



Computer Hardware Engineering (IS1200)

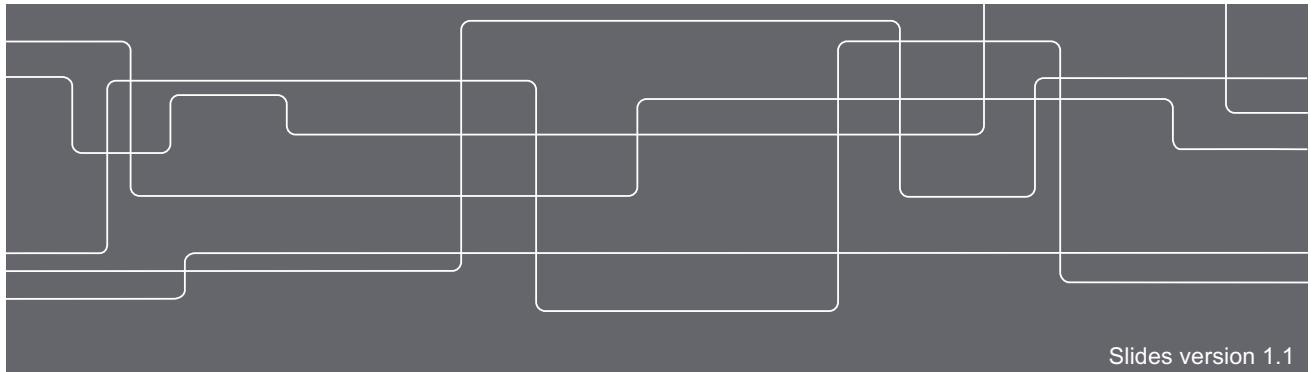
Computer Organization and Components (IS1500)

Spring 2019

Lecture 11: Memory Hierarchy

David Broman

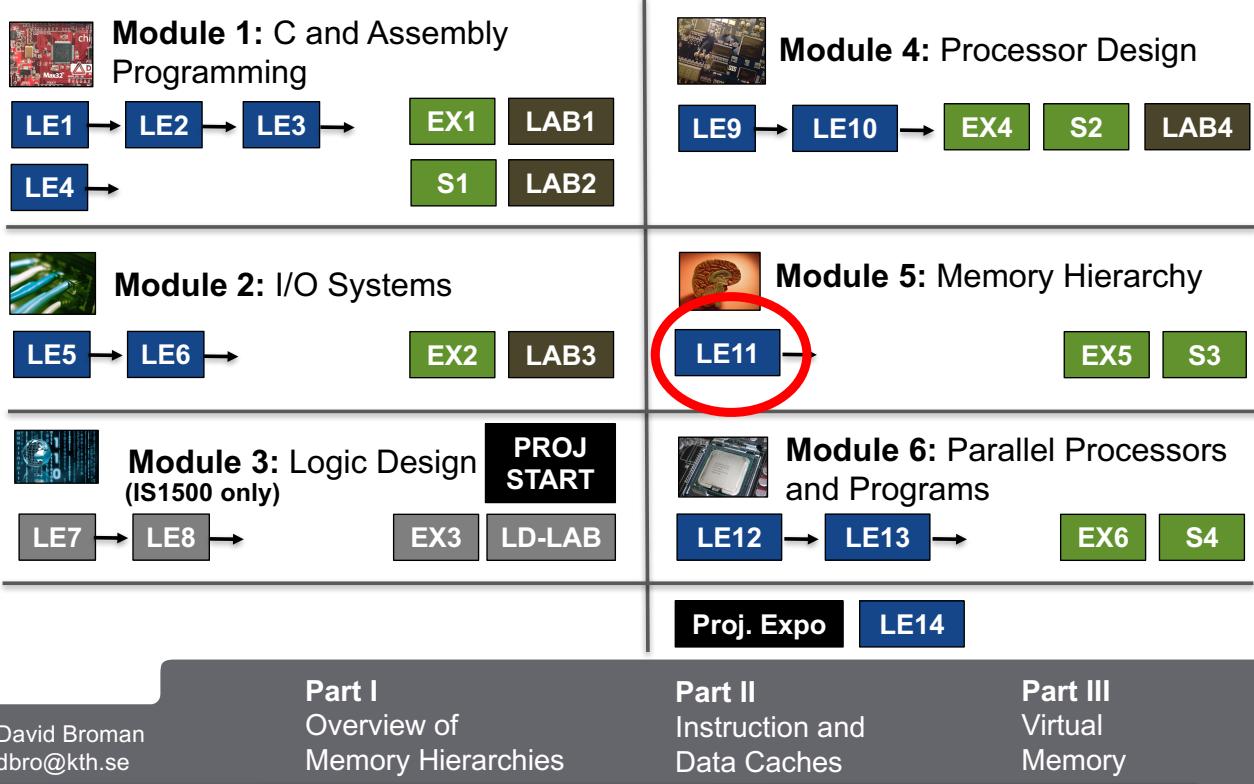
Associate Professor, KTH Royal Institute of Technology



2

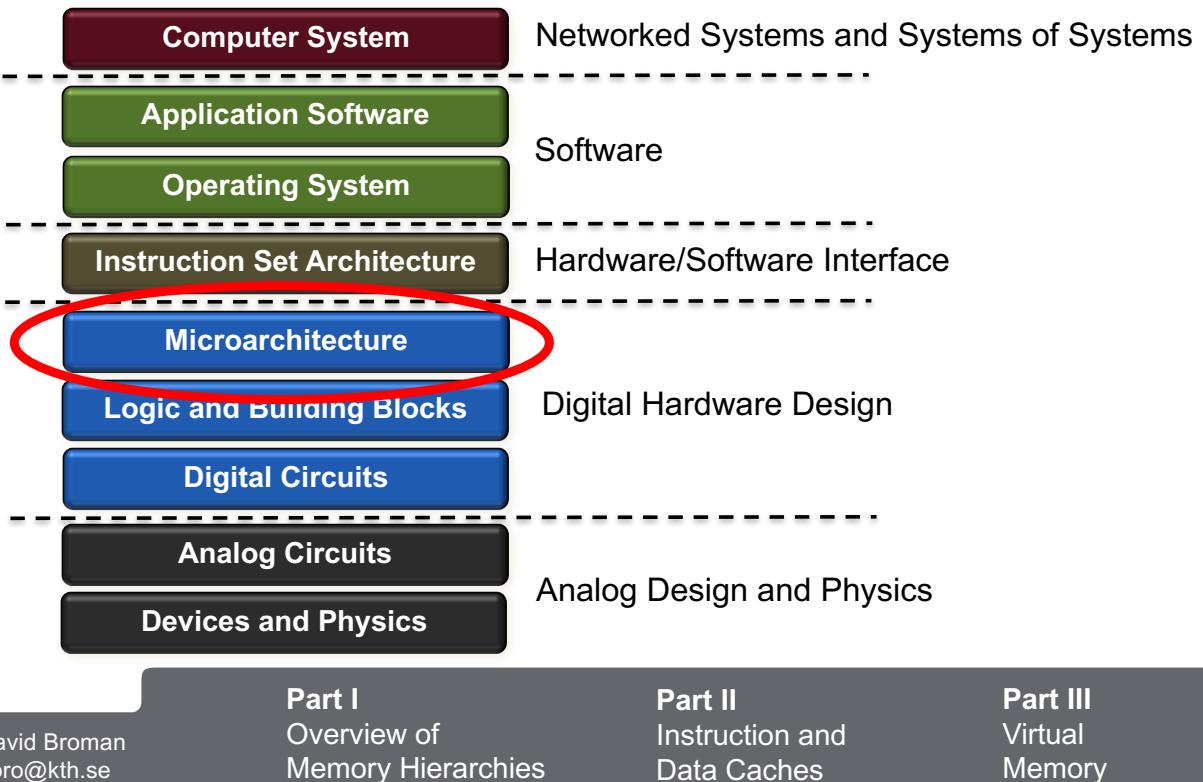


Course Structure

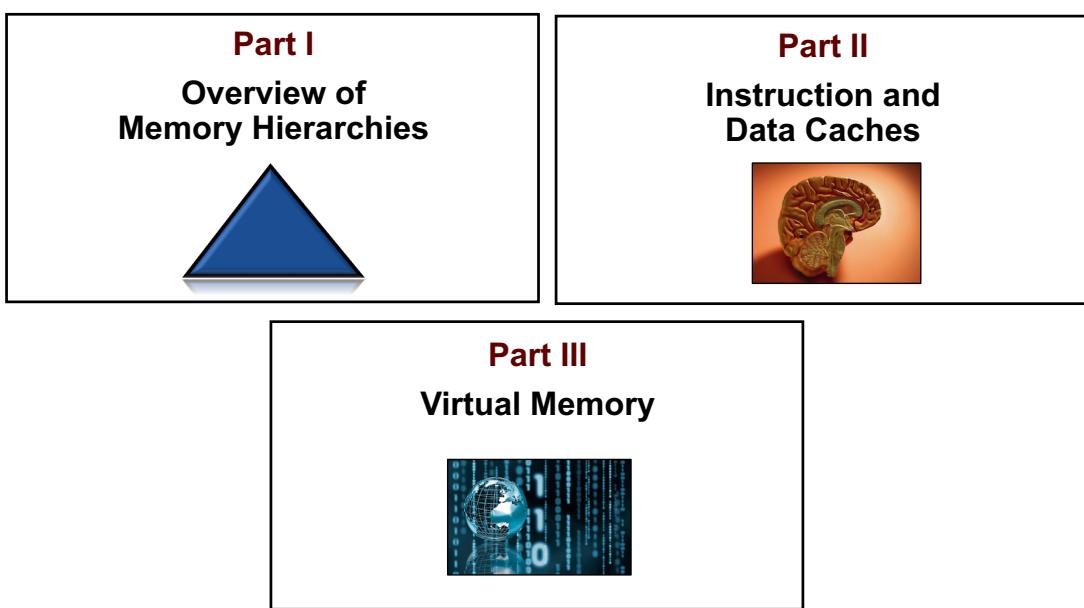




Abstractions in Computer Systems



Agenda

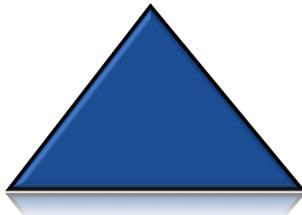


David Broman dbro@kth.se	Part I Overview of Memory Hierarchies	Part II Instruction and Data Caches	Part III Virtual Memory
-----------------------------	--	--	--------------------------------------



Part I

Overview of Memory Hierarchies



Acknowledgement: The structure and several of the good examples are derived from the book "Digital Design and Computer Architecture" (2013) by D. M. Harris and S. L. Harris.

David Broman
dbro@kth.se



Part I
Overview of
Memory Hierarchies

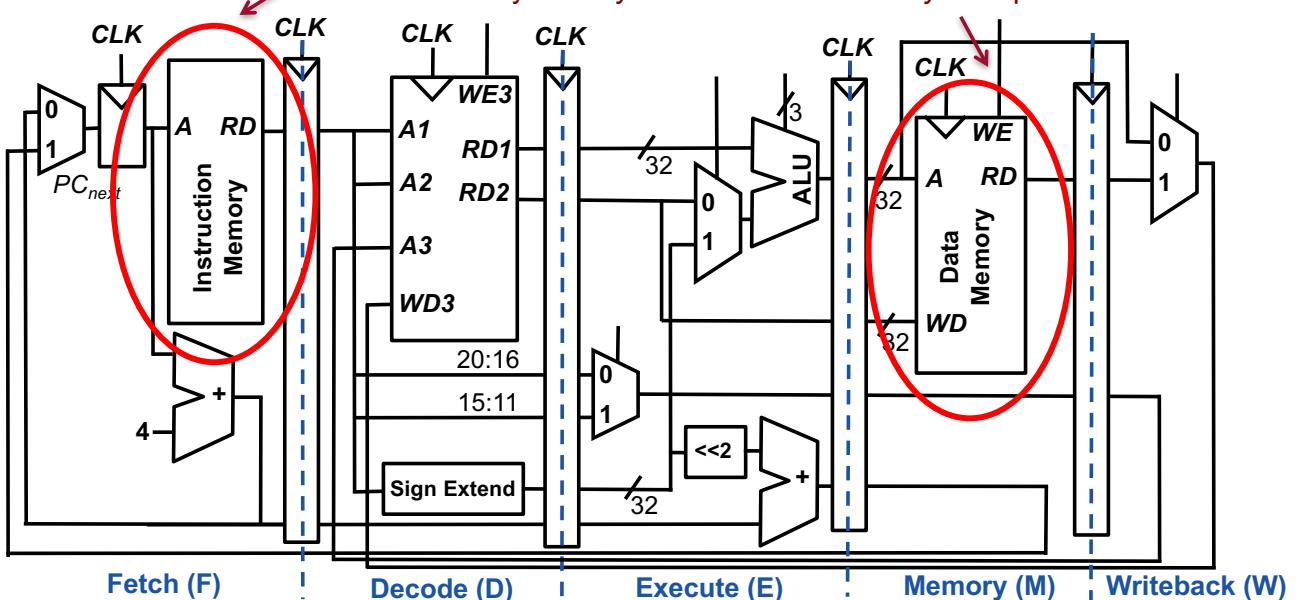
Part II
Instruction and
Data Caches

Part III
Virtual
Memory



Data Path (Revisited) Instruction and Data Memory

Problem: We assumed that each memory access takes 1 clock cycle.
This is *true* only for very small memories or very slow processors.



David Broman
dbro@kth.se



Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory

Temporal and Spatial Locality - The Library Example

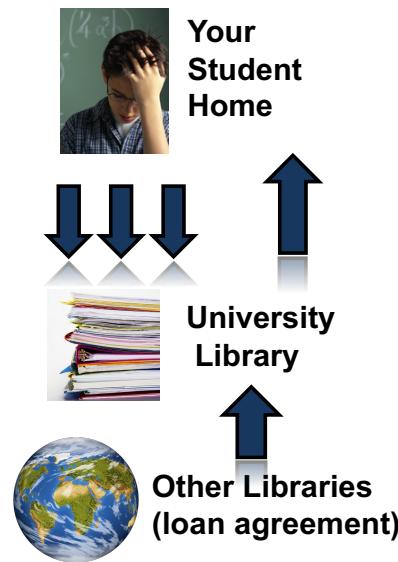
7

Problem 1. Each time you have a problem, you go to the library and read. You have to walk a lot...

Solution 1. You borrow the book you read often.

Temporal Locality
If you recently used a book, it is likely that you will read it again.

Problem 3. The library cannot store all possible books in the world on their shelves.



Problem 2. Each time you find a closely related problem, you borrow a new book. You have to go and borrow books many times.

Solution 2. When you borrow a book, you borrow 10 books on the same shelf, with closely related topics.

Spatial Locality
When you borrow a particular book, you are probably interested in other similar books.

Solution 3. The library can borrow from other libraries, when requested by you.

David Broman
dbro@kth.se



Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory



Memory Technologies (1/2) SRAM and DRAM

8



SRAM (Static Random Access Memory)

- Simple integrated circuit, usually with one access port.
- Today, on-chip memory (same as processor)
- Access time 0.5-2.5ns Cost per GiB in 2012: \$500-\$1000



Douglas Whitaker, Wikipedia, CC BY-SA 2.5

DRAM (Dynamic Random Access Memory)

- Memory stored in capacitors – need to be refreshed
- One transistor per bit – much cheaper than SRAM
- SDRAM (synchronous DRAM). Uses clocks. Transfer data in bursts.
- DDR (Double Data Rate) SDRAM. Transfer data both on rising and falling clock edge.
- Access time: 50-70ns, Cost per GiB in 2012: \$10-\$20

Source: Patterson and Hennessy, 2012

David Broman
dbro@kth.se



Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory



Wikipedia, CC BY-SA 3.0



Wikipedia Evan Amos, CC BY-SA 3.0

Flash Memory

- Electrically erasable programmable read-only memory (EEPROM)
- Can wear out
- Used in solid state drives (SSD)
- Access time: 5,000-50,000 ns, Cost per GiB in 2012 \$0.75-\$1

Magnetic Disk

- Collection of platters that spin 5,400 to 15,000 revolutions per minutes (rpm).
- Access time: 5,000,000-50,000,000 ns
Cost per GiB in 2012 \$0.05-\$0.10

Clearly, there is a tradeoff between cost, access time, and size

How can we utilize these differences? Source: Patterson and Hennessy, 2012

David Broman
dbro@kth.se



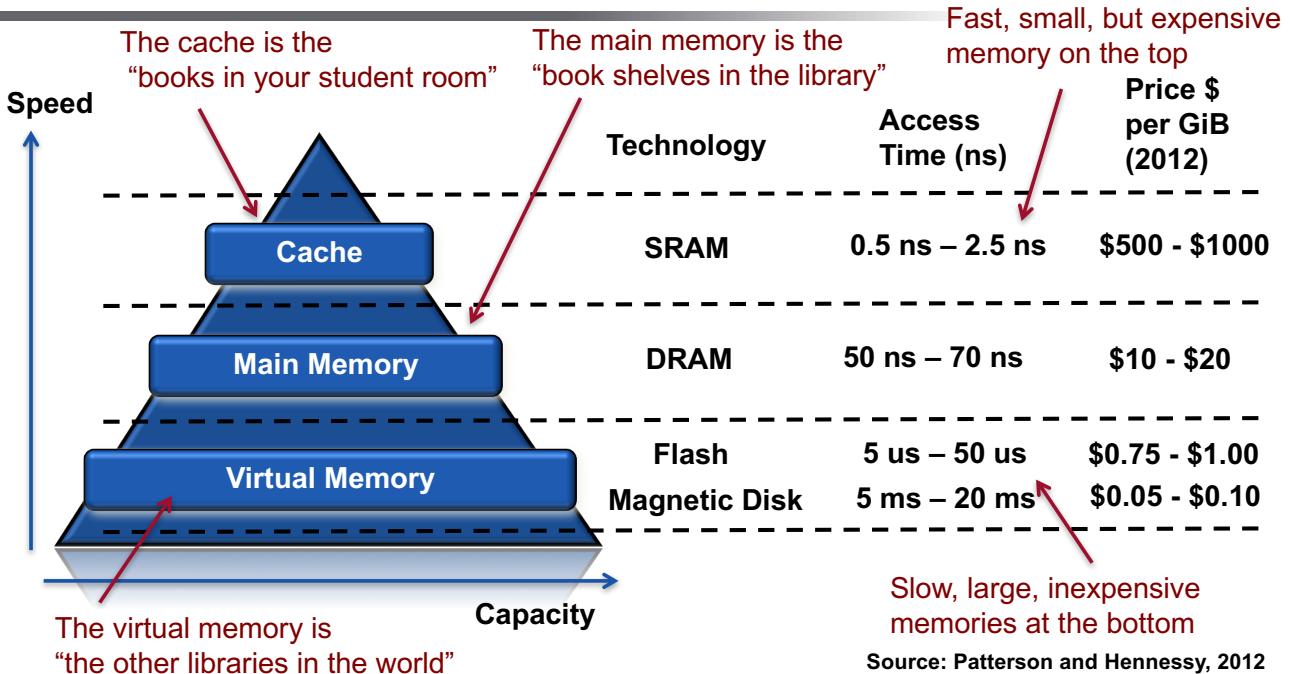
Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory



Memory Hierarchy



David Broman
dbro@kth.se



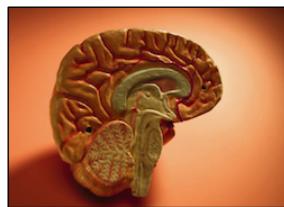
Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory

Part II

Instruction and Data Caches



Acknowledgement: The structure and several of the good examples are derived from the book "Digital Design and Computer Architecture" (2013) by D. M. Harris and S. L. Harris.

David Broman
dbro@kth.se

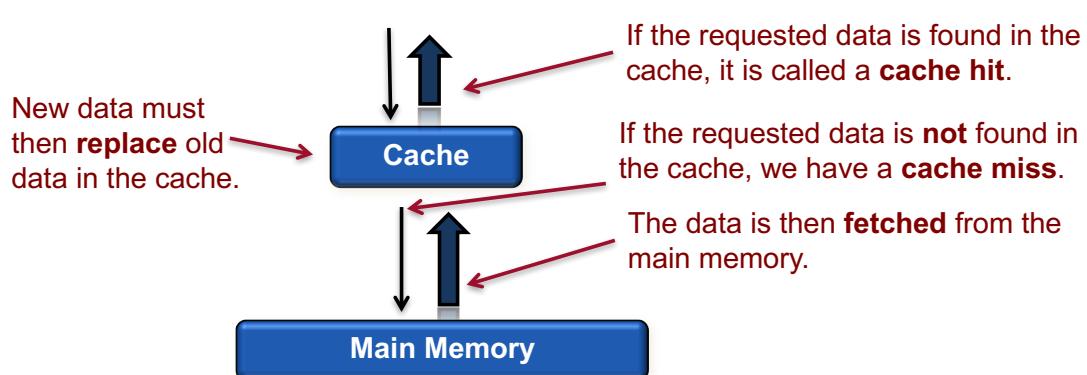
Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory



Reading From Memory



$$\text{Miss Rate} = \frac{\text{Number of misses}}{\text{Total number of memory accesses}}$$

$$\text{Hit Rate} = \frac{\text{Number of hits}}{\text{Total number of memory accesses}}$$

What data should be in the cache so that we maximize the hit rate?

David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

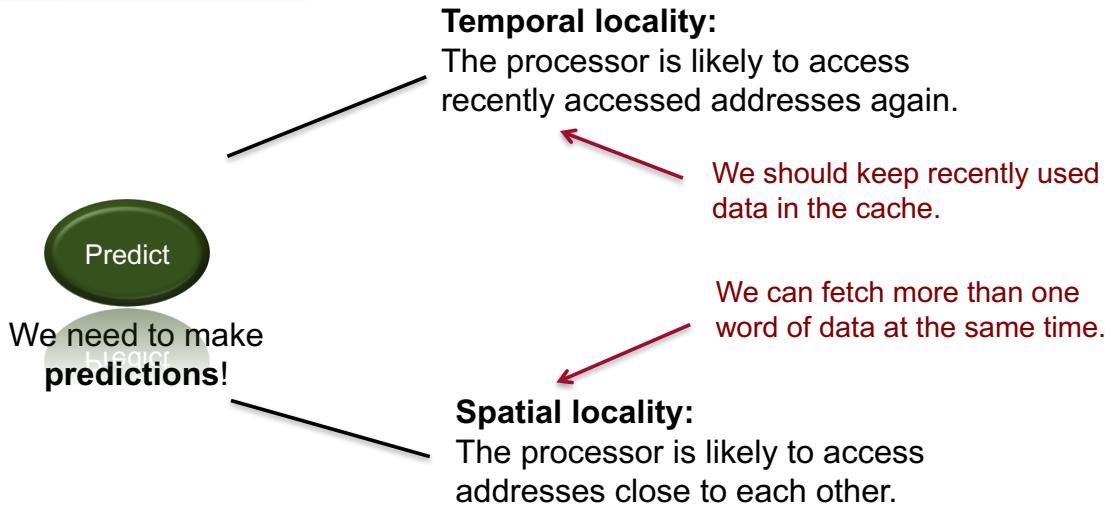
Part II
Instruction and
Data Caches

Part III
Virtual
Memory

How to achieve low miss rates?

**"It's difficult to make predictions,
especially about the future"**

Attributed to various
persons in the history



David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory



Cache Terminology

Capacity (C)

Number of words or bytes in the cache.

Number of Sets (S)

Each set holds one or more blocks of data. (Sometimes the term **row** is used instead of set)

Block size (b)

Number of words or bytes in a block.

Number of Blocks (B)

The total number of blocks.
Always: $B \geq S$

Degree of associativity (N)

$$N = B/S$$

Minimal Cache Example

$$N = 1$$

Here $B=8$,
i.e., $B=S$.

0x0000 0000

The block size is one word, $b=1$.

The **capacity** of cache is $2^3 = 8$ words. The word size is 32-bit.

Set 7, 111₂
Set 6, 110₂
Set 5, 101₂
Set 4, 100₂
Set 3, 011₂
Set 2, 010₂
Set 1, 001₂
Set 0, 000₂

There are 8 sets, i.e., $S=8$.

David Broman
dbro@kth.se

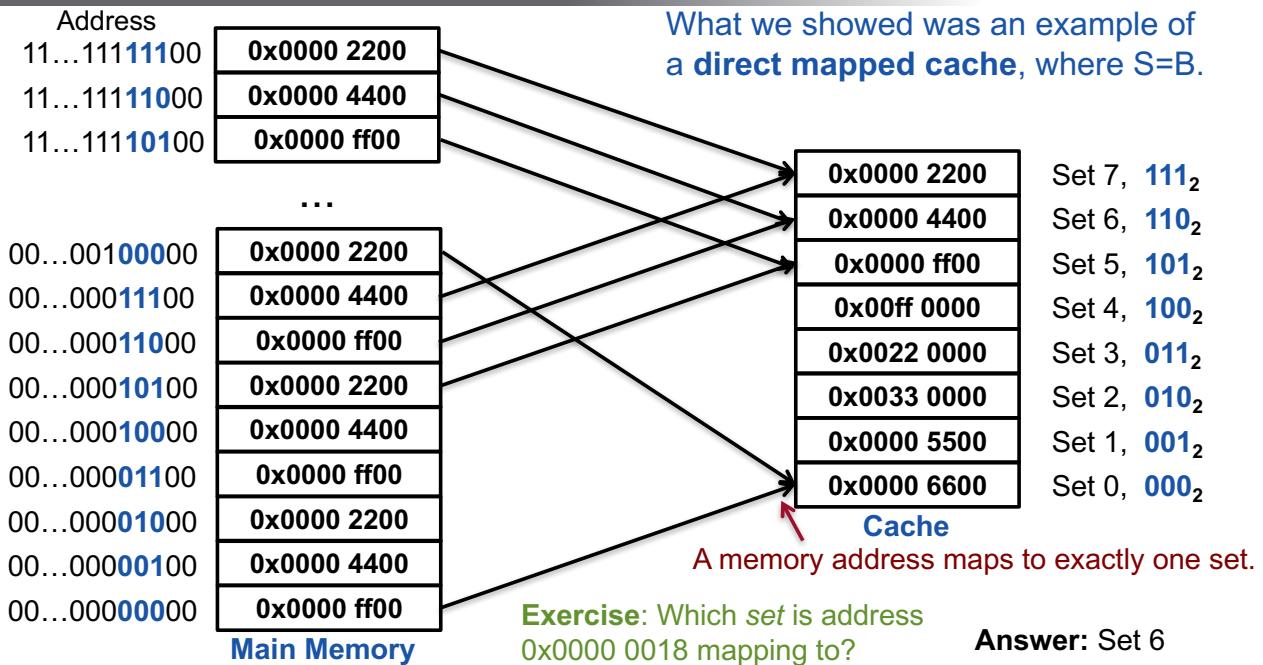
Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory

Direct Mapped Cache (1/3)

The mapping



David Broman
dbro@kth.se

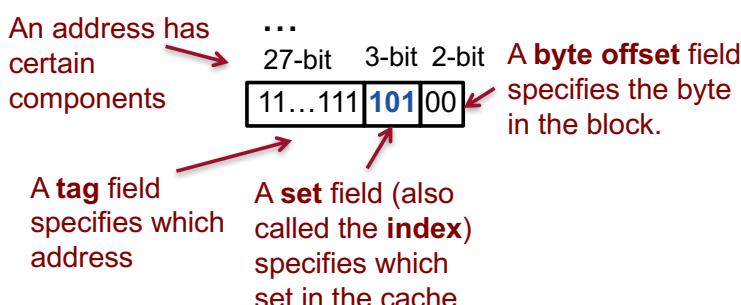
Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory

Direct Mapped Cache (2/3)

Cache Fields



It is possible to further divide the offset into block offsets (see H&H p. 488). However, for simplicity, in the exercises and exams in this course, we just talk about one offset, simply referred to as the **byte offset**.

Exercise: If C = 256 words, N=1, b = 4 words, and address and word size are 32-bits.
How many bits are then needed for the tag and the set fields?

Answer:

byte offset field = 4 bits
set field = 6 bits
tag field = 22 bits

How do we know which memory blocks that are stored in the cache?

Answer: We store the tag field in the cache.

How do we know if the data stored in the cache is valid?

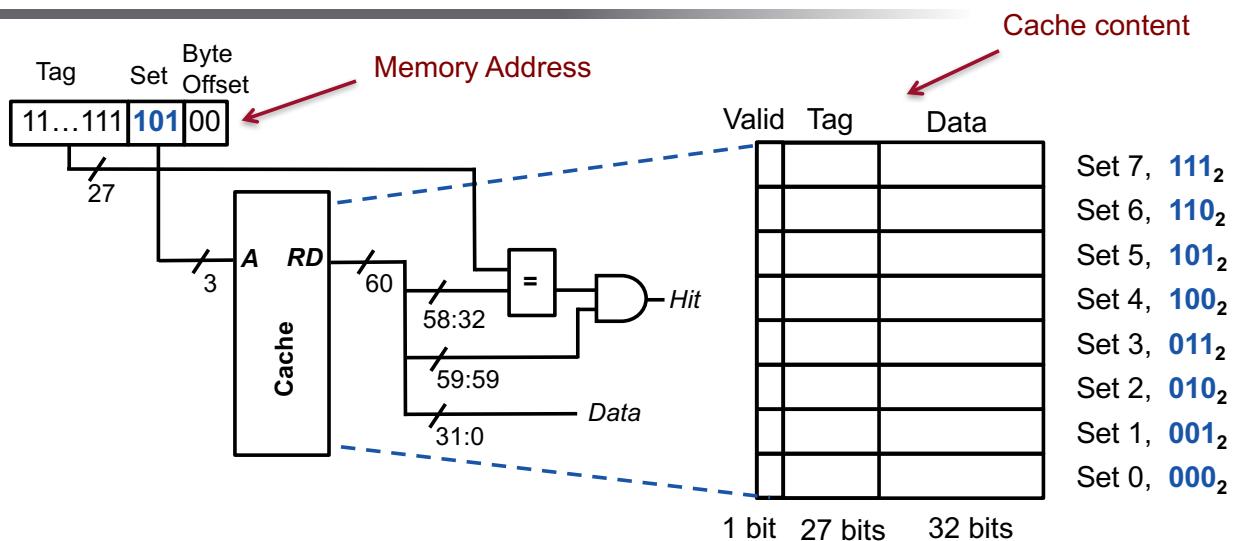
Answer: We add a valid bit in the cache.

David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory



Loop Example 1 Data Cache – Temporal Locality



```

addi $t0, $0, 5
loop:
    beq $t0, $0, done
    lw $t1, 0x4($0)
    lw $t2, 0xC($0)
    addi $t0, $t0, -1
    j loop
done:

```

Valid	Tag	Data	
0			Set 7, 111_2
0			Set 6, 110_2
0			Set 5, 101_2
0			Set 4, 100_2
1	00..00	mem[0x00..0C]	Set 3, 011_2
0			Set 2, 010_2
1	00..00	mem[0x0004]	Set 1, 001_2
0			Set 0, 000_2

Exercise: Assume that the cache is empty when entering the program. What is the *data* cache miss rate and the cache contents when reaching program point **done**.

1 bit 27 bits 32 bits

Answer: The missrate is $2/10 = 20\%$.

$$4_{16} = 00100_2 \quad C_{16} = 12_{10} = 01100_2$$

```

addi $t0, $0, 5
loop:
    beq $t0, $0, done
    lw   $t1, 0x4($0)
    lw   $t2, 0xC($0)
    addi $t0, $t0, -1
    j    loop
done:

```

Exercise: Assume that the first `addi` instruction starts at address 0x0000 4000. The instruction cache has $S = B$, $C = 4096$ bytes and $b = 16$ bytes. How many instruction cache misses occurs when executing the program, presupposed that the cache was empty from the beginning.

Note the **spatial locality**. Since we load 4 instruction each time, we do not get cache misses for each instruction.

Answer: Two cache misses

First, when loading the first 4 instructions, then when loading the two last instructions.

4 bits for representing 16 bytes block
 0x0000 4000 // Address to first addi
 0x0000 4010 // Address to second addi
 The mapping does not conflict.

David Broman
 dbro@kth.se

Part I
 Overview of
 Memory Hierarchies

Part II
 Instruction and
 Data Caches

Part III
 Virtual
 Memory

```

addi $t0, $0, 5
loop:
    beq $t0, $0, done
    lw   $t1, 0x4($0)
    lw   $t2, 0x24($0)
    addi $t0, $t0, -1
    j    loop
done:

```

Exercise: Assume that the cache is empty when entering the program. What is the *data* cache miss rate and the cache contents when reaching program point `done`.

Valid	Tag	Data	
0			Set 7, 111_2
0			Set 6, 110_2
0			Set 5, 101_2
0			Set 4, 100_2
0			Set 3, 011_2
0			Set 2, 010_2
1	00..00	mem[0x0024]	Set 1, 001_2
0			Set 0, 000_2

1 bit 27 bits 32 bits

Answer:

$$4_{16} = 0000\ 0100_2 \quad 24_{16} = 0010\ 0100_2$$

The miss rate is $10/10 = 100\%$.

David Broman
 dbro@kth.se

Part I
 Overview of
 Memory Hierarchies

Part II
 Instruction and
 Data Caches

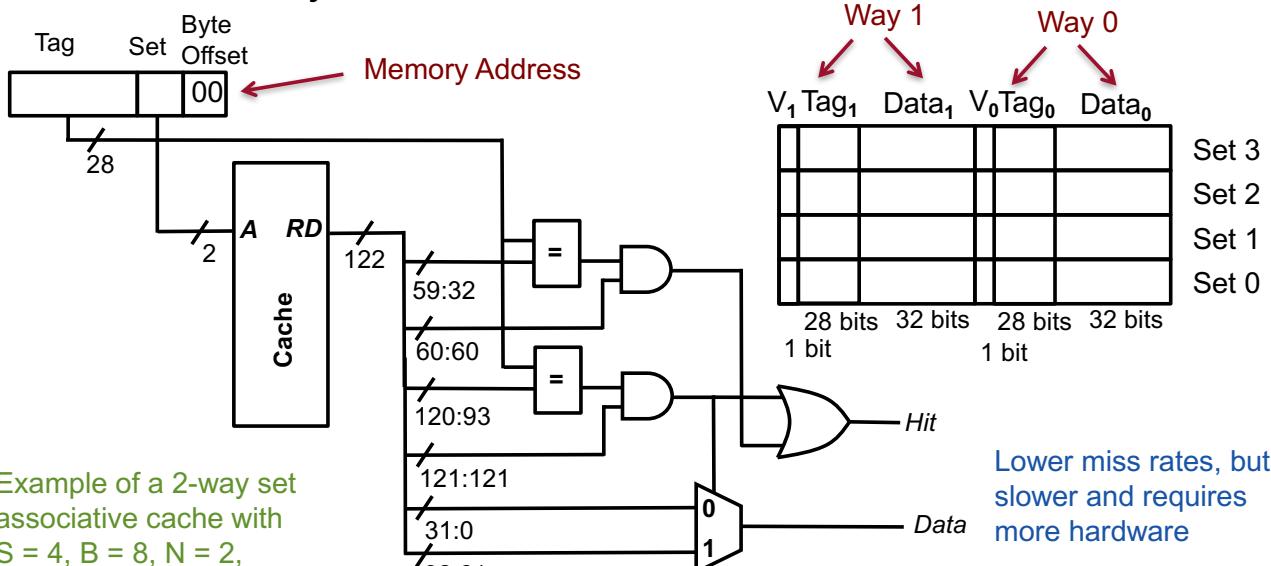
Part III
 Virtual
 Memory



N-way Set Associative Cache

We can reduce conflicts by having N blocks in each set.

This is called an **N-way Set Associative Cache**.



Example of a 2-way set associative cache with $S = 4$, $B = 8$, $N = 2$, $b = 1$ word

David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory

Lower miss rates, but
slower and requires
more hardware



Fully Associative Cache and Direct Mapped Cache revisited

All caches are **N-way set associative caches**.

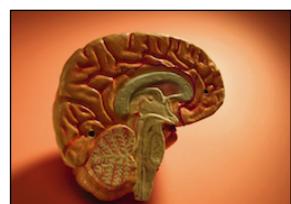
A **fully associative cache** has $B = N$ and $S = 1$.

Another name for a fully associative cache is a **B-way set associative cache**.

Gives the lowest miss rate, but
requires most hardware.

A **direct mapped cache** has $N = 1$ and $B = S$.

Another name for a direct mapped cache is a **one-way set associative cache**.



David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory



Replacement Policy

Direct Mapped Cache

Each address maps to a unique block and set.

Hence, when a set is full, it must be replaced with the new data.

N-way Set Associative Cache where N > 1

- **Least Recently Used (LRU) Policy.** Simple with a 2-way set associative cache by using a *use bit U*. Commonly used.
- **Pseudo-LRU Policy.** For N-ways where N > 2. Indicate the least recently used *group* and upon replacement, randomly selects a way in the group.
- **First-in First-out (FIFO)** replacement policy.
- **Random** replacement policy.

David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

 **Part II**
Instruction and
Data Caches

Part III
Virtual
Memory



Multi-Level Caches

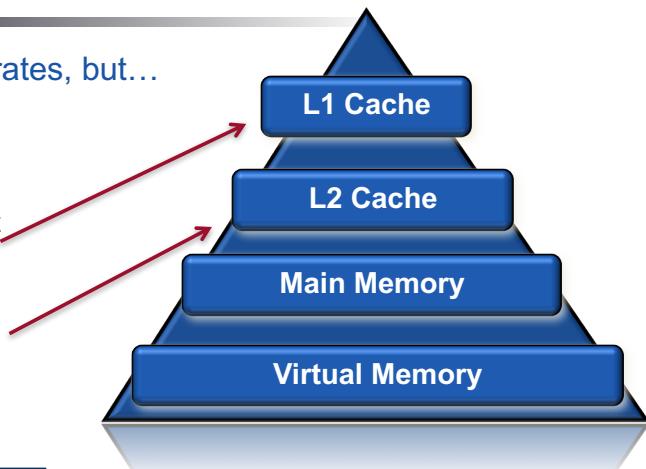
Large caches tend give lower miss rates, but...

Large caches tend to be slower.

Solution: Multi-Level Caches

L1 cache, small enough to get
1-2 cycle times.

The L2 cache is larger
and therefore slower.



Examples from Reality

ARM Cortex-A8

- L1, 4-way, 32KiB, split instruction/data, random replacement
- L2, 8-way, 128KiB, unified inst./data, random replacement
- No L3 cache

Intel Core-i7 920

- L1, 4-way (i), 8-way (d), 32KiB, split instruction/data, Approximate LRU
- L2, 8-way, 256KiB, unified inst./data
- L3, 16-way, 8MiB, Approximate LRU

David Broman
dbro@kth.se

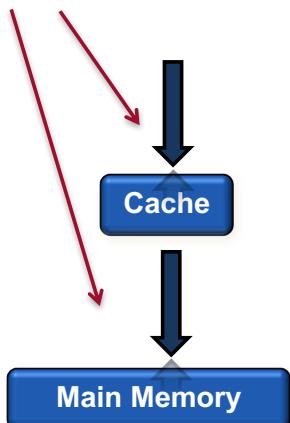
Part I
Overview of
Memory Hierarchies

 **Part II**
Instruction and
Data Caches

Part III
Virtual
Memory

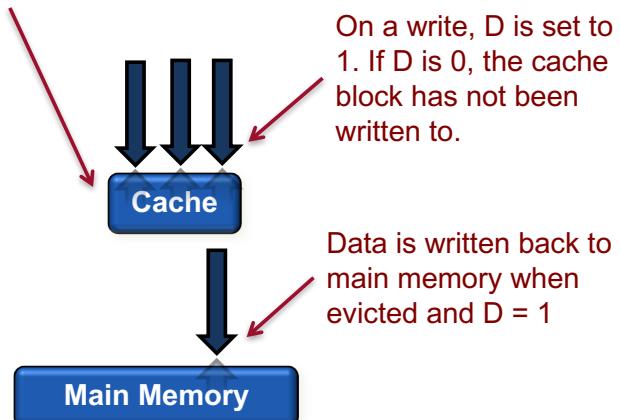
Write-Through Policy

When writing to the cache, the data is simultaneously written back to main memory.



Write-Back Policy

A dirty bit (D) is associated with each cache block.



David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

 **Part II**
Instruction and
Data Caches

Part III
Virtual
Memory



Part III

Virtual Memory



David Broman
dbro@kth.se

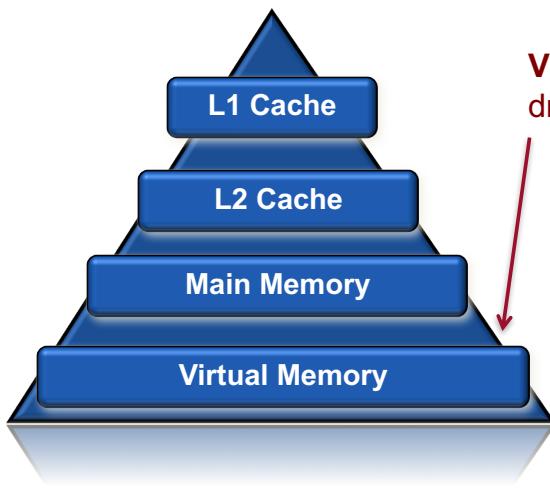
Part I
Overview of
Memory Hierarchies

 **Part II**
Instruction and
Data Caches

 **Part III**
Virtual
Memory



Rationales for Virtual Memory



Virtual memory uses the hard drive. Slow, but large memory.

Why?

- Gives the illusion of a **very big memory**.
- Gives **memory protection** between concurrent running programs (each process has its own virtual memory space).

David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

 **Part III**
Virtual
Memory



Virtual Memory vs. Caches

Virtual memory has similarities to caches, but uses another terminology.

Cache	Virtual Memory
Block	Page
Block size	Page Size
Block offset	Page offset
Miss	Page fault
Tag	Virtual Page number

Typically, 4KB or more

David Broman
dbro@kth.se

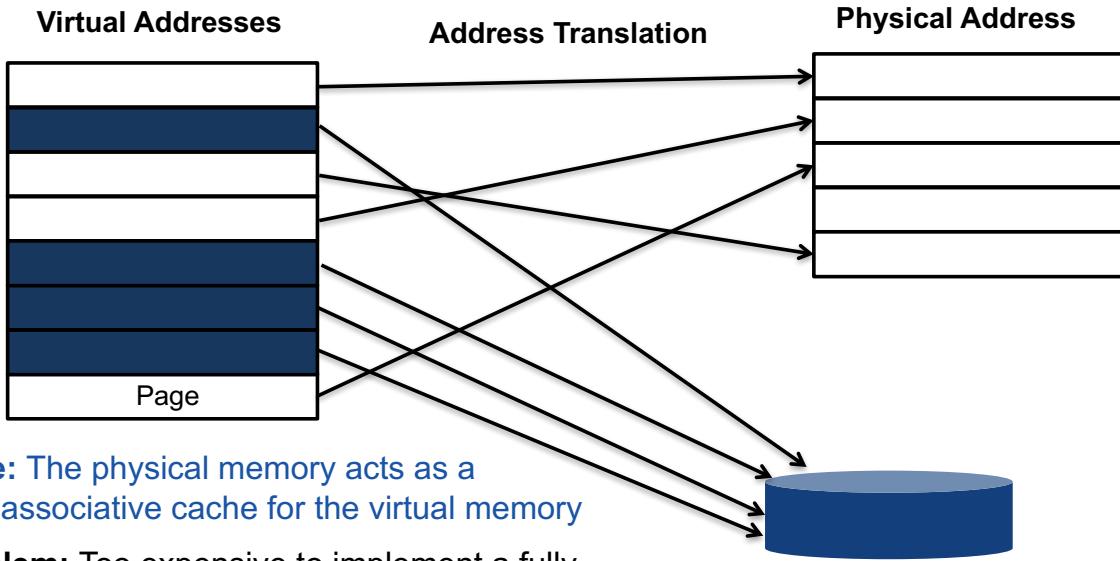
Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

 **Part III**
Virtual
Memory



Virtual Memory Overview



David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory

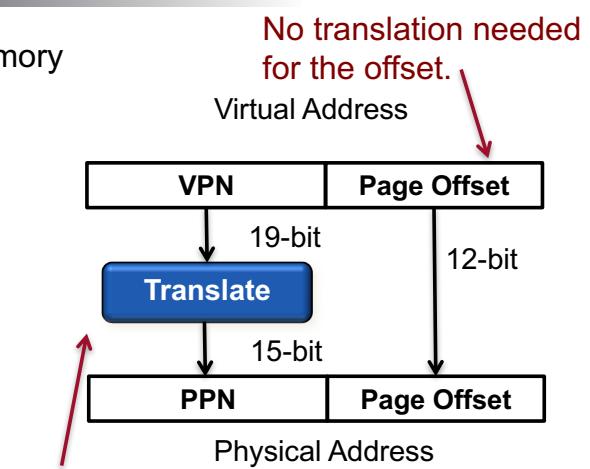


Page Table and the Memory Management Unit (MMU)

Solution: Page table, stored in Physical Memory

Valid	Physical Page Number	Virtual Page Number
0	0x5FFF	0x5FFFF
0	0x5FFE	0x5FFFE
0	0x5FFD	0x5FFFD
0	0x5FFC	0x5FFFC
0	0x5FFB	0x5FFFB
...		
1	0x5FFF	0x00001
0		0x00000

The **operating system (OS)** is responsible for updating the page table and moving data between memory and disk



No translation needed for the offset.
Translate the virtual page number (VPN) to the physical page number (PPN). Done in hardware in the **Memory Management Unit (MMU)**.

David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory



The Translation Lookaside Buffer (TLB)

Problem: Accessing the page table each memory access would result in extreme performance degradation.

A Translation Lookaside Buffer (TLB) caches the latest pages.

A TLB is typically organized as a **fully associative cache**. Holds typically 16 to 512 entries.



Due to temporal and spatial locality, a TLB typically has a hit rate of more than 99%

David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

 **Part III**
Virtual
Memory



You know the drill...
(simma lugnt)



David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory



Reading Guidelines



Module 5: Memory Hierarchy

- H&H Chapters 5.5.1-5.5.6, 8.1-8.3
- H&H Chapter 8.4 (except 8.4.6)

Reading Guidelines

See the course webpage
for more information.

David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory



Summary

Some key take away points:

- **Memory hierarchies** are used because memories have different cost, size, and speed.
- There are two kinds of caches: **instruction caches** and **data caches**.
- Two important properties that make caches useful: **temporal locality** and **spatial locality**.
- Caches can be **direct mapped**, **N-way**, or **fully associative**.
- **Virtual memories** enable large virtual address spaces and enable memory protection between different concurrent programs.



Thanks for listening!

David Broman
dbro@kth.se

Part I
Overview of
Memory Hierarchies

Part II
Instruction and
Data Caches

Part III
Virtual
Memory