# Part V

## Memory System Design

<table>
<thead>
<tr>
<th>Parts</th>
<th>Chapters</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Background and Motivation</td>
<td>1. Combinational Digital Circuits</td>
</tr>
<tr>
<td></td>
<td>2. Digital Circuits with Memory</td>
</tr>
<tr>
<td></td>
<td>3. Computer System Technology</td>
</tr>
<tr>
<td></td>
<td>4. Computer Performance</td>
</tr>
<tr>
<td>II. Instruction-Set Architecture</td>
<td>5. Instructions and Addressing</td>
</tr>
<tr>
<td></td>
<td>6. Procedures and Data</td>
</tr>
<tr>
<td></td>
<td>7. Assembly Language Programs</td>
</tr>
<tr>
<td></td>
<td>8. Instruction-Set Variations</td>
</tr>
<tr>
<td>III. The Arithmetic/Logic Unit</td>
<td>9. Number Representation</td>
</tr>
<tr>
<td></td>
<td>10. Adders and Simple ALUs</td>
</tr>
<tr>
<td></td>
<td>11. Multipliers and Dividers</td>
</tr>
<tr>
<td></td>
<td>12. Floating-Point Arithmetic</td>
</tr>
<tr>
<td>IV. Data Path and Control</td>
<td>13. Instruction Execution Steps</td>
</tr>
<tr>
<td></td>
<td>14. Control Unit Synthesis</td>
</tr>
<tr>
<td></td>
<td>15. Pipelined Data Paths</td>
</tr>
<tr>
<td></td>
<td>16. Pipeline Performance Limits</td>
</tr>
<tr>
<td>V. Memory System Design</td>
<td>17. Main Memory Concepts</td>
</tr>
<tr>
<td></td>
<td>18. Cache Memory Organization</td>
</tr>
<tr>
<td></td>
<td>19. Mass Memory Concepts</td>
</tr>
<tr>
<td></td>
<td>20. Virtual Memory and Paging</td>
</tr>
<tr>
<td>VI. Input/Output and Interfacing</td>
<td>21. Input/Output Devices</td>
</tr>
<tr>
<td></td>
<td>22. Input/Output Programming</td>
</tr>
<tr>
<td></td>
<td>23. Buses, Links, and Interfacing</td>
</tr>
<tr>
<td></td>
<td>24. Context Switching and Interrupts</td>
</tr>
<tr>
<td>VII. Advanced Architectures</td>
<td>25. Road to Higher Performance</td>
</tr>
<tr>
<td></td>
<td>26. Vector and Array Processing</td>
</tr>
<tr>
<td></td>
<td>27. Shared-Memory Multiprocessing</td>
</tr>
<tr>
<td></td>
<td>28. Distributed Multicomputing</td>
</tr>
</tbody>
</table>
About This Presentation

This presentation is intended to support the use of the textbook *Computer Architecture: From Microprocessors to Supercomputers*, Oxford University Press, 2005, ISBN 0-19-515455-X. It is updated regularly by the author as part of his teaching of the upper-division course ECE 154, Introduction to Computer Architecture, at the University of California, Santa Barbara. Instructors can use these slides freely in classroom teaching and for other educational purposes. Any other use is strictly prohibited. © Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
</tr>
</thead>
</table>
V Memory System Design

Design problem – We want a memory unit that:
- Can keep up with the CPU’s processing speed
- Has enough capacity for programs and data
- Is inexpensive, reliable, and energy-efficient

<table>
<thead>
<tr>
<th>Topics in This Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 17    Main Memory Concepts</td>
</tr>
<tr>
<td>Chapter 18    Cache Memory Organization</td>
</tr>
<tr>
<td>Chapter 19    Mass Memory Concepts</td>
</tr>
<tr>
<td>Chapter 20    Virtual Memory and Paging</td>
</tr>
</tbody>
</table>
17 Main Memory Concepts

Technologies & organizations for computer’s main memory
• SRAM (cache), DRAM (main), and flash (nonvolatile)
• Interleaving & pipelining to get around “memory wall”

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.1 Memory Structure and SRAM</td>
</tr>
<tr>
<td>17.2 DRAM and Refresh Cycles</td>
</tr>
<tr>
<td>17.3 Hitting the Memory Wall</td>
</tr>
<tr>
<td>17.4 Interleaved and Pipelined Memory</td>
</tr>
<tr>
<td>17.5 Nonvolatile Memory</td>
</tr>
<tr>
<td>17.6 The Need for a Memory Hierarchy</td>
</tr>
</tbody>
</table>
17.1 Memory Structure and SRAM

Fig. 17.1 Conceptual inner structure of a $2^h \times g$ SRAM chip and its shorthand representation.
Fig. 17.2 Eight 128K × 8 SRAM chips forming a 256K × 32 memory unit.
SRAM with Bidirectional Data Bus

Fig. 17.3 When data input and output of an SRAM chip are shared or connected to a bidirectional data bus, output must be disabled during write operations.
17.2 DRAM and Refresh Cycles

DRAM vs. SRAM Memory Cell Complexity

(a) DRAM cell

(b) Typical SRAM cell

Fig. 17.4 Single-transistor DRAM cell, which is considerably simpler than SRAM cell, leads to dense, high-capacity DRAM memory chips.
Fig. 17.5 Variations in the voltage across a DRAM cell capacitor after writing a 1 and subsequent refresh operations.
Loss of Bandwidth to Refresh Cycles

Example 17.2

A 256 Mb DRAM chip is organized as a $32M \times 8$ memory externally and as a $16K \times 16K$ array internally. Rows must be refreshed at least once every 50 ms to forestall data loss; refreshing a row takes 100 ns. What fraction of the total memory bandwidth is lost to refresh cycles?

Solution

Refreshing all 16K rows takes $16 \times 1024 \times 100 \text{ ns} = 1.64 \text{ ms}$. Loss of 1.64 ms every 50 ms amounts to $1.64/50 = 3.3\%$ of the total bandwidth.
DRAM Packaging

24-pin dual in-line package (DIP)

Fig. 17.6 Typical DRAM package housing a $16M \times 4$ memory.
Fig. 17.7
Trends in DRAM main memory.
17.3 Hitting the Memory Wall

Fig. 17.8  Memory density and capacity have grown along with the CPU power and complexity, but memory speed has not kept pace.
Bridging the CPU-Memory Speed Gap

Idea: Retrieve more data from memory with each access

Fig. 17.9 Two ways of using a wide-access memory to bridge the speed gap between the processor and memory.
17.4 Pipelined and Interleaved Memory

Memory latency may involve other supporting operations besides the physical access itself:

- Virtual-to-physical address translation (Chap 20)
- Tag comparison to determine cache hit/miss (Chap 18)

**Fig. 17.10** Pipelined cache memory.
Memory Interleaving

Fig. 17.11 Interleaved memory is more flexible than wide-access memory in that it can handle multiple independent accesses at once.
17.5 Nonvolatile Memory

Fig. 17.12  Read-only memory organization, with the fixed contents shown on the right.
Flash Memory

Fig. 17.13   EEPROM or Flash memory organization. Each memory cell is built of a floating-gate MOS transistor.
17.6 The Need for a Memory Hierarchy

The widening speed gap between CPU and main memory

- Processor operations take of the order of 1 ns
- Memory access requires 10s or even 100s of ns

Memory bandwidth limits the instruction execution rate

- Each instruction executed involves at least one memory access
- Hence, a few to 100s of MIPS is the best that can be achieved
- A fast buffer memory can help bridge the CPU-memory gap
- The fastest memories are expensive and thus not very large
- A second (third?) intermediate cache level is thus often used
Typical Levels in a Hierarchical Memory

<table>
<thead>
<tr>
<th>Capacity</th>
<th>Access latency</th>
<th>Cost per GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>100s B</td>
<td>ns</td>
<td>$Millions</td>
</tr>
<tr>
<td>10s KB</td>
<td>a few ns</td>
<td>$100s Ks</td>
</tr>
<tr>
<td>MBs</td>
<td>10s ns</td>
<td>$10s Ks</td>
</tr>
<tr>
<td>100s MB</td>
<td>100s ns</td>
<td>$1000s</td>
</tr>
<tr>
<td>10s GB</td>
<td>10s ms</td>
<td>$10s</td>
</tr>
<tr>
<td>TBs</td>
<td>min+</td>
<td>$1s</td>
</tr>
</tbody>
</table>

Fig. 17.14 Names and key characteristics of levels in a memory hierarchy.
Memory Price Trends

Source: https://www1.hitachigst.com/hdd/technolo/overview/chart03.html
18 Cache Memory Organization

Processor speed is improving at a faster rate than memory's
- Processor-memory speed gap has been widening
- Cache is to main as desk drawer is to file cabinet

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.1 The Need for a Cache</td>
</tr>
<tr>
<td>18.2 What Makes a Cache Work?</td>
</tr>
<tr>
<td>18.3 Direct-Mapped Cache</td>
</tr>
<tr>
<td>18.4 Set-Associative Cache</td>
</tr>
<tr>
<td>18.5 Cache and Main Memory</td>
</tr>
<tr>
<td>18.6 Improving Cache Performance</td>
</tr>
</tbody>
</table>
18.1 The Need for a Cache

All three of our MicroMIPS designs assumed 2-ns data and instruction memories; however, typical RAMs are 10-50 times slower.
Cache, Hit/Miss Rate, and Effective Access Time

Cache is transparent to user; transfers occur automatically.

One level of cache with hit rate $h$

$$C_{\text{eff}} = hC_{\text{fast}} + (1 - h)(C_{\text{slow}} + C_{\text{fast}}) = C_{\text{fast}} + (1 - h)C_{\text{slow}}$$
Multiple Cache Levels

Fig. 18.1 Cache memories act as intermediaries between the superfast processor and the much slower main memory.
## Performance of a Two-Level Cache System

### Example 18.1

A system with L1 and L2 caches has a CPI of 1.2 with no cache miss. There are 1.1 memory accesses on average per instruction.

What is the effective CPI with cache misses factored in?

What are the effective hit rate and miss penalty overall if L1 and L2 caches are modeled as a single cache?

<table>
<thead>
<tr>
<th>Level</th>
<th>Local hit rate</th>
<th>Miss penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>95 %</td>
<td>8 cycles</td>
</tr>
<tr>
<td>L2</td>
<td>80 %</td>
<td>60 cycles</td>
</tr>
</tbody>
</table>

### Solution

\[
C_{\text{eff}} = C_{\text{fast}} + (1 - h_1)[C_{\text{medium}} + (1 - h_2)C_{\text{slow}}]
\]

Because \( C_{\text{fast}} \) is included in the CPI of 1.2, we must account for the rest

\[
\text{CPI} = 1.2 + 1.1(1 - 0.95)[8 + (1 - 0.8)60] = 1.2 + 1.1 \times 0.05 \times 20 = 2.3
\]

Overall: hit rate 99% (95% + 80% of 5%), miss penalty 60 cycles
Cache Memory Design Parameters

*Cache size* (in bytes or words). A larger cache can hold more of the program’s useful data but is more costly and likely to be slower.

*Block or cache-line size* (unit of data transfer between cache and main). With a larger cache line, more data is brought in cache with each miss. This can improve the hit rate but also may bring low-utility data in.

*Placement policy*. Determining where an incoming cache line is stored. More flexible policies imply higher hardware cost and may or may not have performance benefits (due to more complex data location).

*Replacement policy*. Determining which of several existing cache blocks (into which a new cache line can be mapped) should be overwritten. Typical policies: choosing a random or the least recently used block.

*Write policy*. Determining if updates to cache words are immediately forwarded to main (*write-through*) or modified blocks are copied back to main if and when they must be replaced (*write-back* or *copy-back*).
18.2 What Makes a Cache Work?

Assuming no conflict in address mapping, the cache will hold a small program loop in its entirety, leading to fast execution.

Fig. 18.2  Assuming no conflict in address mapping, the cache will hold a small program loop in its entirety, leading to fast execution.
Desktop, Drawer, and File Cabinet Analogy

Once the “working set” is in the drawer, very few trips to the file cabinet are needed.

Fig. 18.3 Items on a desktop (register) or in a drawer (cache) are more readily accessible than those in a file cabinet (main memory).
Temporal and Spatial Localities

**Temporal:**
Accesses to the same address are typically clustered in time

**Spatial:**
When a location is accessed, nearby locations tend to be accessed also

From Peter Denning’s CACM paper, July 2005 (Vol. 48, No. 7, pp. 19-24)
Caching Benefits Related to Amdahl’s Law

Example 18.2

In the drawer & file cabinet analogy, assume a hit rate $h$ in the drawer. Formulate the situation shown in Fig. 18.2 in terms of Amdahl’s law.

Solution

Without the drawer, a document is accessed in 30 s. So, fetching 1000 documents, say, would take 30 000 s. The drawer causes a fraction $h$ of the cases to be done 6 times as fast, with access time unchanged for the remaining $1 - h$. Speedup is thus $1/(1 - h + h/6) = 6 / (6 - 5h)$. Improving the drawer access time can increase the speedup factor but as long as the miss rate remains at $1 - h$, the speedup can never exceed $1 / (1 - h)$. Given $h = 0.9$, for instance, the speedup is 4, with the upper bound being 10 for an extremely short drawer access time.

Note: Some would place everything on their desktop, thinking that this yields even greater speedup. This strategy is not recommended!
Compulsory, Capacity, and Conflict Misses

**Compulsory misses:** With *on-demand fetching*, first access to any item is a miss. Some “compulsory” misses can be avoided by prefetching.

**Capacity misses:** We have to oust some items to make room for others. This leads to misses that are not incurred with an infinitely large cache.

**Conflict misses:** Occasionally, there is free room, or space occupied by useless data, but the mapping/placement scheme forces us to displace useful items to bring in other items. This may lead to misses in future.

Given a fixed-size cache, dictated, e.g., by cost factors or availability of space on the processor chip, compulsory and capacity misses are pretty much fixed. Conflict misses, on the other hand, are influenced by the data mapping scheme which is under our control.

We study two popular mapping schemes: direct and set-associative.
18.3 Direct-Mapped Cache

Fig. 18.4  Direct-mapped cache holding 32 words within eight 4-word lines. Each line is associated with a tag and a valid bit.
Accessing a Direct-Mapped Cache

Example 18.4

Show cache addressing for a byte-addressable memory with 32-bit addresses. Cache line $W = 16$ B. Cache size $L = 4096$ lines (64 KB).

Solution

Byte offset in line is $\log_2 16 = 4$ b. Cache line index is $\log_2 4096 = 12$ b. This leaves $32 - 12 - 4 = 16$ b for the tag.

Fig. 18.5 Components of the 32-bit address in an example direct-mapped cache with byte addressing.
Direct-Mapped Cache Behavior

Address trace:
1, 7, 6, 5, 32, 33, 1, 2, ... 

1: miss, line 3, 2, 1, 0 fetched
7: miss, line 7, 6, 5, 4 fetched
6: hit
5: hit
32: miss, line 35, 34, 33, 32 fetched (replaces 3, 2, 1, 0)
33: hit
  1: miss, line 3, 2, 1, 0 fetched (replaces 35, 34, 33, 32)
  2: hit
... and so on
18.4 Set-Associative Cache

Fig. 18.6 Two-way set-associative cache holding 32 words of data within 4-word lines and 2-line sets.
Accessing a Set-Associative Cache

Example 18.5

Show cache addressing scheme for a byte-addressable memory with 32-bit addresses. Cache line width $2^W = 16$ B. Set size $2^S = 2$ lines. Cache size $2^L = 4096$ lines (64 KB).

Solution

Byte offset in line is $\log_2 16 = 4$ b. Cache set index is $(\log_2 4096/2) = 11$ b. This leaves $32 - 11 - 4 = 17$ b for the tag.

Fig. 18.7   Components of the 32-bit address in an example two-way set-associative cache.

11-bit set index in cache
17-bit line tag
4-bit byte offset in line

Address in cache used to read out two candidate items and their control info
Cache Address Mapping

Example 18.6

A 64 KB four-way set-associative cache is byte-addressable and contains 32 B lines. Memory addresses are 32 b wide.

a. How wide are the tags in this cache?
b. Which main memory addresses are mapped to set number 5?

Solution

a. Address (32 b) = 5 b byte offset + 9 b set index + 18 b tag

b. Addresses that have their 9-bit set index equal to 5. These are of the general form $2^{14}a + 2^5 \times 5 + b$; e.g., 160-191, 16 554-16 575, . . .

<table>
<thead>
<tr>
<th>32-bit address</th>
<th>Tag width = 18</th>
<th>Set size = 4 \times 32 B = 128 B</th>
<th>Line width = 32 B = 2^5 B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>18 bits</td>
<td>Set index</td>
<td>Offset</td>
</tr>
<tr>
<td>Offset</td>
<td>9 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>5 bits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
18.5 Cache and Main Memory

Split cache: separate instruction and data caches (L1)
Unified cache: holds instructions and data (L1, L2, L3)

Harvard architecture: separate instruction and data memories
von Neumann architecture: one memory for instructions and data

The writing problem:
Write-through slows down the cache to allow main to catch up
Write-back or copy-back is less problematic, but still hurts performance due to two main memory accesses in some cases.

Solution: Provide write buffers for the cache so that it does not have to wait for main memory to catch up.
Faster Main-Cache Data Transfers

Fig. 18.8 A 256 Mb DRAM chip organized as a $32M \times 8$ memory module: four such chips could form a 128 MB main memory unit.
18.6 Improving Cache Performance

For a given *cache size*, the following design issues and tradeoffs exist:

*Line width* \((2^W)\). Too small a value for \(W\) causes a lot of main memory accesses; too large a value increases the miss penalty and may tie up cache space with low-utility items that are replaced before being used.

*Set size or associativity* \((2^S)\). Direct mapping \((S = 0)\) is simple and fast; greater associativity leads to more complexity, and thus slower access, but tends to reduce conflict misses. More on this later.

*Line replacement policy*. Usually LRU (least recently used) algorithm or some approximation thereof; not an issue for direct-mapped caches. Somewhat surprisingly, random selection works quite well in practice.

*Write policy*. Modern caches are very fast, so that write-through is seldom a good choice. We usually implement write-back or copy-back, using write buffers to soften the impact of main memory latency.
Effect of Line Width on Cache Performance

Variation of cache performance as a function of line size.
Effect of Associativity on Cache Performance

Fig. 18.9 Performance improvement of caches with increased associativity.
19 Mass Memory Concepts

Today’s main memory is huge, but still inadequate for all needs
• Magnetic disks provide extended and back-up storage
• Optical disks & disk arrays are other mass storage options

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.1  Disk Memory Basics</td>
</tr>
<tr>
<td>19.2  Organizing Data on Disk</td>
</tr>
<tr>
<td>19.3  Disk Performance</td>
</tr>
<tr>
<td>19.4  Disk Caching</td>
</tr>
<tr>
<td>19.5  Disk Arrays and RAID</td>
</tr>
<tr>
<td>19.6  Other Types of Mass Memory</td>
</tr>
</tbody>
</table>
19.1 Disk Memory Basics

Fig. 19.1 Disk memory elements and key terms.
Disk Drives

Typically 2-8 cm
Access Time for a Disk

1. Head movement from current position to desired cylinder: **Seek time** (0-10s ms)

2. Disk rotation until the desired sector arrives under the head: **Rotational latency** (0-10s ms)

3. Disk rotation until sector has passed under the head: **Data transfer time** (< 1 ms)

The three components of disk access time. Disks that spin faster have a shorter average and worst-case access time.
Table 19.1 Key attributes of three representative magnetic disks, from the highest capacity to the smallest physical size (ca. early 2003). [More detail (weight, dimensions, recording density, etc.) in textbook.]

<table>
<thead>
<tr>
<th>Manufacturer and Model Name</th>
<th>Seagate Barracuda 180</th>
<th>Hitachi DK23DA</th>
<th>IBM Microdrive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application domain</td>
<td>Server</td>
<td>Laptop</td>
<td>Pocket device</td>
</tr>
<tr>
<td>Capacity</td>
<td>180 GB</td>
<td>40 GB</td>
<td>1 GB</td>
</tr>
<tr>
<td>Platters / Surfaces</td>
<td>12 / 24</td>
<td>2 / 4</td>
<td>1 / 2</td>
</tr>
<tr>
<td>Cylinders</td>
<td>24 247</td>
<td>33 067</td>
<td>7 167</td>
</tr>
<tr>
<td>Sectors per track, avg</td>
<td>604</td>
<td>591</td>
<td>140</td>
</tr>
<tr>
<td>Buffer size</td>
<td>16 MB</td>
<td>2 MB</td>
<td>1/8 MB</td>
</tr>
<tr>
<td>Seek time, min, avg, max</td>
<td>1, 8, 17 ms</td>
<td>3, 13, 25 ms</td>
<td>1, 12, 19 ms</td>
</tr>
<tr>
<td>Diameter</td>
<td>3.5”</td>
<td>2.5”</td>
<td>1.0”</td>
</tr>
<tr>
<td>Rotation speed, rpm</td>
<td>7 200</td>
<td>4 200</td>
<td>3 600</td>
</tr>
<tr>
<td>Typical power</td>
<td>14.1 W</td>
<td>2.3 W</td>
<td>0.8 W</td>
</tr>
</tbody>
</table>
19.2 Organizing Data on Disk

Fig. 19.2  Magnetic recording along the tracks and the read/write head.

<table>
<thead>
<tr>
<th>0</th>
<th>16</th>
<th>32</th>
<th>48</th>
<th>1</th>
<th>17</th>
<th>33</th>
<th>49</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>46</td>
<td>62</td>
<td>15</td>
<td>31</td>
<td>47</td>
<td>0</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>60</td>
<td>13</td>
<td>29</td>
<td>45</td>
<td>61</td>
<td>14</td>
<td>30</td>
<td>46</td>
<td>62</td>
</tr>
<tr>
<td>27</td>
<td>43</td>
<td>59</td>
<td>12</td>
<td>28</td>
<td>44</td>
<td>60</td>
<td>13</td>
<td>29</td>
</tr>
</tbody>
</table>

Track \( i \)
Track \( i + 1 \)
Track \( i + 2 \)
Track \( i + 3 \)

Fig. 19.3  Logical numbering of sectors on several adjacent tracks.
19.3 Disk Performance

Seek time = $a + b(c - 1) + \beta(c - 1)^{1/2}$

Average rotational latency = $(30 / \text{rpm}) \; s = (30 \, 000 / \text{rpm}) \; \text{ms}$

Fig. 19.4 Reducing average seek time and rotational latency by performing disk accesses out of order.
19.4 Disk Caching

Same idea as processor cache: bridge main-disk speed gap

Read/write an entire track with each disk access:
   “Access one sector, get 100s free,” hit rate around 90%
Disks listed in Table 19.1 have buffers from 1/8 to 16 MB
Rotational latency eliminated; can start from any sector
Need back-up power so as not to lose changes in disk cache
   (need it anyway for head retraction upon power loss)

Placement options for disk cache

In the disk controller:
   Suffers from bus and controller latencies even for a cache hit

Closer to the CPU:
   Avoids latencies and allows for better utilization of space

Intermediate or multilevel solutions
19.5 Disk Arrays and RAID

The need for high-capacity, high-throughput secondary (disk) memory

<table>
<thead>
<tr>
<th>Processor speed</th>
<th>RAM size</th>
<th>Disk I/O rate</th>
<th>Number of disks</th>
<th>Disk capacity</th>
<th>Number of disks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GIPS</td>
<td>1 GB</td>
<td>100 MB/s</td>
<td>1</td>
<td>100 GB</td>
<td>1</td>
</tr>
<tr>
<td>1 TIPS</td>
<td>1 TB</td>
<td>100 GB/s</td>
<td>1000</td>
<td>100 TB</td>
<td>100</td>
</tr>
<tr>
<td>1 PIPS</td>
<td>1 PB</td>
<td>100 TB/s</td>
<td>1 Million</td>
<td>100 PB</td>
<td>100 000</td>
</tr>
<tr>
<td>1 EIPS</td>
<td>1 EB</td>
<td>100 PB/s</td>
<td>1 Billion</td>
<td>100 EB</td>
<td>100 Million</td>
</tr>
</tbody>
</table>

Amdahl’s rules of thumb for system balance

1 RAM byte for each IPS
1 I/O bit per sec for each IPS
100 disk bytes for each RAM byte
Redundant Array of Independent Disks (RAID)

**RAID0**: Multiple disks for higher data rate; no redundancy

**RAID1**: Mirrored disks

**RAID2**: Error-correcting code

**RAID3**: Bit- or byte-level striping with parity/checksum disk

\[ A \oplus B \oplus C \oplus D \oplus P = 0 \rightarrow B = A \oplus C \oplus D \oplus P \]

**RAID4**: Parity/checksum applied to sectors, not bits or bytes

**RAID5**: Parity/checksum distributed across several disks

**RAID6**: Parity and 2nd check distributed across several disks

**Fig. 19.5** RAID levels 0-6, with a simplified view of data organization.
RAID Product Examples

IBM ESS Model 750
19.6 Other Types of Mass Memory

Fig. 3.12 Magnetic and optical disk memory units.

(a) Cutaway view of a hard disk drive

(b) Some removable storage media

Typically 2-9 cm

Floppy disk

CD-ROM

Magnetic tape cartridge

Flash drive
Thumb drive
Travel drive
Fig. 19.6  Simplified view of recording format and access mechanism for data on a CD-ROM or DVD-ROM.
Automated Tape Libraries
Managing data transfers between main & mass is cumbersome
• Virtual memory automates this process
• Key to virtual memory’s success is the same as for cache

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.1 The Need for Virtual Memory</td>
</tr>
<tr>
<td>20.2 Address Translation in Virtual Memory</td>
</tr>
<tr>
<td>20.3 Translation Lookaside Buffer</td>
</tr>
<tr>
<td>20.4 Page Placement and Replacement</td>
</tr>
<tr>
<td>20.5 Main and Mass Memories</td>
</tr>
<tr>
<td>20.6 Improving Virtual Memory Performance</td>
</tr>
</tbody>
</table>
20.1 The Need for Virtual Memory

Fig. 20.1 Program segments in main memory and on disk.
Memory Hierarchy: The Big Picture

Fig. 20.2  Data movement in a memory hierarchy.

(transfered explicitly via load/store)

(transfered automatically upon cache miss)

(transfered automatically upon page fault)

Virtual memory
20.2 Address Translation in Virtual Memory

Virtual address

Virtual page number $V - P$ bits

Offset in page $P$ bits

Address translation

Physical address

Physical page number $M - P$ bits

Offset in page $P$ bits

Example 20.1

Determine the parameters in Fig. 20.3 for 32-bit virtual addresses, 4 KB pages, and 128 MB byte-addressable main memory.

**Solution:** Physical addresses are 27 b, byte offset in page is 12 b; thus, virtual (physical) page numbers are $32 - 12 = 20$ b (15 b)
Fig. 20.4 The role of page table in the virtual-to-physical address translation process.
Protection and Sharing in Virtual Memory

Fig. 20.5 Virtual memory as a facilitator of sharing and memory protection.
The Latency Penalty of Virtual Memory

Fig. 20.4
20.3 Translation Lookaside Buffer

Virtual page number → Byte offset

Virtual address

Valid bits

Tags match and entry is valid

Physical page number

Other flags

Physical address tag

Cache index

Byte offset in word

Program page in virtual memory

lw  $t0,0($s1)
addi $t1,$zero,0
L: add  $t1,$t1,1
beq $t1,$s2,D
add  $t2,$t1,$t1
add  $t2,$t2,$t2
add  $t2,$t2,$s1
lw  $t3,0($t2)
slt $t4,$t0,$t3
beq $t4,$zero,L
addi $t0,$t3,0
j  L
D: ...

All instructions on this page have the same virtual page address and thus entail the same translation.

Fig. 20.6 Virtual-to-physical address translation by a TLB and how the resulting physical address is used to access the cache memory.
Example 20.2

An address translation process converts a 32-bit virtual address to a 32-bit physical address. Memory is byte-addressable with 4 KB pages. A 16-entry, direct-mapped TLB is used. Specify the components of the virtual and physical addresses and the width of the various TLB fields.

Solution

Virtual Page number

16

Tag

16-entry TLB

Valid bits

Tags match and entry is valid

20

Virtual page number

Byte offset

Physical page number

20

Other flags

Physical address tag

Cache index

Physical address

Translation

Valid

bits

Other

flags

Physical

address tag

12

Cache index

Byte offset in word

Physical address

16-bit tag + 20-bit phys page # + 1 valid bit + Other flags ≥ 37 bits

TLB word width

16-entry TLB

Fig. 20.6
Virtual- or Physical-Address Cache?

Fig. 20.7 Options for where virtual-to-physical address translation occurs.

TLB access may form an extra pipeline stage, thus the penalty in throughput can be insignificant.

Cache may be accessed with part of address that is common between virtual and physical addresses.
20.4 Page Replacement Policies

Least-recently used (LRU) policy

Implemented by maintaining a stack

Pages → A B A F B E A

LRU stack

<table>
<thead>
<tr>
<th>MRU</th>
<th>D A B A F B E A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B D A B A F B E</td>
</tr>
<tr>
<td></td>
<td>E B D D B A F B</td>
</tr>
<tr>
<td>LRU</td>
<td>C E E E D D A F</td>
</tr>
</tbody>
</table>
Approximate LRU Replacement Policy

Least-recently used policy: effective, but hard to implement

Approximate versions of LRU are more easily implemented

Clock policy: diagram below shows the reason for name
Use bit is set to 1 whenever a page is accessed

(a) Before replacement
(b) After replacement

Fig. 20.8  A scheme for the approximate implementation of LRU.
LRU Is Not Always the Best Policy

Example 20.2

Computing column averages for a 17 × 1024 table; 16-page memory

for j = [0 ... 1023] {
    temp = 0;
    for i = [0 ... 16]
        temp = temp + T[i][j]
    print(temp/17.0); }

Evaluate the page faults for row-major and column-major storage.

Solution

Fig. 20.9 Pagination of a 17×1024 table with row- or column-major storage.
20.5 Main and Mass Memories

Working set of a process, $W(t, x)$: The set of pages accessed over the last $x$ instructions at time $t$

Principle of locality ensures that the working set changes slowly

Fig. 20.10 Variations in the size of a program’s working set.
# 20.6 Improving Virtual Memory Performance

<table>
<thead>
<tr>
<th>Parameter variation</th>
<th>Potential advantages</th>
<th>Possible disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larger main or cache size</td>
<td>Fewer capacity misses</td>
<td>Longer access time</td>
</tr>
<tr>
<td>Larger pages or longer lines</td>
<td>Fewer compulsory misses</td>
<td>Greater miss penalty</td>
</tr>
<tr>
<td>Greater associativity (for cache only)</td>
<td>Fewer conflict misses</td>
<td>Longer access time</td>
</tr>
<tr>
<td>More sophisticated replacement policy</td>
<td>Fewer conflict misses</td>
<td>Longer decision time, more hardware</td>
</tr>
<tr>
<td>Write-through policy (for cache only)</td>
<td>No write-back time penalty, easier write-miss handling</td>
<td>Wasted memory bandwidth, longer access time</td>
</tr>
</tbody>
</table>
Impact of Technology on Virtual Memory

Fig. 20.11  Trends in disk, main memory, and CPU speeds.
Performance Impact of the Replacement Policy

Fig. 20.12  Dependence of page faults on the number of pages allocated and the page replacement policy
Summary of Memory Hierarchy

**Cache memory:** provides illusion of very high speed

**Main memory:** reasonable cost, but slow & small

**Virtual memory:** provides illusion of very large size

Locality makes the illusions work

**Fig. 20.2** Data movement in a memory hierarchy.