Cache
Memory
Cache Memory

* Why? Temporal Locality (Access Again)

CPU ➔ Memory

< 0.5 ns
Cycle time
2 GHz

Hundreds of Cycles

* How?

CPU

* Granularity

- Line Size (Block Size)
  (Spatial Locality)
The Abstraction
(Physically Addressed)

Virtual Addr \[
\rightarrow \]\[\]

TLB

1. To get the PFN

Physical Addr

Tag

2. To see if it is there

Store

Yes! A Hit

Data

3. To get it

Store

Three sequential accesses to storage

Can we do better?
FIRST EXAMPLE:
MEMORY: 256 B
CACHES: 64 B
LINE SIZE: 8 B
DIRECT MAPPED
Further Examples (Increases Associativity) Cache 4

Memory: 1256 B  
Cache: 64 B  
Line Size: 8 B  
4-Way Set Assoc

Why?

Tag Store
Cache A

Cache D
Data

Index

To Cache Line

Hit/Miss

Logic

0 1

01000

11000

00000

PA

TAG

Tag Store Entry: (Bookkeeping Information) LRU

Do You Need It Always?

Do You Need It Always?
The Access

Offset = (Byte within Block)

Cache A

Cache A

Cache D

Set

Set

Combinational Logic

Block

Note: Tag store and data store accessed at the same time.
(No longer 3 sequential accesses)
Can we do it in one access?

Virtual/Physical

VA  TAG  INDEX

Tag Store ...

Logic

Data Store ...

Cold Start

Plus: Don't wait for translation

Minuses: Same VA refers to Two Things
One PA Stored in Two Places

How do we handle this

1. Flush on Context Switch (Cold Start)
2. Include Process ID in Tag Store
Can We Do It In One Access?

Virtual/Physical

VA PAGE NO INDEX

TLB

PA FRAME NO INDEX

Use Frame number for Tag

Use Unmapped Bits for Access Index

Tag Store

Data Store

Logic

... = ...

Plus: Don't wait for translation

Minus: Limits size of cache

(although you can increase associativity)
Can we do it in one access time?

Virtually Indexed / Physically Tagged

\[ \text{VA} \quad \text{PAGE NUMBER} \quad \text{PAGE OFFSET} \]

INDEX

TLB

TAG STORE

DATA STORE

PFN \( \ldots \) PA TAGS COMPARE

\[ \text{PLUS: DON'T WAIT FOR TRANSLATION} \]

\[ \text{MINUS: SYNONYM PROBLEM} \]

(Block can be in more than one place)
Characteristics

* Set Associative (Set Size)
  - Fully Associative
  - Direct Mapped
  - \( \text{Hit Ratio} = \frac{\text{HITS}}{\text{HITS} + \text{MISSES}} \)

* Write Back, Write Through

* Replacement Algorithm
  - LRU
  - FIFO
  - Random

* Instructions/Data

* Supervisor/User

* Virtual/Physical
Write Through/Write Back

Issues

* Simplicity of Design
* Bus Traffic
* Application Environment (Stack Frame)
* Allocate on Write Miss
  - Sector Cache
REPLACEMENT

* Two-Way (LRU)

One Bit To Tell You Which Way Is LRU

* Four-Way (Pseudo-LRU)

1. Way | Way | Way | Way | Set
   A   B   C   D

   Three Bits/Set: A, B, C, D

2. Victim, Next Victim

<table>
<thead>
<tr>
<th>Way</th>
<th>V</th>
<th>NV</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

B is Victim

D is Next Victim

We flip a coin to determine A or C is NV
**This Microprocessors**

- **First Cache - Motorola 68020**
  - Mid-1980s, One Level, 256 Bytes I-Cache

- **Today - Several Levels (Caching Hierarchy)**

- Separate L1 Inst / L1 Data
  - Inclusion Property (Important for Coherence)

- Prefetch
  - Make Cache Visible (Prefetch Instruction)
  - Hardware - (Not Visible)
COHERENCE

* When we study multi-processors
  - Cache Coherences (hardware)
  - Memory Consistency (software)

* Even in a uniprocessor
  (if) we have intelligent controllers

Diagram:

- Processor
- Cache
- Memory
- Intelligent I/O Controller
- DMA

For example