Memory Technology

Erik Hagersten
Uppsala University, Sweden
eh@it.uu.se

UPPSALA UNIVERSITET

Memory characteristics

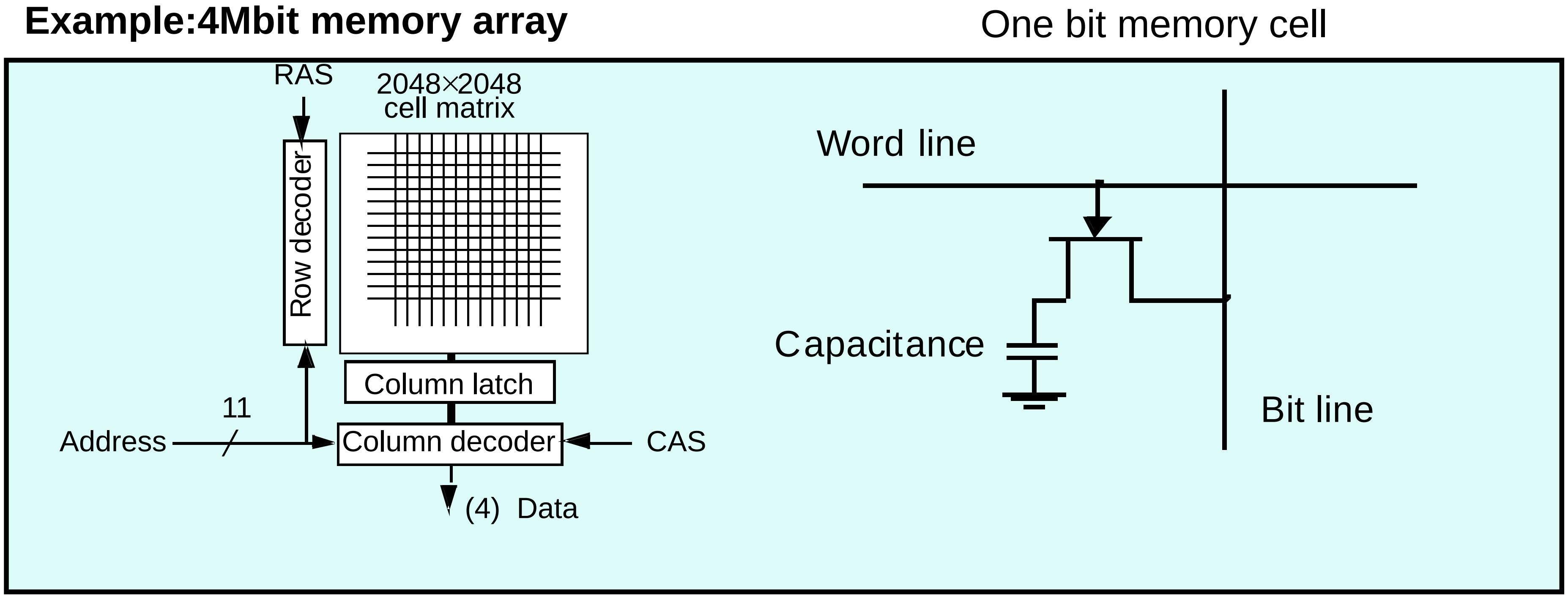
DRAM:

- Main memory is built from ***DRAM***: Dynamic Random Access Memory
- 1 transistor/bit
- Error prone and slow
- Refresh and precharge overhead and complexity
- May require error detection and correction (e.g. ECC)

SRAM

- Cache memory is built from ***SRAM***: Static Random Access Memory
- About 4-6 transistors/bit → fast but requires more transistors per bit stored

DRAM organization



- The address is multiplexed Row/Address Strobe (RAS/CAS)
  1. Read all bits from the row indexed by the **Row Address**
  2. Select the bits identified by the Column Address (wide mux)
- Periodic "refresh" of memory cells
- Bit-error rate creates a need for error-correction (ECC)

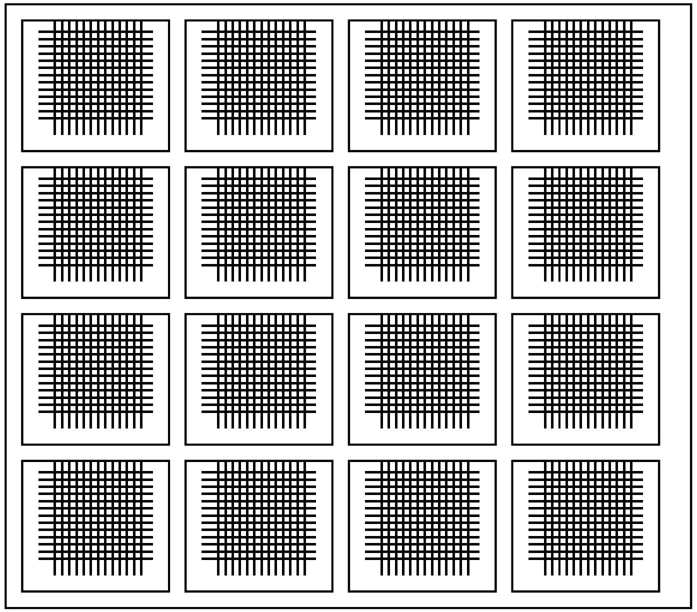
UPPSALA UNIVERSITET

DRAM

- Reading out the first piece of data is slow
- Other data from the same line ”streamed”
- SDRAM (5-1-1-1 @100 MHz)
  - 5 cycles for first data, next data in following cycles
- DDR-DRAM (e.g., 5-½-½-½ @800MHz)
  - Transfer data on both edges of the clock

AVDARK 2013

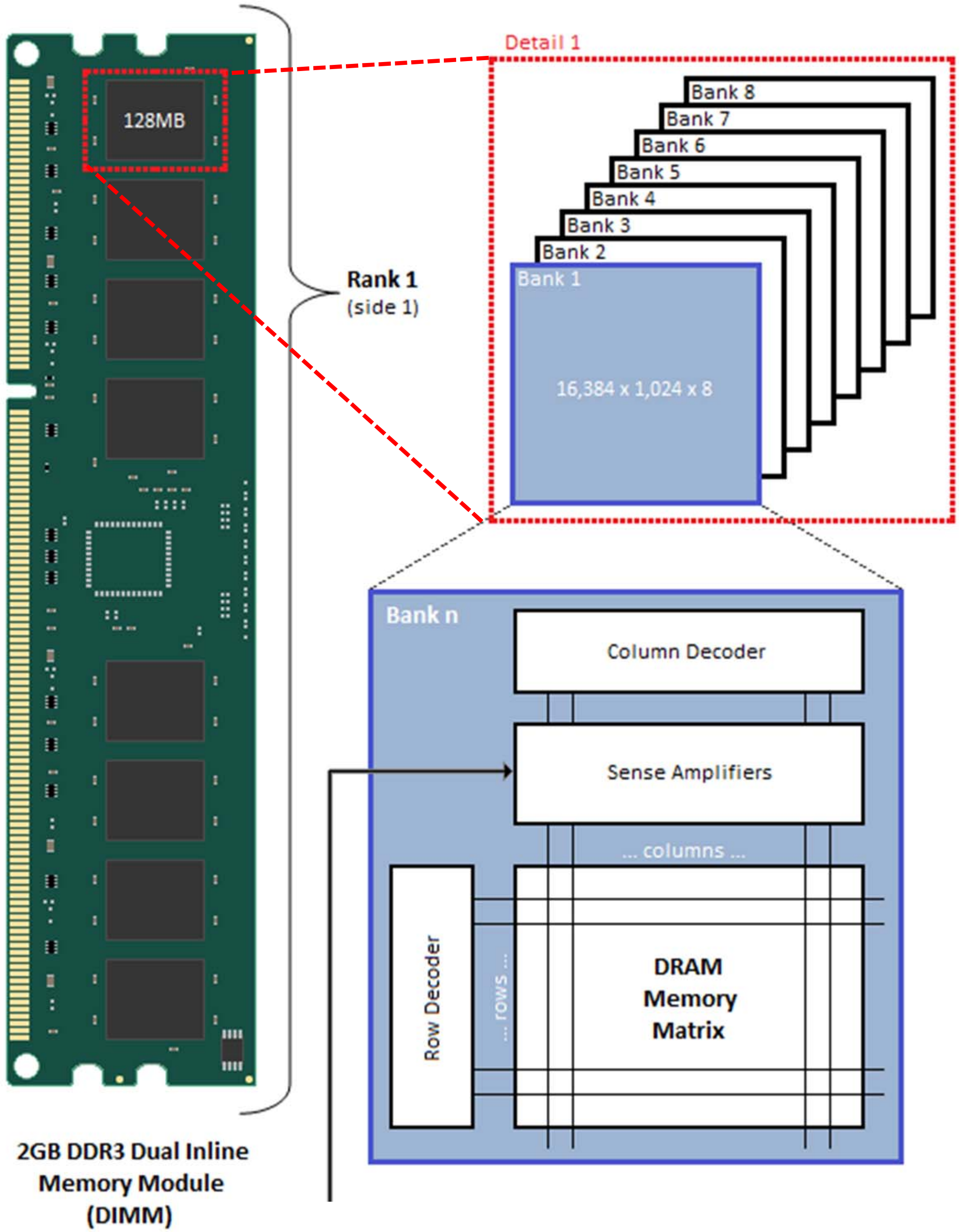
Many DRAM types...



Name	Clock rate (MHz)	BW (GB/s per DIMM)
DDR-260	133	2,1
DDR-300	150	2,4
DDR2-533	266	4,3
DDR2-800	400	6,4
DDR3-1066	533	8,5
DDR3-1600	800	12,8

UPPSALA
UNIVERSITET

Modern DRAM DIMM
Dual InLine Memory Module



From AnandTech:
Everything You Always Wanted to Know About SDRAM: But Were Afraid to Ask
http://www.anandtech.com/show/3851/everything-you-always-wanted-to-know-about-sdram-memory-but-were-afraid-to-ask

Timing ”page hit”

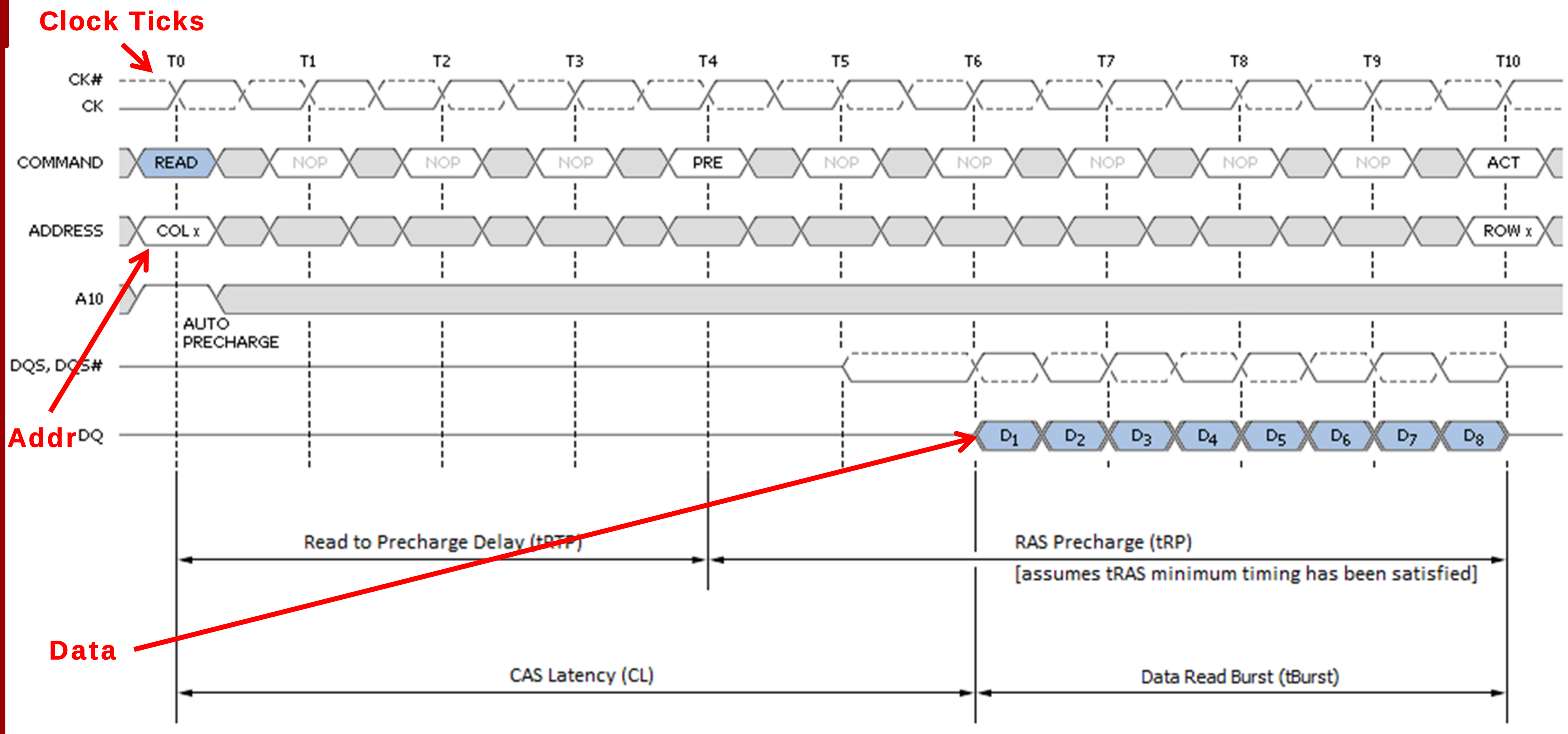


Figure 6. Page-hit timing (with precharge and subsequent bank access)

From AnandTech:
Everything You Always Wanted to Know About SDRAM: But Were Afraid to Ask
http://www.anandtech.com/show/3851/everything-you-always-wanted-to-know-about-sdram-memory-but-were-afraid-to-ask

Timing ”page miss”

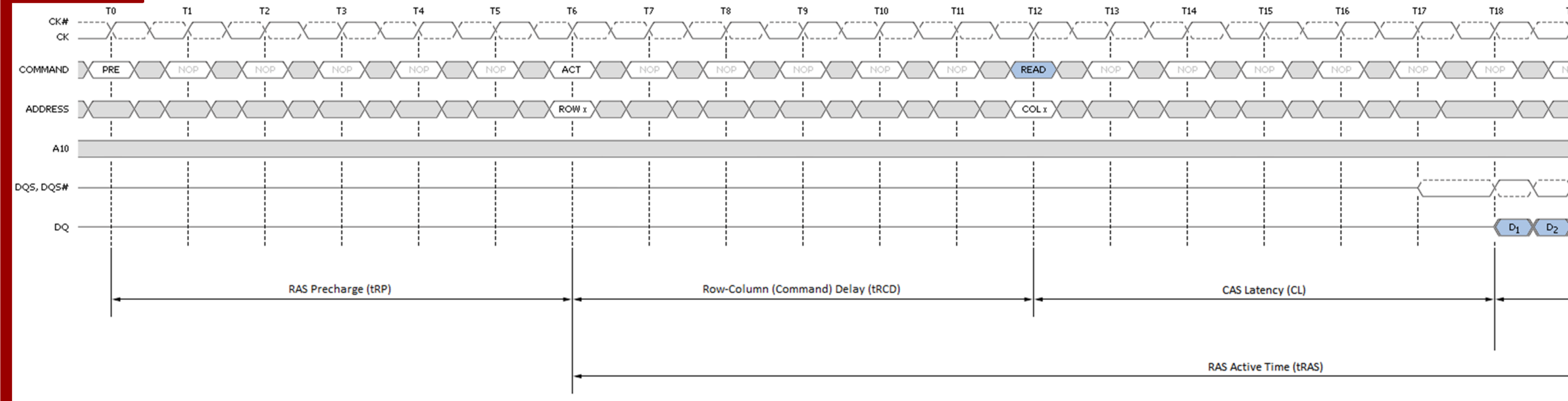


Figure 8. Page-miss timing

From AnandTech:
Everything You Always Wanted to Know About SDRAM: But Were Afraid to Ask
http://www.anandtech.com/show/3851/everything-you-always-wanted-to-know-about-sdram-memory-but-were-afraid-to-ask

UPPSALA UNIVERSITET

The Endian Mess

	Numbering the bytes	Store the value 0x5F	Store the string Hello
Big Endian	0: 0 1 2 3 / 4 5 6 7 … 64MB (msb … lsb)	0: 00 00 00 5f … 64MB (msb … lsb)	0: H e l l / o … 64MB (msb … lsb)
Little Endian	0: 3 2 1 0 / 7 6 5 4 … 64MB (msb … lsb)	0: 00 00 00 5f … 64MB (msb … lsb)	0: l l e H / o … 64MB (msb … lsb)

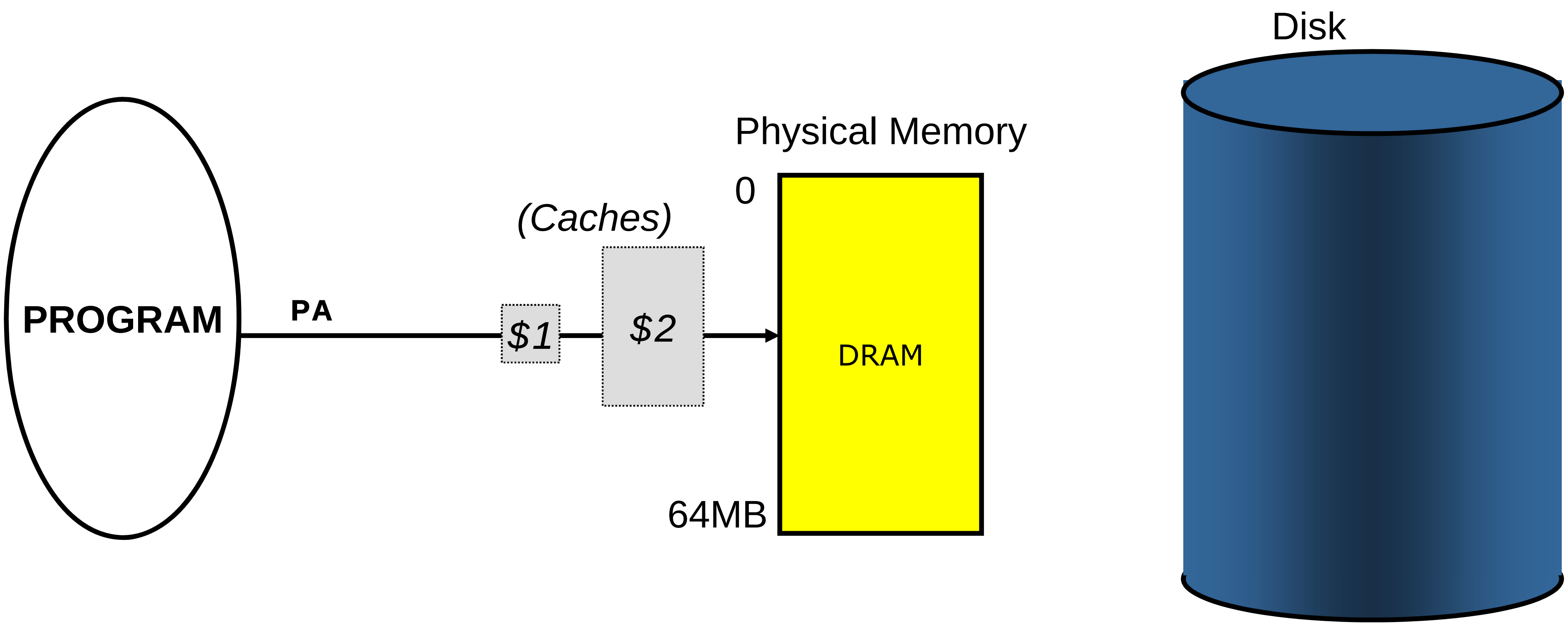
Word number

AVDARK 2013

Virtual Memory System

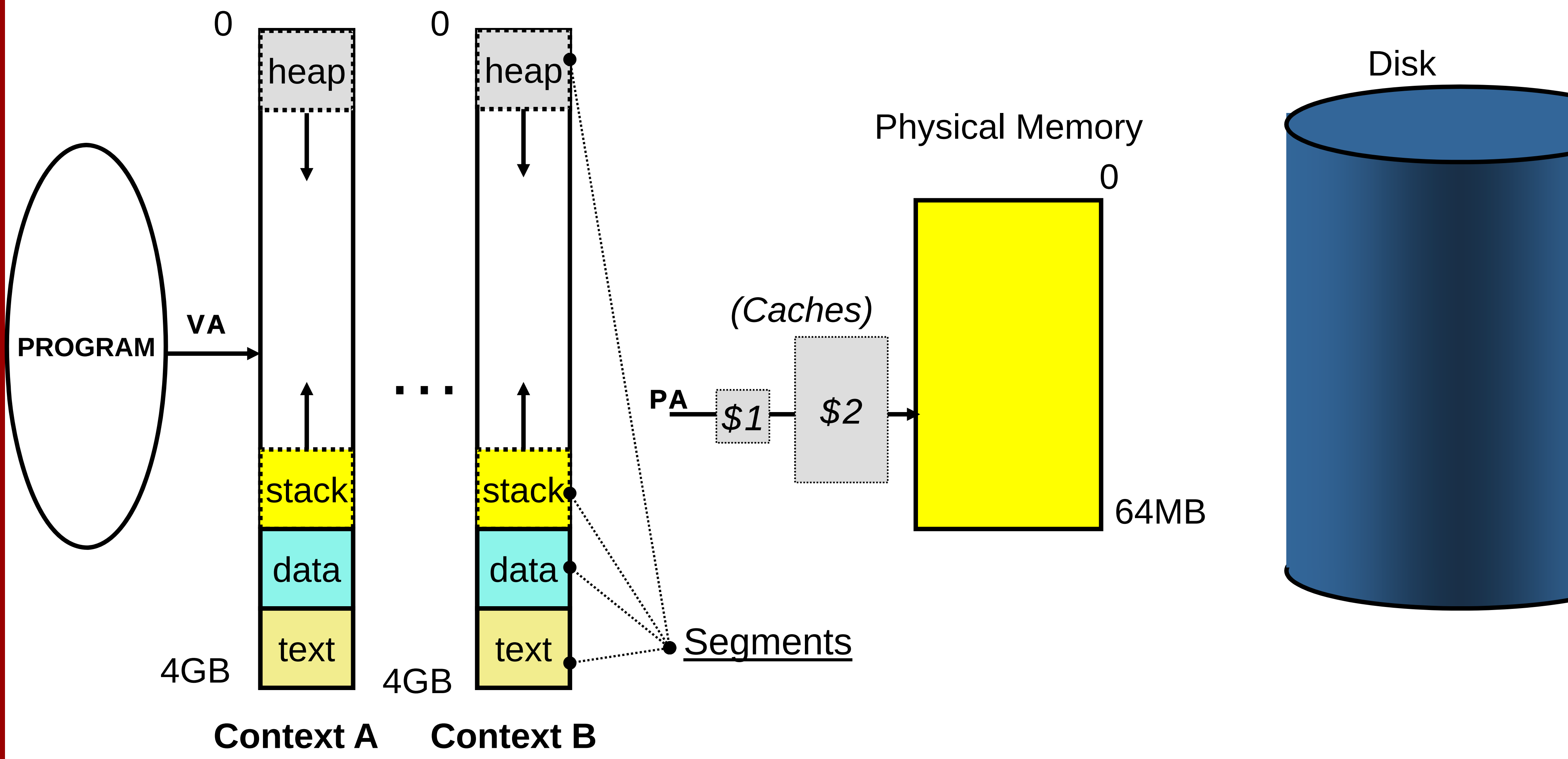
Erik Hagersten
Uppsala University, Sweden
eh@it.uu.se

Physical Memory

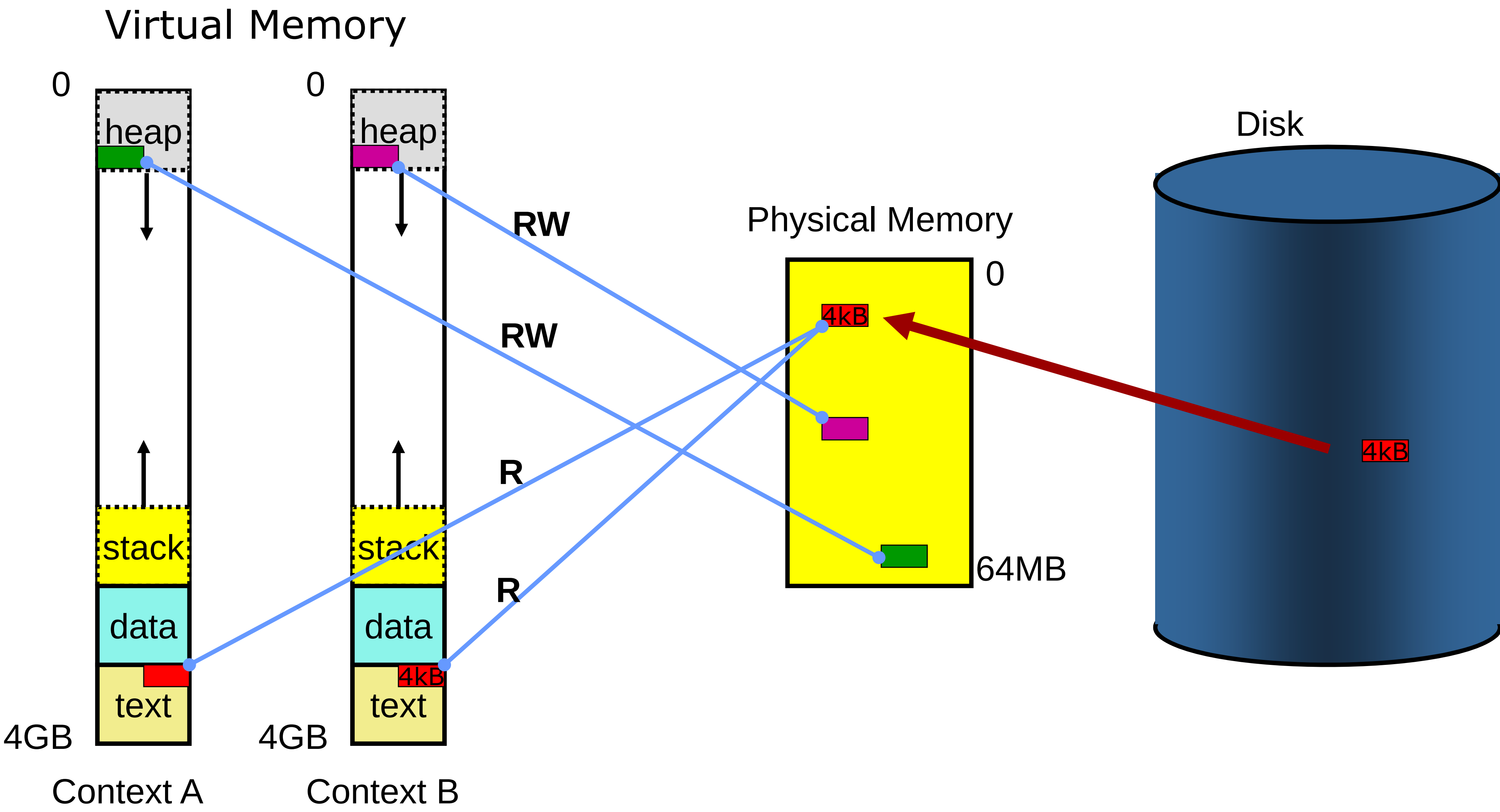


- **Requires the programmer to explicitly manage the DRAM**
- **Need to know how much memory the system has**
- **Hard to execute more than one program (process/task/...)**

Virtual and Physical Memory



Translation & Protection



VM: Block placement

Where can a block (page) be placed in main memory?

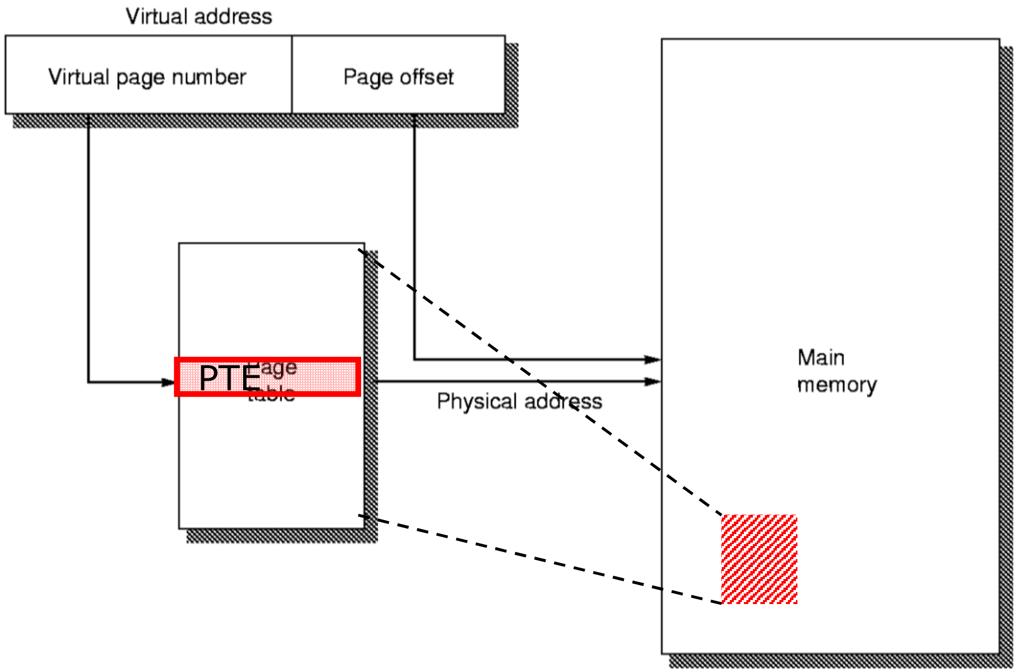
What is the organization of the VM?

- A page from disk may occupy any pageframe in DRAM (there is no ”index function”)
- The high miss penalty makes SW solutions to implement a ***fully associative address mapping*** feasible at page faults
- Some restriction can be helpful (page coloring)

- **Fully associative: Does that mean that you need to compare with a huge number of address tags?**

VM: Block identification

Use a page table stored in main memory:



Problem: Table size

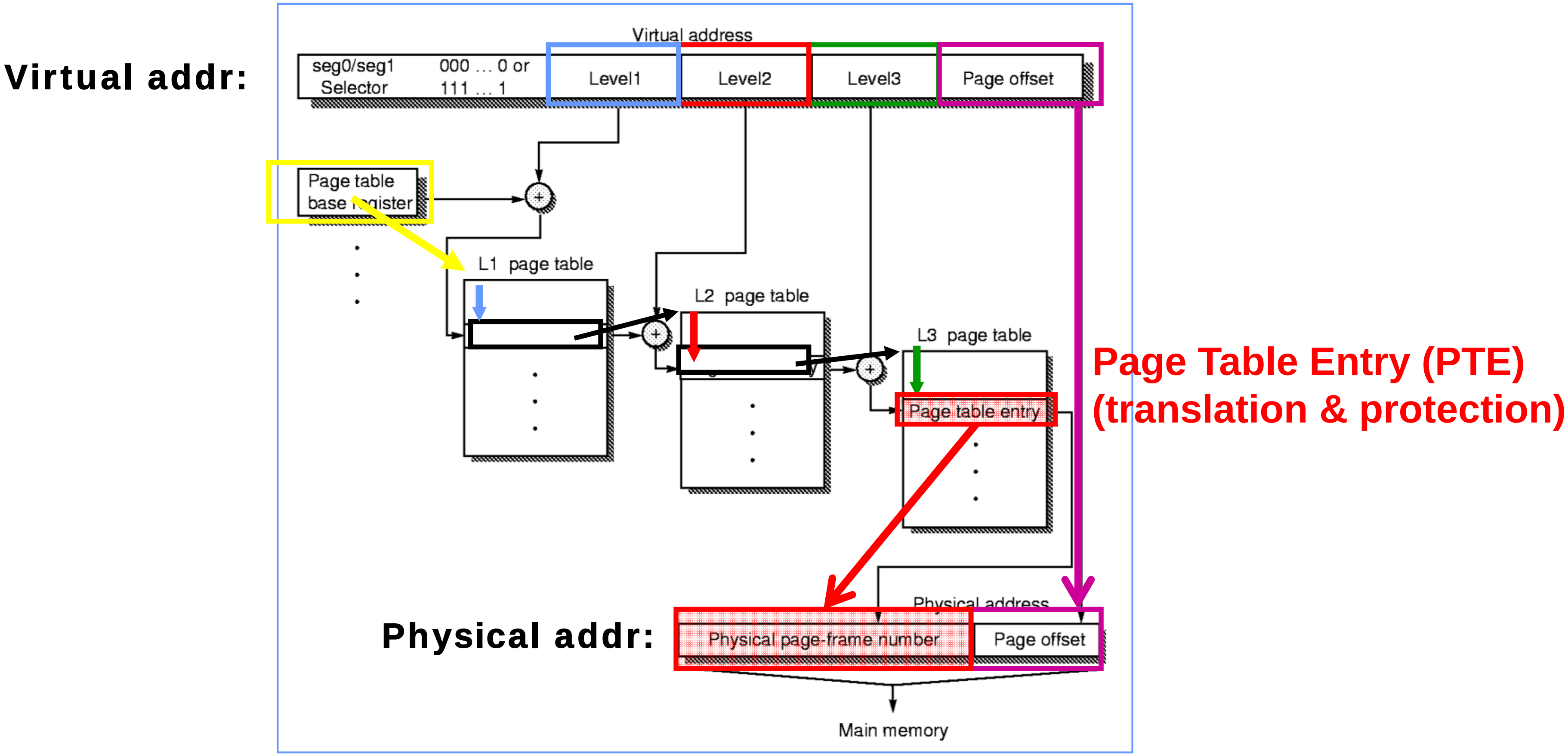
- **Suppose 4 Kbyte pages, 48 bit virtual address**
- **Page table occupies**

  $2^{48}/2^{12}$* 4B = 2^{38} = 256GB!!!

- **Solutions:**
  - ***Only one entry per <u>physical</u> page is needed***
  - **Multi-level page table (dynamic)**
  - **Inverted page table (~hashing)**

Address translation

- Multi-level table: Example: *Alpha 21064*



Protection mechanisms

The address translation mechanism can be used to provide memory protection:

- Use ***protection attribute bits*** for each page
- Stored **in the page table entry**, PTE (and TLB...)
- Each physical page gets its own **per process protection**
- **Violations** detected during the address translation **cause exceptions** (i.e., SW trap)
- ***Supervisor/user modes*** necessary to prevent user processes from changing e.g. PTEs

UPPSALA UNIVERSITET

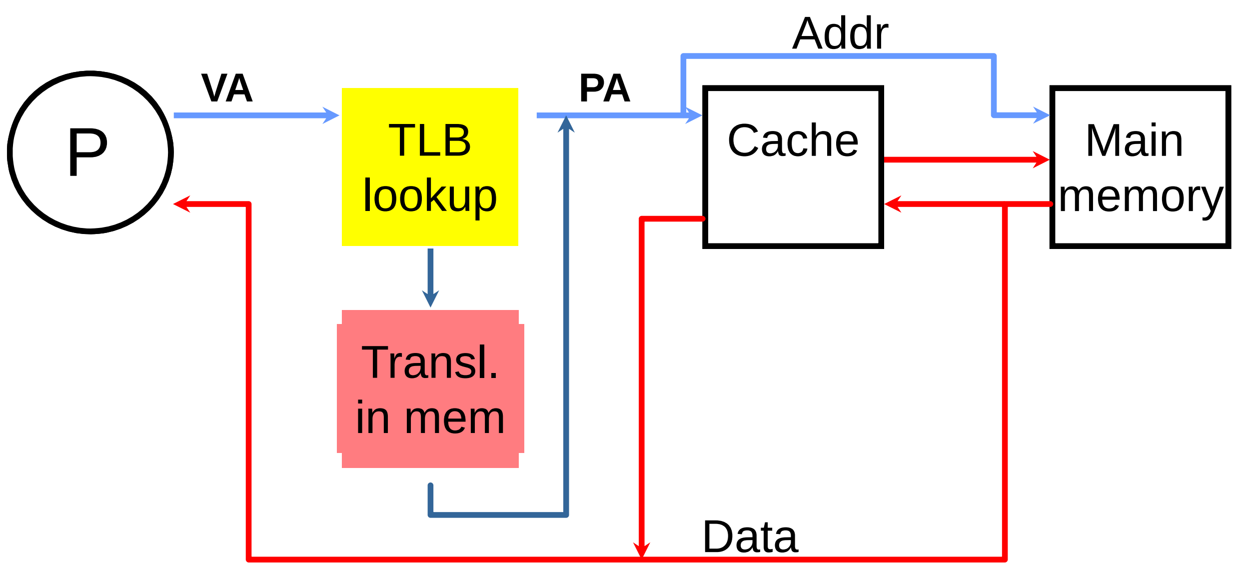
AVDARK 2013

Fast address translation

How can we avoid three extra memory references for each original memory reference?

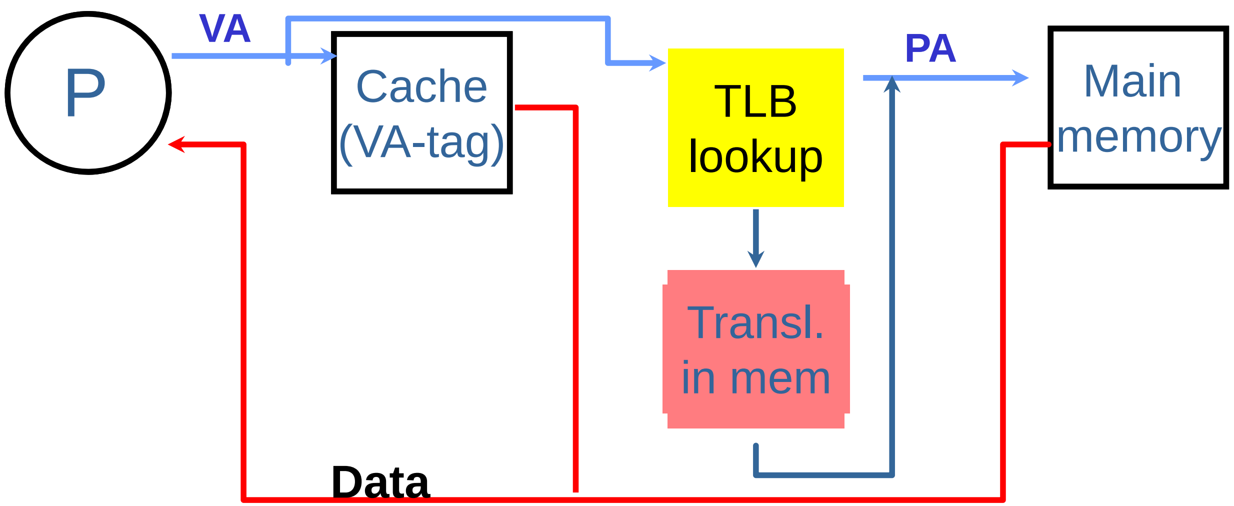
- Store the most recently used address translations in a cache—***Translation Look-aside Buffer*** (TLB)

==> The caches rears their ugly faces again!



Do we need a fast TLB?

- Why do a TLB lookup for every L1 access?
- Why not cache virtual addresses instead?
  - **Move the TLB to the other side of the cache**
  - **Physical A only used to find data in memory anyhow**
  - **The TLB can be made larger and slower.**
  - **Any problems with such virtual caches?**



Aliasing Problem with Virtual Caches

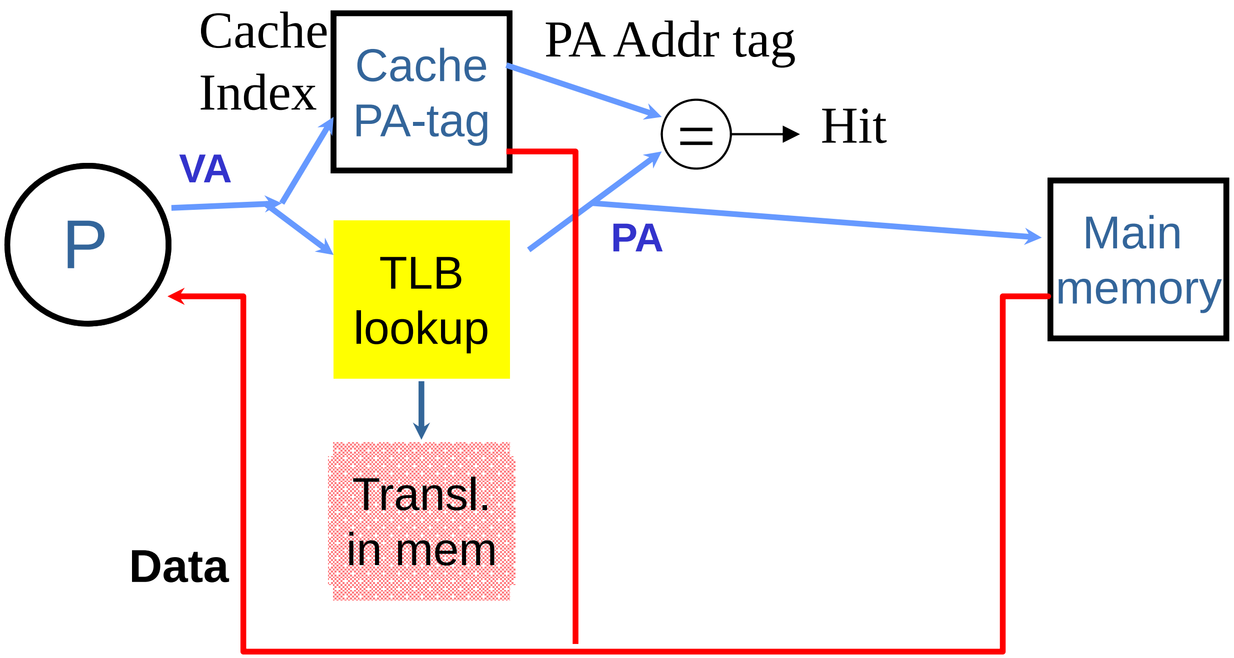
UPPSALA UNIVERSITET

The same physical page may be accessed using different virtual addresses (aliasing)

Different physical addresses may be referred to by the same virtual address (homonymes)

- A virtual cache can cause confusion – e.g., a write by one process may not be observed others
- Flushing the cache on each process switch is slow (and only helps partly)
- =>VIPT (VirtuallyIndexedPhysicallyTagged) is the answer
- (Some other solutions to handle synonymes will be discussed later)

Virtually Indexed Physically Tagged =VIPT



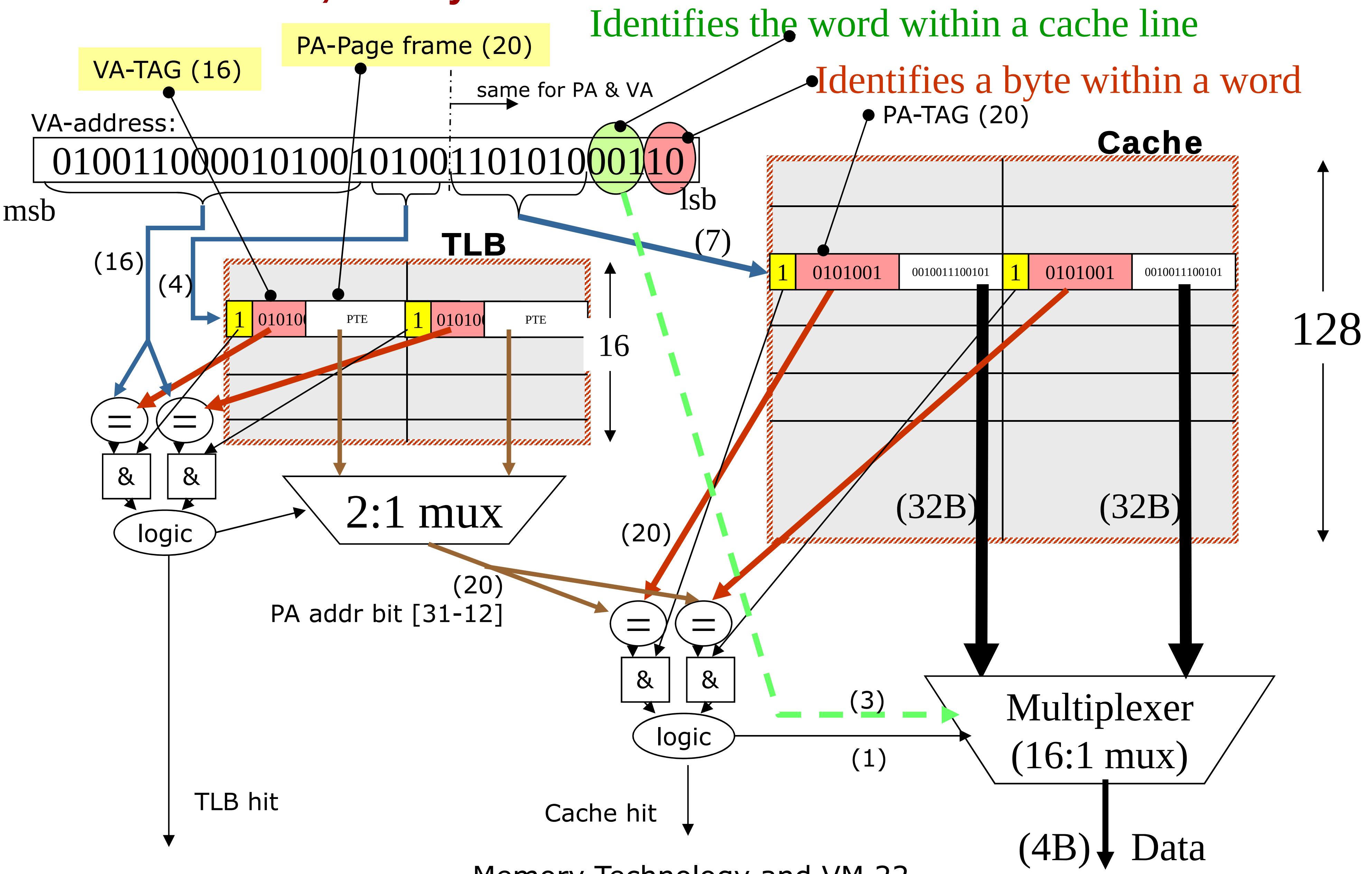
Have to guarantee that all aliases have the same index

- VIPT_cache_size ≤ (page-size * associativity)
- (Page coloring can help further)

UPPSALA UNIVERSITET

Putting it all together: VIPT

Cache: 8kB, 2-way, CL=32B, word=4B, page =4kB
TLB: 32 entries, 2-way



UPPSALA UNIVERSITET

AVDARK 2013

What is the capacity of the TLB

Typical TLB size = 0.5 - 2kB

Each translation entry 4 - 8B ==> 32 - 512 entries

Typical page size = 4kB - 16kB

TLB-reach = 0.1MB - 8MB

FIX:

- ***Multiple page sizes, e.g., 4kB and 4 MB***
- ***A hierarchy of TLB:s DTLB1, DTLB2, DTLB3…***

VM: Page replacement

Most important: ***minimize number of page faults***

Page replacement strategies:

- FIFO—First-In-First-Out
- Approximation to LRU
  - Each page has a ***reference bit*** that is set on a reference
  - The OS periodically resets the reference bits
  - When a page is replaced, a page with a reference bit that is not set is chosen

VM: Write strategy

Write back or Write through?

- ***Write back***!
- Write through is impossible to use:
  - Many small writes (ill-suited for disk)
  - Not enough bandwidth to disk

UPPSALA UNIVERSITET

VM dictionary

Virtual Memory System	The "cache" languge
Virtual address	~Cache address
Physical address	~Cache location
Page	~Huge cache block
Page fault	~Extremely painfull $miss
Page-fault handler	~The software filling the $
Page-out	Write-back if dirty

AVDARK 2013

Caches Everywhere…

- D cache
- I cache
- L2 cache
- L3 cache
- ITLB1
- ITLB2
- DTLB1
- DTLB2
- Virtual memory system
- … and more to come