Memory & Cache

Computer Organization
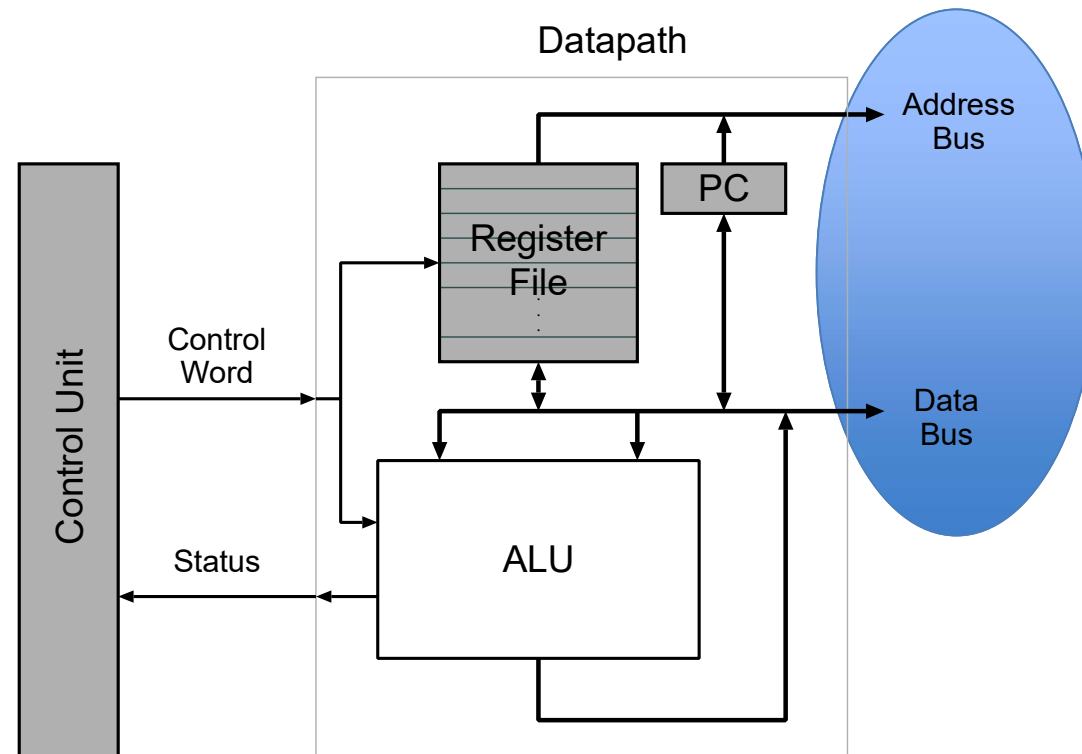
Wednesday, 25 September 2024
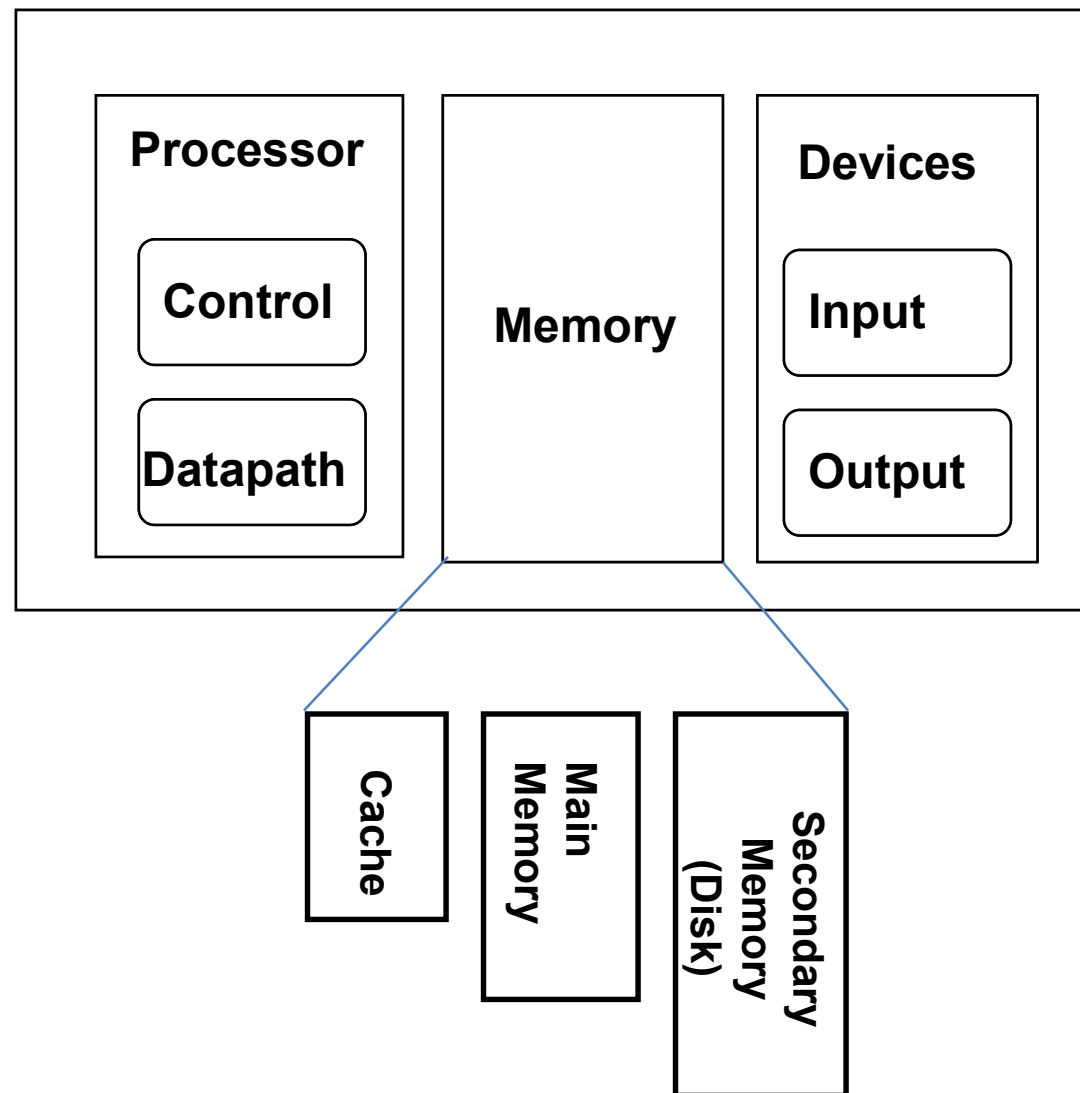
TÉCNICO LISBOA

Summary

- ## Previous Class
  - Role of the Compiler

- ## Today:
  - Memory hierarchy
  - Caches

Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: sequences datapath, memory, …

- ## Trace < command, address, [data] >

Datapath

Address Bus

PC

Register File

Control Word
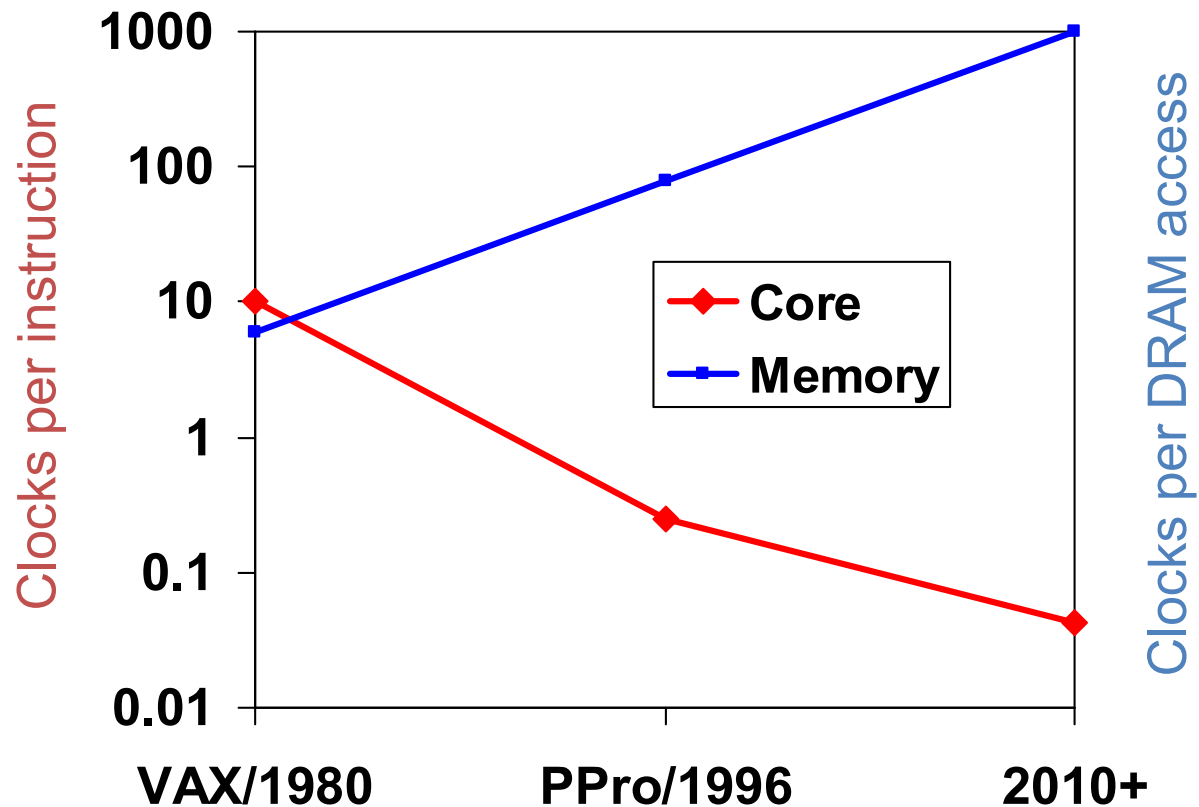
Control Unit

Status

ALU

Data Bus

Review: Major Components of a Computer

The "Memory Wall"

- Processor vs DRAM speed disparity continues to grow

- Good memory hierarchy (cache) design is increasingly important to overall performance

Memory Technology

- Static RAM (SRAM)
  - 0.5ns – 2.5ns   €1000 – €1000 per GB

- Dynamic RAM (DRAM)
  - 50ns – 70ns   €10 – €20 per GB

- Flash memory
  - 5µs – 50µs   €0.5 – €1 per GB

- Magnetic disk
  - 5ms – 20ms   €0.05 – €0.5 per GB

- Ideal memory
  - Access time of SRAM
  - Capacity and cost/GB of disk

A Typical Memory Hierarchy

- Take advantage of the principle of locality to present the user with as much memory as is available in the *cheapest* technology at the speed offered by the *fastest* technology

Speed (%cycles):	½'s	1's	10's	100's	10,000's
Size (bytes):	100's	10K's	M's	G's	T's
Cost:	highest				lowest
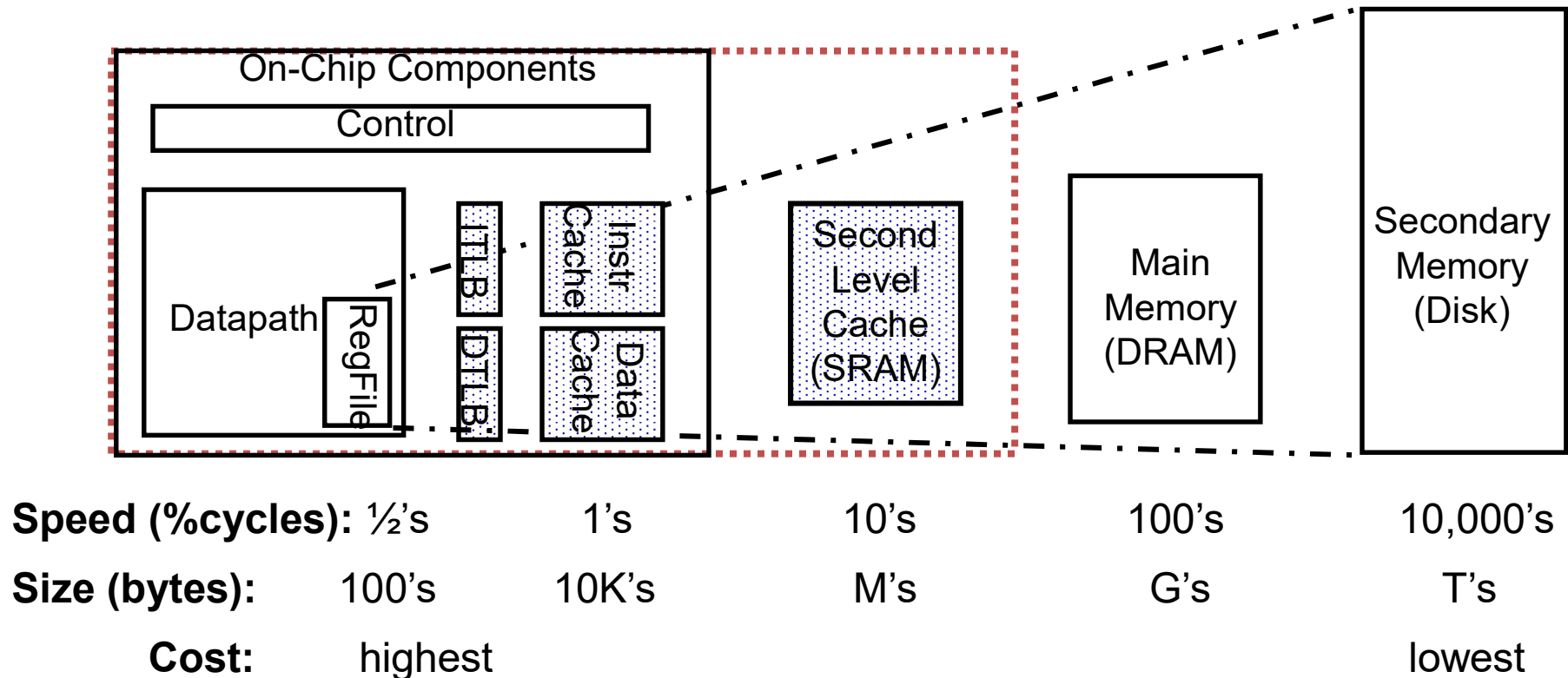
Memory Hierarchy Technologies

- Caches use SRAM for speed and technology compatibility
  - Fast (typical access times of 0.5 to 2.5 nsec)
  - Low density (6 transistor cells), higher power, expensive
  - Static: content will last "forever" (as long as power is left on)

- Main memory uses DRAM for size (density)
  - Slower (typical access times of 50 to 70 nsec)
  - High density (1 transistor cells), lower power, cheaper
  - Dynamic: needs to be "refreshed" regularly (~ every 8 ms)
    - consumes1% to 2% of the active cycles of the DRAM
  - Addresses divided into 2 halves (row and column)
    - RAS or Row Access Strobe triggering the row decoder
    - CAS or Column Access Strobe triggering the column selector

The Memory Hierarchy: Why Does it Work?

- ## Temporal Locality (locality in time)

  If a memory location is referenced then it will tend to be referenced again soon

  ➡ Keep most recently accessed data items closer to the processor

  e.g., instructions in a loop, induction variables

- ## Spatial Locality (locality in space)

  If a memory location is referenced, the locations with nearby addresses will tend to be referenced soon

 ➡ Move blocks consisting of contiguous words closer to the processor

  e.g., sequential instruction access, array data

Taking Advantage of Locality

- **Memory hierarchy**

- **Store everything on disk**

- **Copy recently accessed (and nearby) items from disk to smaller DRAM memory**
  - Main memory

- **Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory**
  - Cache memory attached to CPU

Memory Hierarchy Levels
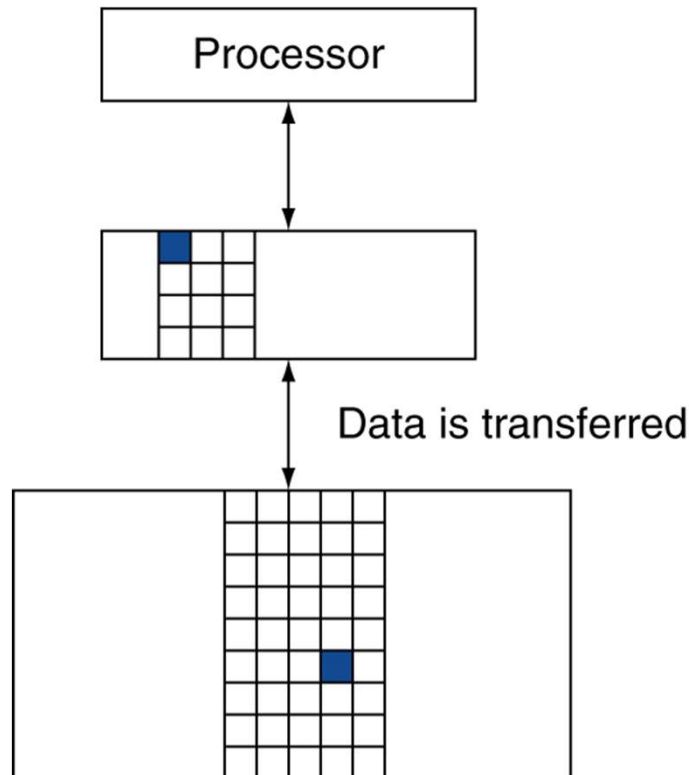
- **Block (aka line): unit of copying**
  - May be multiple words

- **If accessed data is present in upper level**
  - Hit: access satisfied by upper level
    - Hit ratio: hits/accesses

- **If accessed data is absent**
  - Miss: block copied from lower level
    - Time taken: miss penalty
    - Miss ratio: misses/accesses = 1 – hit ratio
  - Then accessed data supplied from upper level

The Memory Hierarchy: Terminology

- **Block** (or line): the minimum unit of information that is present (or not) in a cache

- **Hit Rate**: the fraction of memory accesses found in a level of the memory hierarchy
  - **Hit Time**: Time to access that level which consists of

    Time to access the block + Time to determine hit/miss

- **Miss Rate:** the fraction of memory accesses *not* found in a level of the memory hierarchy $\Rightarrow$ 1 - (Hit Rate)
  - **Miss Penalty**: Time to replace a block in that level with the corresponding block from a lower

    Hit Time << Miss Penalty

How is the Hierarchy Managed?

- registers $\leftrightarrow$ memory
  - by compiler (programmer?)

- cache $\leftrightarrow$ main memory
  - by the cache controller hardware

- main memory $\leftrightarrow$ disks
  - by the operating system (virtual memory)
  - virtual to physical address mapping assisted by the hardware (TLB)
  - by the programmer (files)

Cache Memory

- ## Cache memory

  – The level of the memory hierarchy closest to the CPU

- ## Given accesses $X_1, \ldots, X_{n-1}, X_n$

X_4
X_1
X_{n-2}
X_{n-1}
X_2
X_3

a. Before the reference to X_n

X_4
X_1
X_{n-2}
X_{n-1}
X_2
X_n
X_3

b. After the reference to X_n

– How do we know if the data is present?

– Where do we look?

Direct Mapped Cache

- Location determined by address
- Direct mapped: only one choice

  (Block address) modulo (#Blocks in cache)
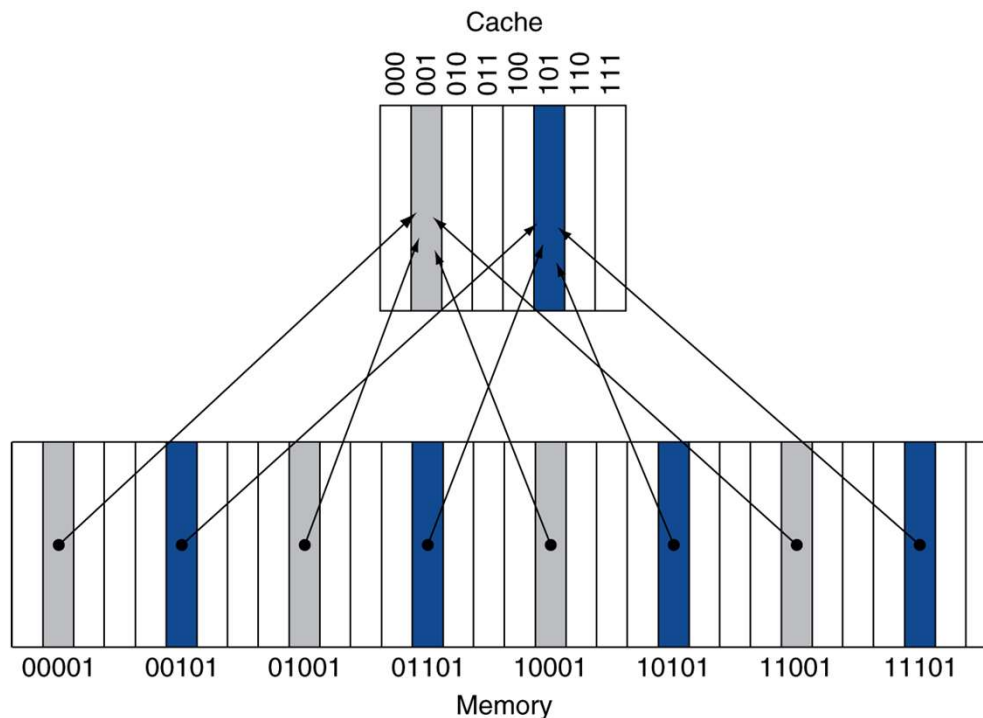
- #Blocks are a power of 2

- Use low-order address bits

Tags and Valid Bits

- How do we know which particular block is stored in a cache location?
  - Store block address as well as the data
    - Actually, only need the high-order bits
    - Called the tag

- What if there is no data in a location?
  - Valid bit: 1 = present, 0 = not present
  - Initially 0

Cache Example

- 8-blocks, 1 word/block, direct mapped
- Initial state

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110		

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 **110**	Miss	110

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

TÉCNICO LISBOA

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
26	11 010		

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
26	11 **010**	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	**Y**	**11**	**Mem[11010]**
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110		

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

TÉCNICO LISBOA

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

Index	V	Tag	Data
000	**Y**	**10**	**Mem[10000]**
001	N		
010	Y	11	Mem[11010]
011	**Y**	**00**	**Mem[00011]**
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
18	10 010		

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	**Y**	**10**	**Mem[10010]**
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

MIPS Direct Mapped Cache Example
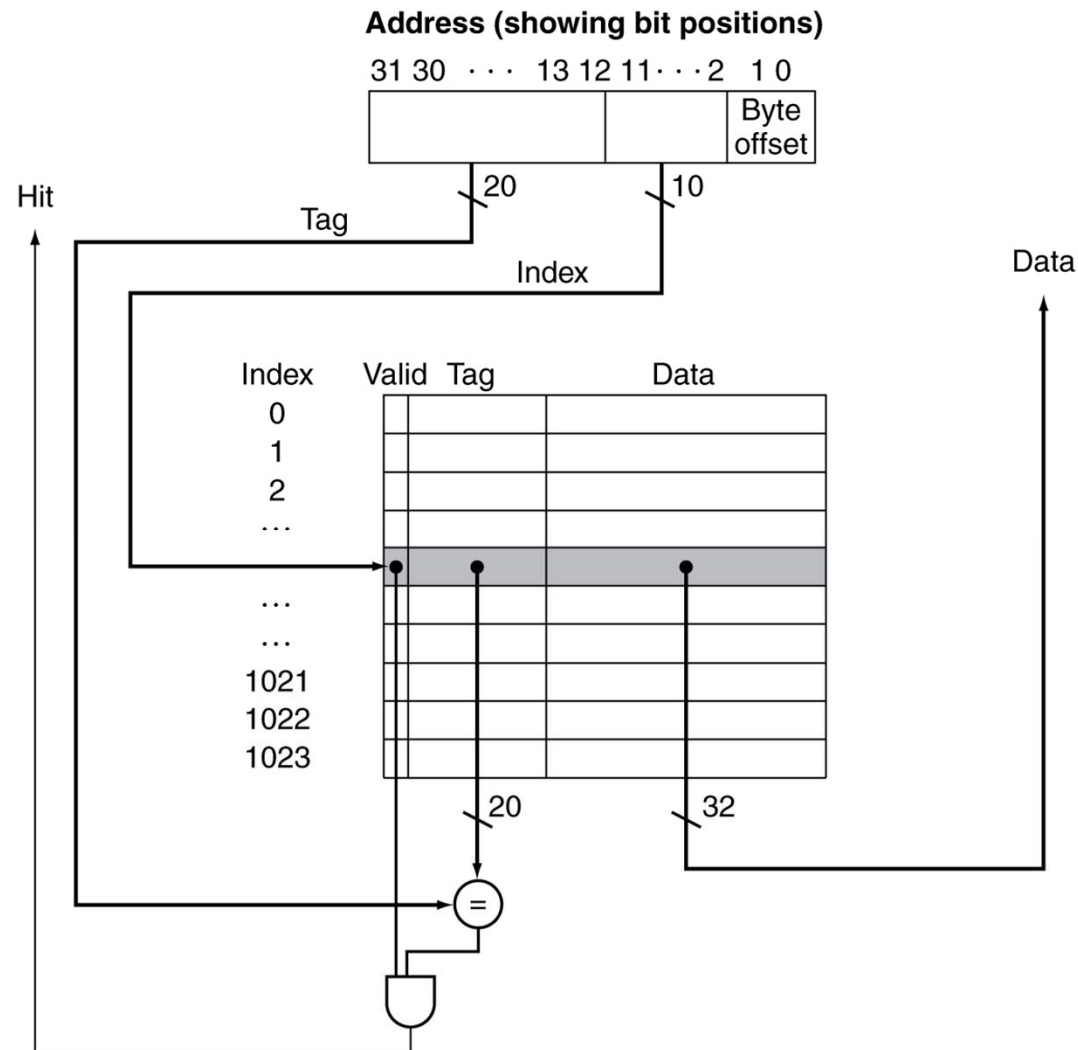
Larger Block Size

- ## 256 blocks, 16 words/block
  - To what block number does address 19200 map?

- ## Block number ?
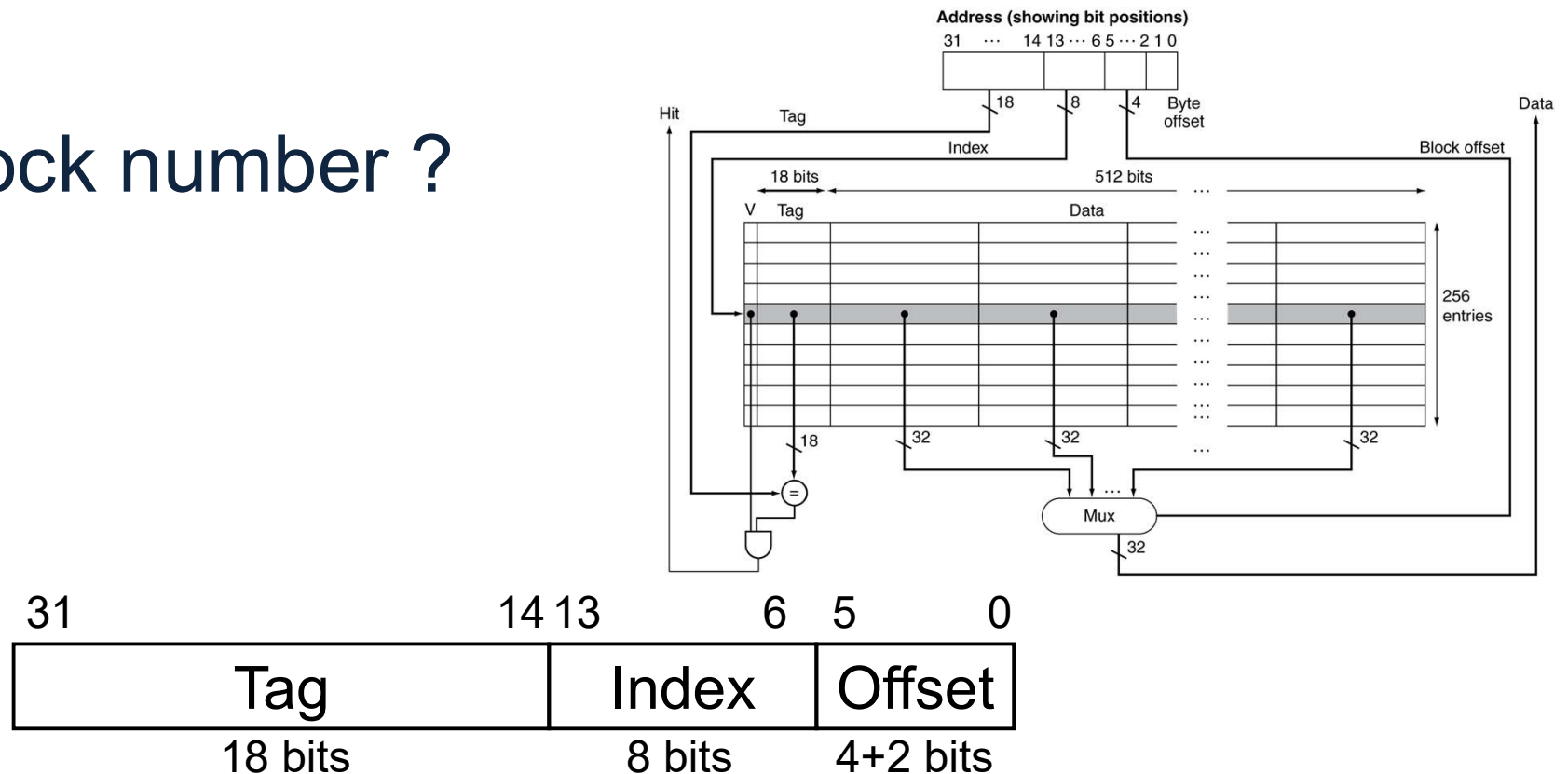
Tag	Index	Offset
18 bits	8 bits	4+2 bits

(bit positions: 31 ... 14 13 ... 6 5 ... 0)
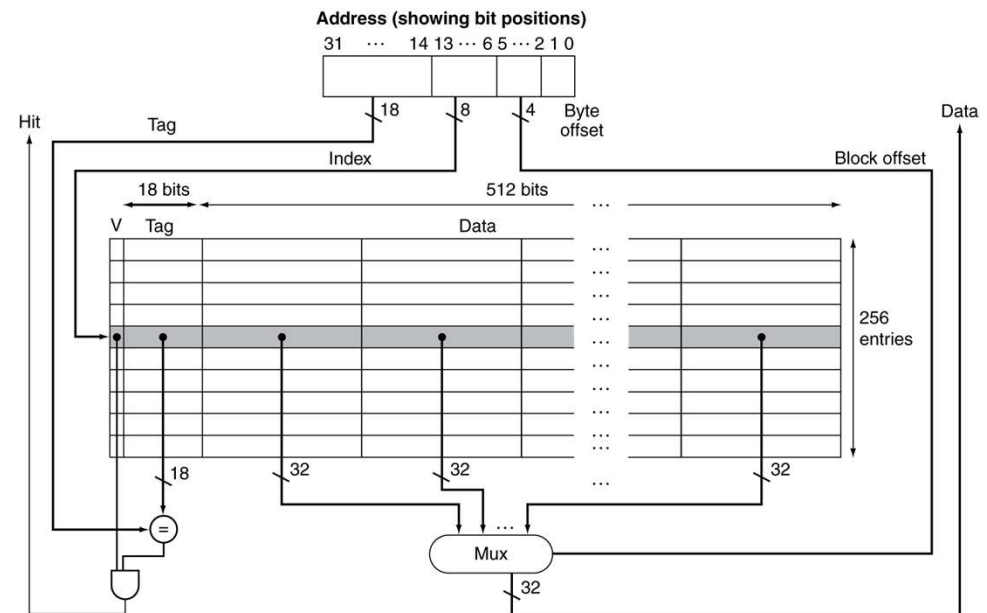
Larger Block Size

- ## 256 blocks, 16 words/block
  - To what block number does address 19200 map?

- ## Block address
  $$= \lfloor 19200/(16 \times 4) \rfloor = 300$$
  -- Removing the Offset

31	14 13	6 5	0
Tag	Index	Offset	
18 bits	8 bits	4+2 bits	

TÉCNICO LISBOA
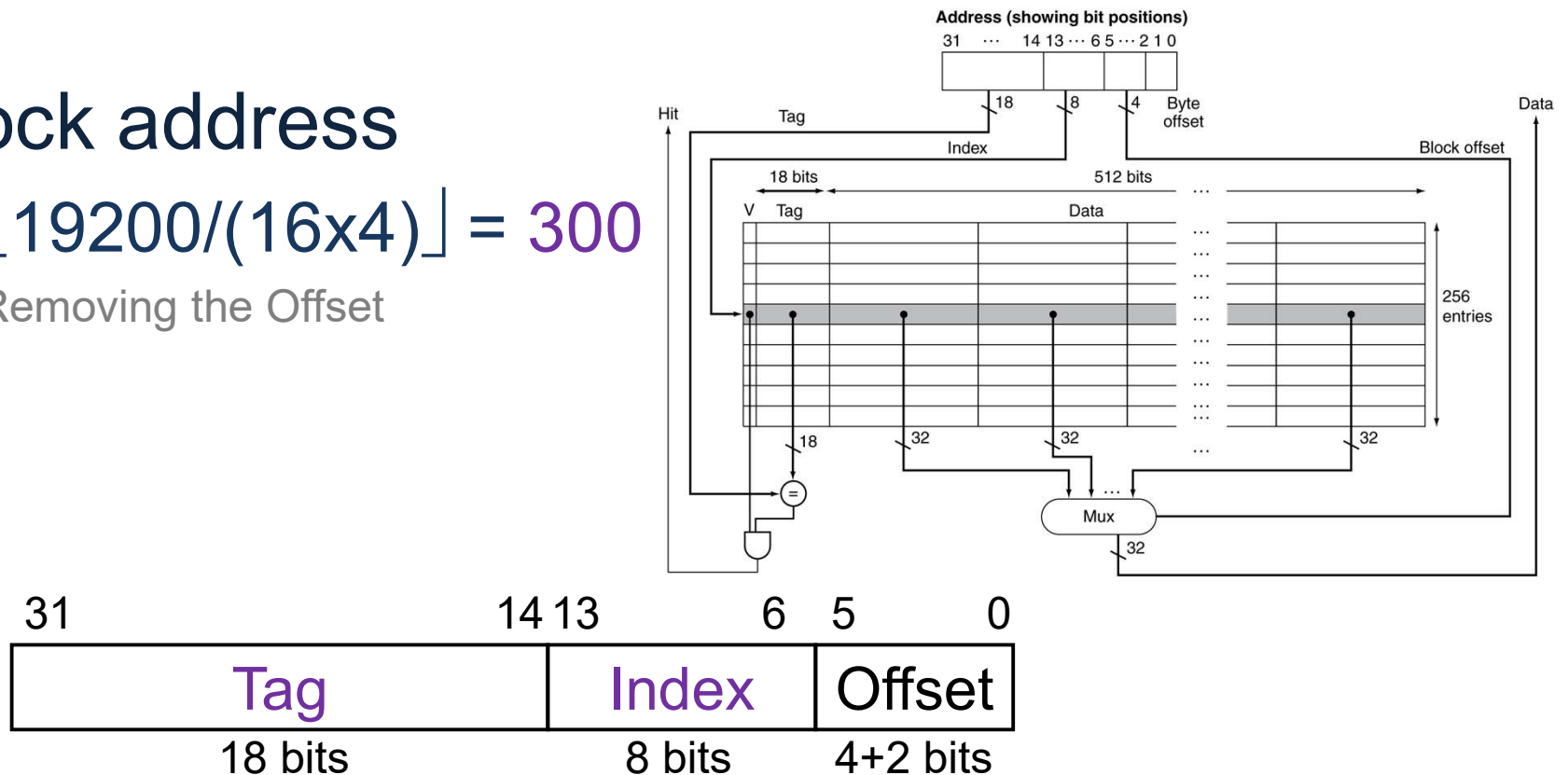
Larger Block Size

- ## 256 blocks, 16 words/block
  - To what block number does address 19200 map?

- ## Block address
  $$= \lfloor 19200/(16 \times 4) \rfloor = 300$$
 -- Removing the Offset

- ## Block number
  $$= 300 \bmod 256 = 44$$

31	14 13	6 5	0
Tag	Index	Offset	
18 bits	8 bits	4+2 bits	
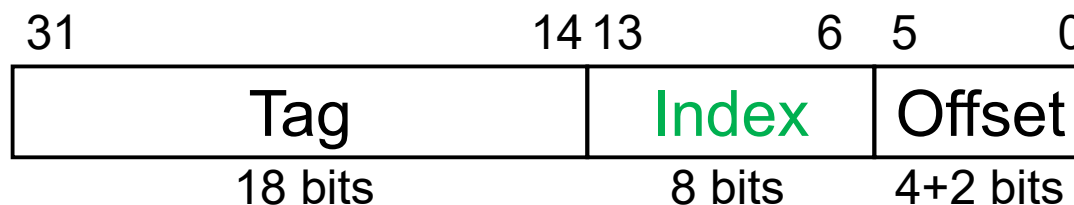
Block Size Considerations

- **Larger blocks should reduce miss rate**
  - Due to spatial locality

- **But in a fixed-sized cache**
  - Larger blocks $\Rightarrow$ fewer blocks
    - More competition $\Rightarrow$ increased miss rate
  - Larger blocks $\Rightarrow$ pollution

- **Larger miss penalty**
  - » since we have more bytes to transfer
  - Can override benefit of reduced miss rate
 - Early restart and critical-word-first can help

Miss Rate vs Block Size vs Cache Size
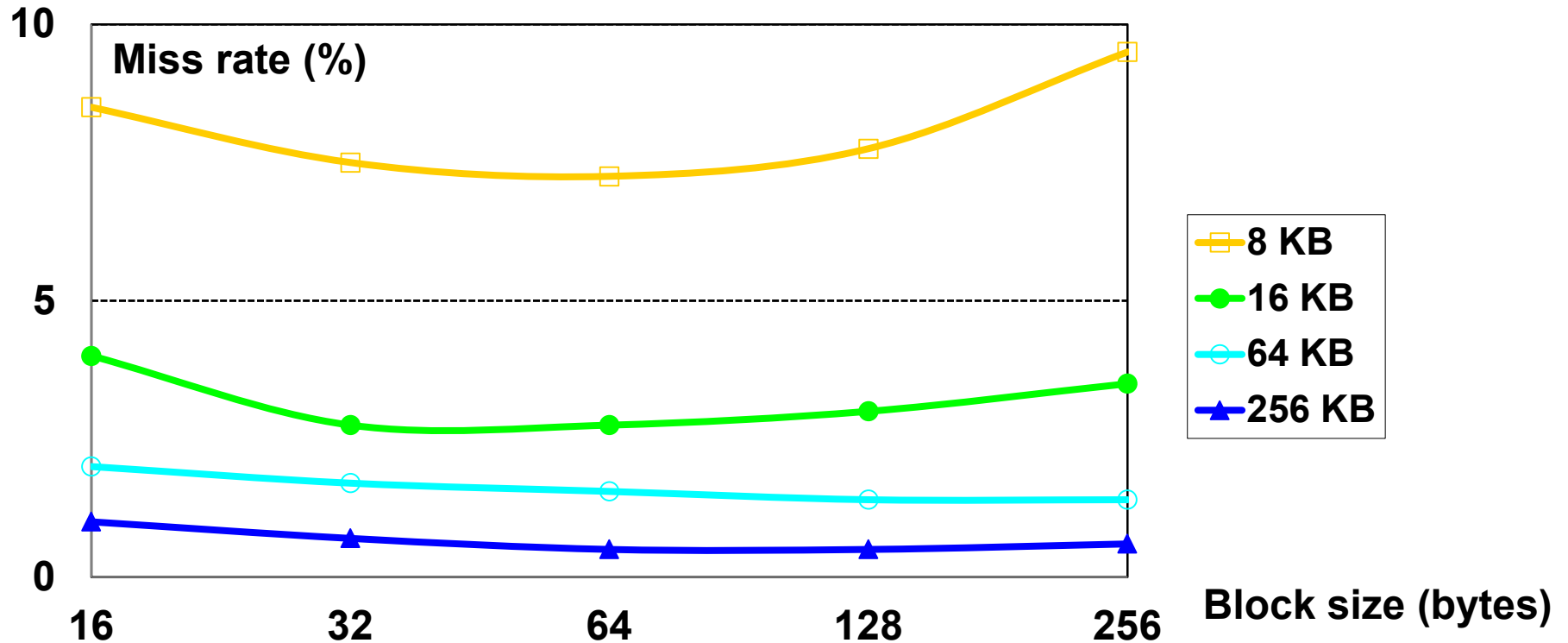
- Miss rate goes up if the block size becomes a significant fraction of the cache size because the number of blocks that can be held in the same size cache is smaller (increasing capacity misses)

Concluding Remarks

- Fast memories are small, large memories are slow
  - We really want fast, large memories
  - Caching gives this illusion
- Principle of locality
  - Programs use a small part of their memory space frequently
- The optimal choice is a compromise
  - depends on access characteristics
    - workload
    - use (I-cache, D-cache, TLB)
  - depends on technology / cost
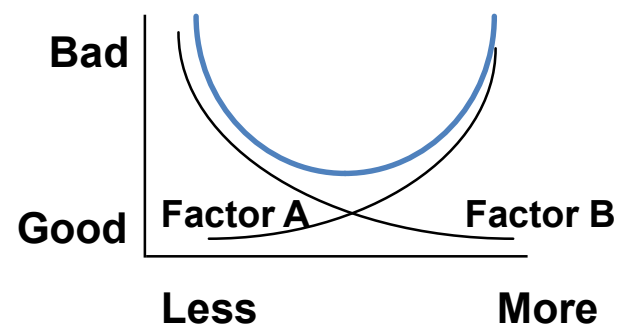
Next Class

- **Improving Cache Performance**

Memory & Cache

Computer Organization

Wednesday, 25 September 2024

TÉCNICO LISBOA

Larger Block Size

- ## 64 blocks, 16 bytes/block
  - To what block number does address 1200 map?

- ## Block address
  $$= \lfloor 1200/16 \rfloor = 75$$

- ## Block number
  $$= 75 \bmod 64 = 11$$

31	10 9	4 3	0
Tag	Index	Offset	
22 bits	6 bits	4 bits	