tr>
<td>Flash</td>
<td>1/2–1</td>
<td>10000X</td>
<td>Yes</td>
<td>No</td>
<td>1X</td>
<td>disk substitute</td>
</tr>
</tbody>
</table>
Conventional DRAM Organization

DRAM is typically organized as a $d \times w$ array of $d$ supercells of size $w$ bits.
Step 1(a): Row access strobe (RAS) selects row 2.
Step 1(b): Row copied from DRAM array to row buffer.
Step 2(a): Column access strobe (CAS) selects col 1.
Step 2(b): Supercell (2, 1) copied from buffer to data lines, and eventually back to the CPU.
64 MB memory module consisting of eight 8Mx8 DRAMs.

64-bit quadword at main memory address.

addr (row = i, col = j)

Memory controller

Memory Modules
DRAM and SRAM are *volatile memories*; they lose information if powered off.

*Nonvolatile memories* retain their value even if powered off.

- The generic name is read-only memory (ROM).
- This is misleading because some ROMs can be read and modified.
Types of ROM

- Programmable ROM (PROM)
- Eraseable programmable ROM (EPROM)
- Electrically eraseable PROM (EEPROM)
- Flash memory

Firmware: Program stored in a ROM

- Boot time code, BIOS (basic input/output system)
- Graphics cards, disk controllers
A *bus* is a collection of parallel wires that carry address, data, and control signals.

- Buses are typically shared by multiple devices.
CPU places address A on the memory bus.

load operation: movl A, %eax
Main memory reads A from the memory bus, retrieves word x, and places it on the bus.

load operation: movl A, %eax

Places x on the bus.
CPU reads word $x$ from the bus and copies it into register $\%eax$. 

load operation: movl A, $\%eax$
CPU places address A on bus. Main memory reads it and waits for the corresponding data word to arrive.

store operation: \texttt{movl \%eax, A}
CPU places data word $y$ on the bus.

store operation: `movl %eax, A`

CPU places data word $y$ on the bus.
Main memory reads data word $y$ from the bus and stores it at address $A$. 

\[
\text{store operation: } \text{movl } \%eax, A
\]
Disks consist of platters, typically each have two *surfaces* though not always.

Each surface consists of concentric rings called *tracks*.

Each track consists of *sectors* separated by gaps.
Aligned tracks form a cylinder. Read/write heads move in unison so are all on the same cylinder at any one time.
Capacity: maximum number of bits that can be stored. Vendors express this in terms of gigabytes (GB), where 1GB = 10^9 bytes.

Capacity is determined by these technology factors:

- **Recording density** (bits/in): number of bits that can be squeezed into a 1 inch segment of a track.
- **Track density** (tracks/in): number of tracks that can be squeezed into a 1 inch radial segment.
- **Areal density** (bits/in^2): product of recording and track density.
Modern disks partition tracks into disjoint subsets called **recording zones**.

- Each track in a zone has the same number of sectors, determined by the circumference of the innermost track.
- Each zone has a different number of sectors/track.
- **Why does this make sense?**
Computing Disk Capacity

Capacity = ( bytes/sector) \times (avg. sectors/track) \times 
(tracks/surface) \times (surfaces/platter) \times (platters/disk)

Example:
- 512 bytes/sector
- 300 sectors/track (on average)
- 20,000 tracks/surface
- 2 surfaces/platter
- 5 platters/disk

Capacity = 512 \times 300 \times 20000 \times 2 \times 5 = 30,720,000,000 = 30.72GB
The disk surface spins at a fixed rotational rate.

The read/write head is attached to the end of the arm and flies over the disk surface on a very thin cushion of air (around 0.1 microns).

By moving radially, the arm can position the read/write head over any track.
To read a sector on a disk requires:

- **Seek:** the read head is moved to the proper track.
- **Rotational latency:** the desired sector must rotate to the read head.
- **Data transfer:** the sector is read as it rotates under the read head.

Writing is the same.

Which of these do you suppose is longest?
The average time to access a target sector is approximately:

\[ T_{\text{access}} = T_{\text{seek}} + T_{\text{rotation}} + T_{\text{transfer}} \]

- **Seek time** \((T_{\text{seek}})\)
  - Time to position heads over cylinder containing the target sector.
  - Average \(T_{\text{seek}}\) is given by device documentation (e.g., 9 ms).

- **Rotational latency** \((T_{\text{rotation}})\)
  - Time waiting for first bit of target sector to pass under read/write head.
  - Average \(T_{\text{rotation}} = 1/2 \times 1/\text{RPMs} \times 60\text{sec}/\text{1min}\)

- **Transfer time** \((T_{\text{transfer}})\)
  - Time to read the bits in the target sector.
  - Average \(T_{\text{transfer}} = 1/\text{RPM} \times 1/(\text{average sectors/track}) \times 60\text{sec}/\text{1min}\)
Disk Access Time Example

Given:
- Rotational rate: 7,200 RPM
- Average seek time: 9 ms
- Average sectors/track: 400

Derived:
- Average $T_{\text{rotation}}$:
  \[\frac{1}{2} \times \left(\frac{60\text{sec}}{7200\text{RPM}}\right) \times 1000\text{ms/sec} = 4\text{ms}\]
- Average $T_{\text{transfer}}$:
  \[\frac{60}{7200\text{RPM}} \times \frac{1}{(400\text{ sectors/track})} \times 1000\text{ms/sec} = 0.02\text{ms}\]
- $T_{\text{access}}$: 9 ms + 4 ms + 0.02 ms
Important points:

- Access time is dominated by seek time and rotational latency.
- The first bit in a sector is the most expensive; the rest are basically free.
- SRAM access time is about 4ns / doubleword; DRAM about 60ns.
- Disk is about 40,000 times slower than SRAM, and 2,500 times slower than DRAM.
Modern disks present a simpler abstract view of the complex sector geometry.

The set of available sectors is modeled as a sequence of b-sized **logical blocks** $(0, 1, 2, \ldots)$.

Mapping between logical blocks and actual (physical) sectors:

- Is maintained by a hardware/firmware device called a disk controller.
- Converts requests for logical blocks into (surface, track, sector) triples.

Allows the controller to set aside spare cylinders for each zone.

This accounts for the difference between “formatted capacity” and “maximum capacity.”
I/O Bus

CPU chip

- register file
- %eax
- ALU
- bus interface
- system bus

I/O Bridge

- memory bus

main memory

I/O bus

- USB controller
  - mouse
  - keyboard
- graphics adapter
  - monitor
- disk controller
  - Disk

Storage Technologies
The CPU initiates a disk read by writing a command, logical block number, and destination memory address to a *port* (address) associated with the disk controller.

The disk controller reads the associated sector and performs a direct memory access (DMA) transfer into main memory.

When the DMA transfer completes, the disk controller notifies the CPU with an *interrupt* (i.e., asserts a special “interrupt” pin on the CPU).
- Requests to read and write logical blocks come across the I/O bus to the Flash translation layer.
- Pages are 512KB to 4KB; blocks are 32 to 128 pages.
- Data is read/written in units of pages.
- A page can only be written after its block has been erased.
- A block wears out after around 100,000 repeated writes.
SSDs Performance Characteristics

<table>
<thead>
<tr>
<th></th>
<th>Sequential read tput</th>
<th>250 MB/s</th>
<th>Sequential write tput</th>
<th>170 MB/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random read tput</td>
<td>140 MB/s</td>
<td></td>
<td>Random write tput</td>
<td>14 MB/s</td>
</tr>
<tr>
<td>Random read access</td>
<td>30 $\mu$s</td>
<td></td>
<td>Random write access</td>
<td>300 $\mu$s</td>
</tr>
</tbody>
</table>

Why are random writes so slow?

- Erasing a block is slow (around 1 ms).
- Write to a page triggers a copy of all useful pages in the block.
- Must find a used block (new block) and erase it.
- Write the page into the new block.
- Copy other pages from the old block to the new block.
Advantages:
- No moving parts; faster, less power, more rugged.

Disadvantages:
- Have the potential to wear out. This is mitigated by “wear leveling logic” in the flash translation layer.
- E.g., Intel X25 guarantees 1 petabyte ($10^{15}$ bytes) of random writes before they wear out.
- In 2010, they were about 100X more expensive. But by November, 2013 this has fallen to 10X. By February, 2015, this was about 2X.

Applications:
- MP3 players, smart phones, laptops.
- They are beginning to appear in desktops and servers.
## Storage Trends

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SRAM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$/MB access (ns)</td>
<td>19.2K</td>
<td>2.9K</td>
<td>320</td>
<td>256</td>
<td>100</td>
<td>75</td>
<td>60</td>
<td>320</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>150</td>
<td>35</td>
<td>15</td>
<td>3</td>
<td>2</td>
<td>1.5</td>
<td>200</td>
</tr>
<tr>
<td><strong>DRAM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$/MB access (ns)</td>
<td>8K</td>
<td>880</td>
<td>100</td>
<td>30</td>
<td>1</td>
<td>0.1</td>
<td>0.06</td>
<td>130K</td>
</tr>
<tr>
<td></td>
<td>375</td>
<td>200</td>
<td>100</td>
<td>70</td>
<td>60</td>
<td>50</td>
<td>40</td>
<td>9</td>
</tr>
<tr>
<td>typical size</td>
<td>0.064</td>
<td>0.256</td>
<td>4</td>
<td>16</td>
<td>64</td>
<td>2K</td>
<td>8K</td>
<td>125K</td>
</tr>
<tr>
<td><strong>Disk</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$/MB access (ms)</td>
<td>500</td>
<td>100</td>
<td>8</td>
<td>0.30</td>
<td>0.001</td>
<td>0.005</td>
<td>0.0003</td>
<td>1.6M</td>
</tr>
<tr>
<td></td>
<td>87</td>
<td>75</td>
<td>28</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>typical size</td>
<td>1</td>
<td>10</td>
<td>160</td>
<td>1K</td>
<td>20K</td>
<td>160K</td>
<td>1.5M</td>
<td>1.5M</td>
</tr>
</tbody>
</table>
### CPU Clock Rates

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>8080</td>
<td>386</td>
<td>Pentium</td>
<td>P-III</td>
<td>P-4</td>
<td>Core 2</td>
<td>Core i7</td>
<td></td>
</tr>
<tr>
<td>Clock MHz</td>
<td>1</td>
<td>20</td>
<td>150</td>
<td>600</td>
<td>3300</td>
<td>2000</td>
<td>2500</td>
<td>2500</td>
</tr>
<tr>
<td>Cycle (ns)</td>
<td>1000</td>
<td>50</td>
<td>6</td>
<td>1.6</td>
<td>0.3</td>
<td>0.5</td>
<td>0.4</td>
<td>2500</td>
</tr>
<tr>
<td>Cores</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Effective Cycle time</td>
<td>1000</td>
<td>50</td>
<td>6</td>
<td>1.6</td>
<td>0.3</td>
<td>0.25</td>
<td>0.1</td>
<td>10K</td>
</tr>
</tbody>
</table>

Around 2003, was the inflection point in computer history when designers hit the “Power Wall.” Cores increased, but the clock rate actually decreased.
CPU speed increases *faster* than memory speed, meaning that:
- memory is more and more a limiting factor on performance;
- increased importance for caching and similar techniques.