Computer Architectures

Memory

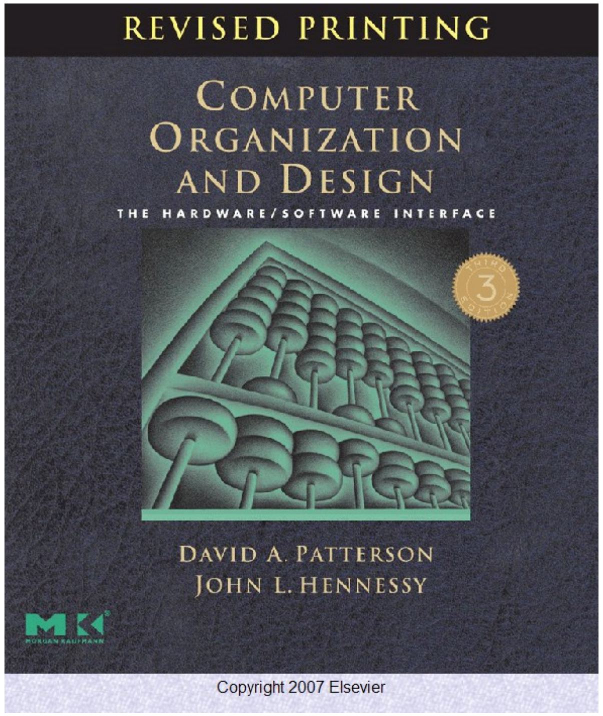
Pavel Píša, Michal Štepanovský, Miroslav Šnorek

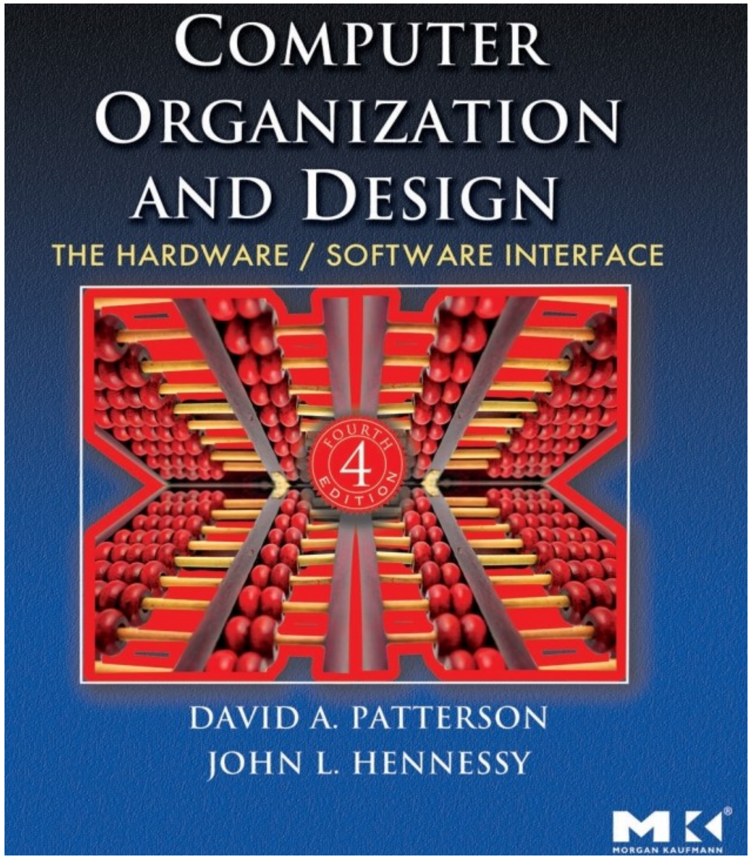
Main source of inspiration: Patterson

Czech Technical University in Prague, Faculty of Electrical Engineering

English version partially supported by:
European Social Fund Prague & EU: We invests in your future.

Suggested literature for the course





Lecture motivation

Quick Quiz 1.: Is the result of both code fragments a same?

Quick Quiz 2.: Which of the code fragments is processed faster and why?

A:

```
int matrix[M][N];
int i, j, sum = 0;
…
for(i=0; i<M; i++)
  for(j=0; j<N; j++)
    sum += matrix[i][j];
```

B:

```
int matrix[M][N];
int i, j, sum = 0;
…
for(j=0; j<N; j++)
  for(i=0; i<M; i++)
    sum += matrix[i][j];
```

Is there a rule how to iterate over matrix element efficiently?

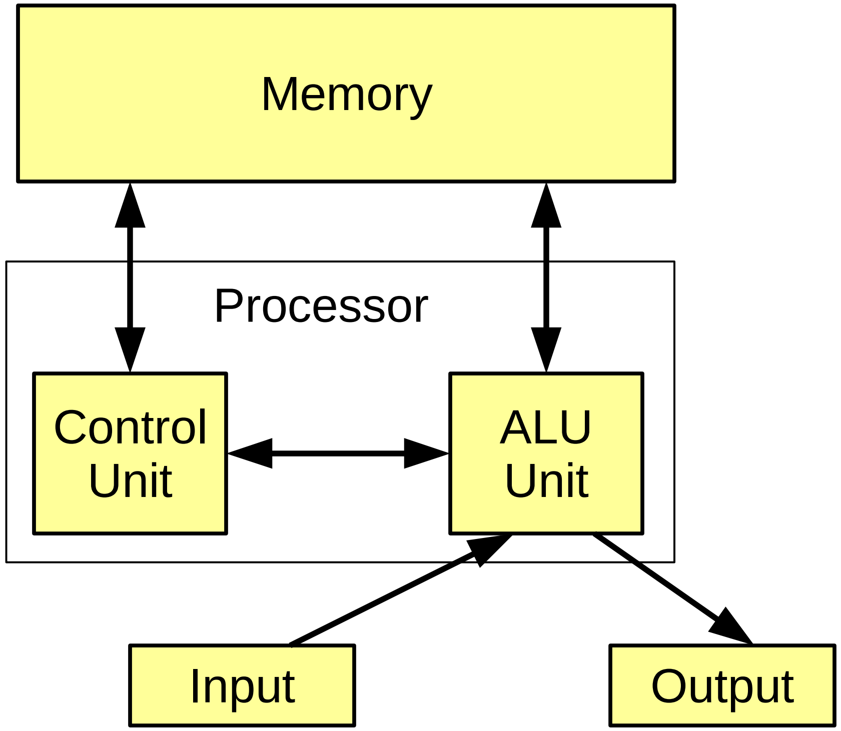
Lecture outline

- Overview of memory related terms and definitions
- Memory hierarchy
  - Management and mapping of data between levels
- Cache memory
  - Basic concept
  - More realistic approach
  - Multi-level cache memory
- Virtual memory
- Memory hierarchy and related problems
- Main memory implementation – memory chips
- Secondary(+more) storage (mass storage)

John von Neumann, Hungarian physicist

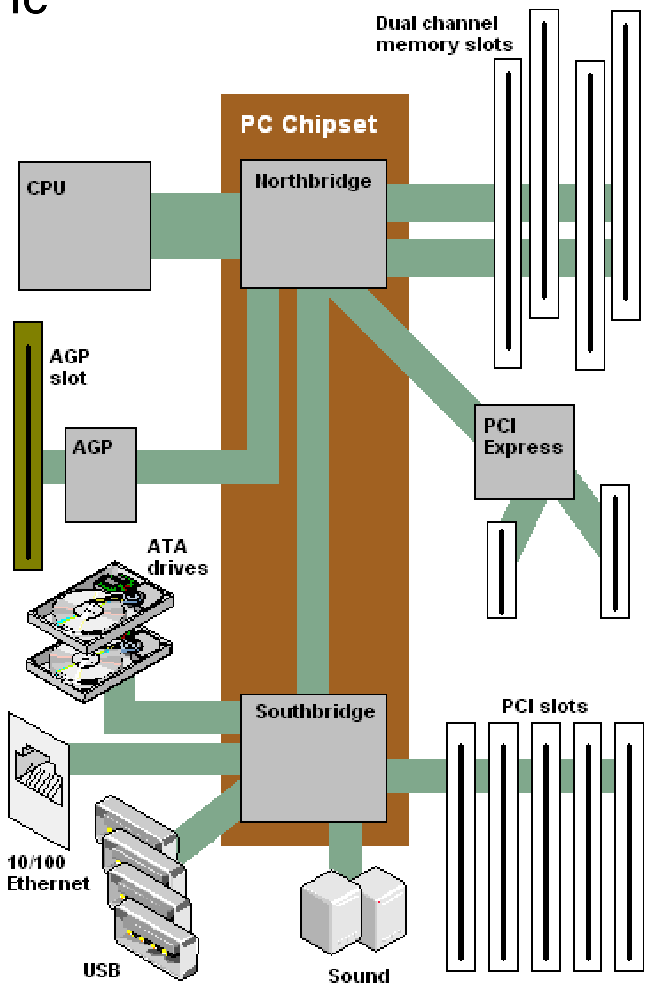
von Neumann's computer architecture

28. 12. 1903 - 8. 2. 1957

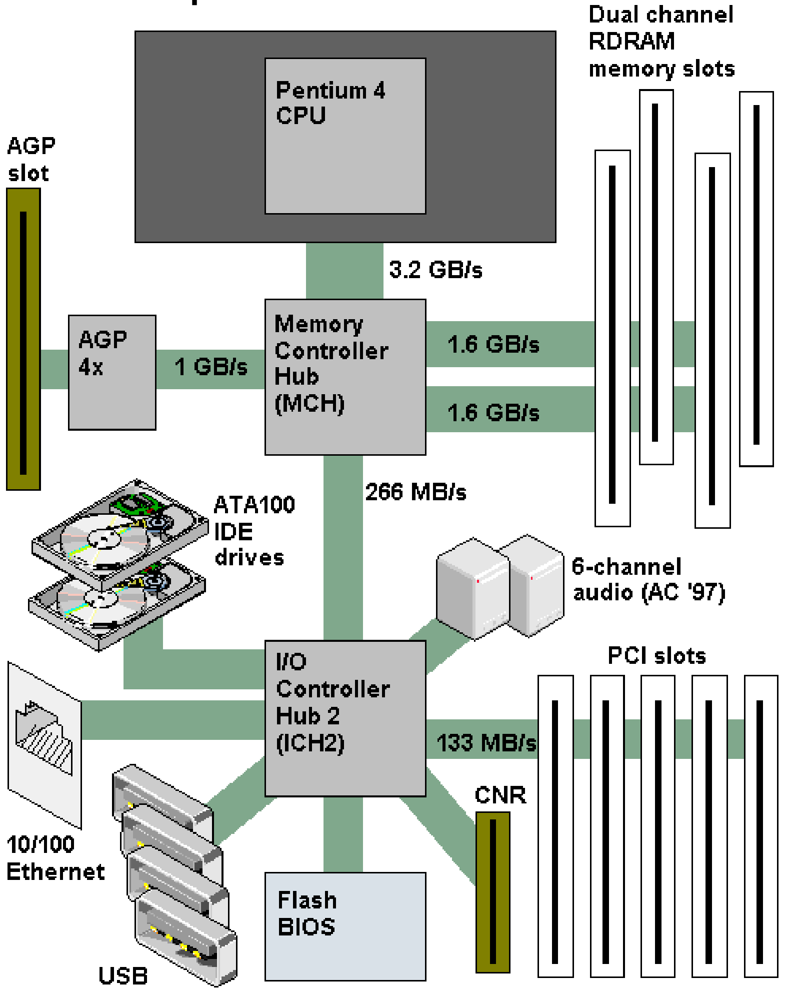


Computer architecture (desktop x86 PC)

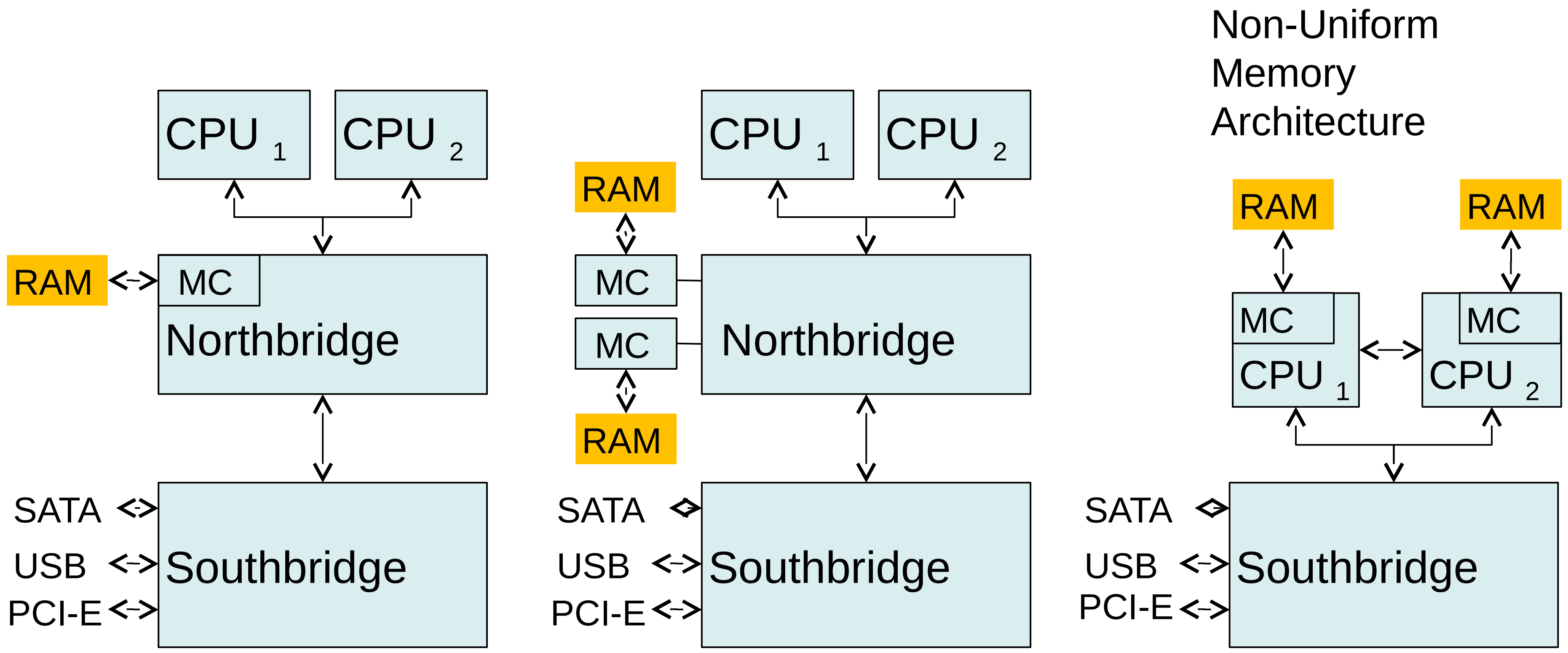
generic



example

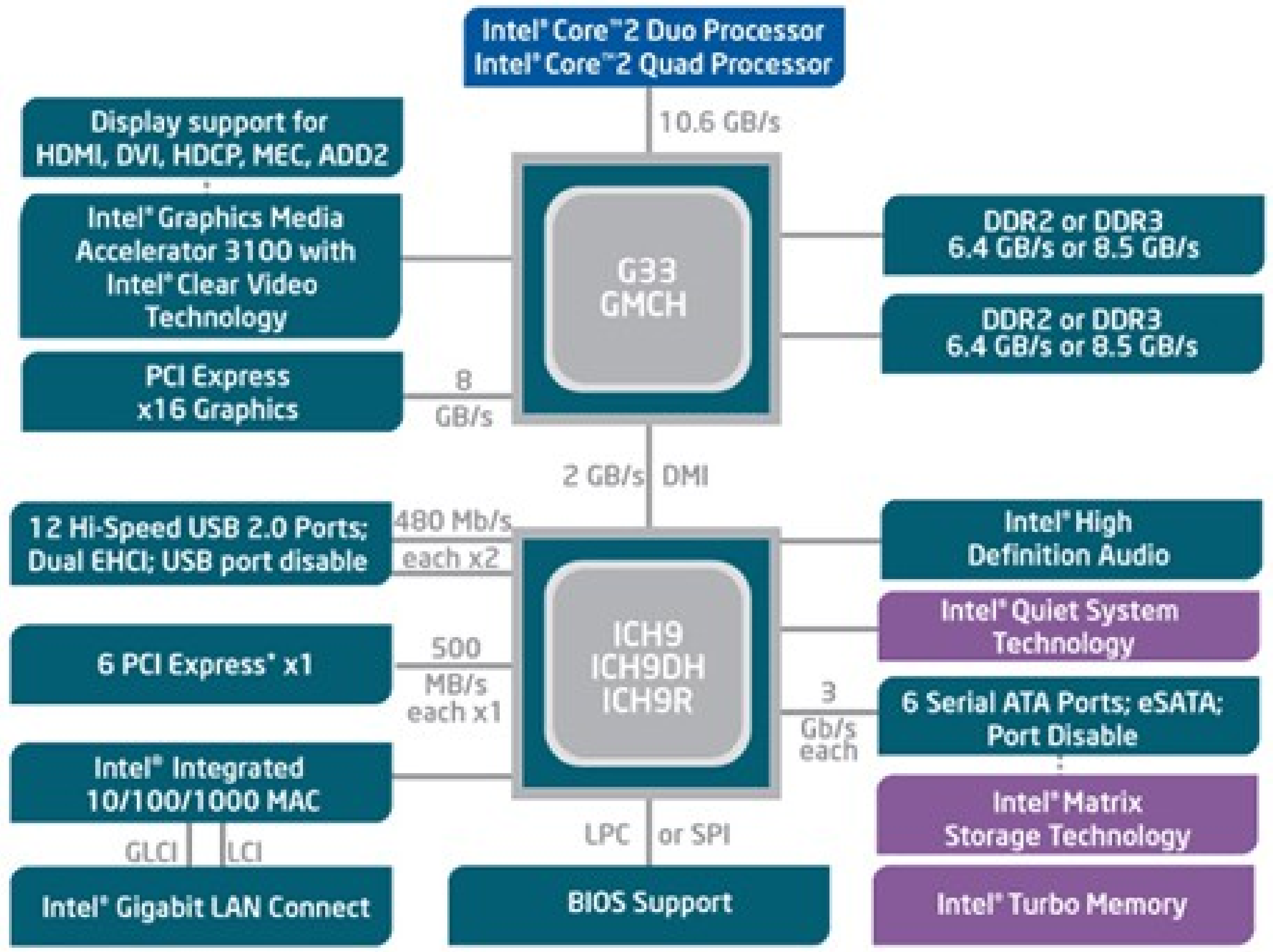


From UMA to NUMA development (even in PC segment)



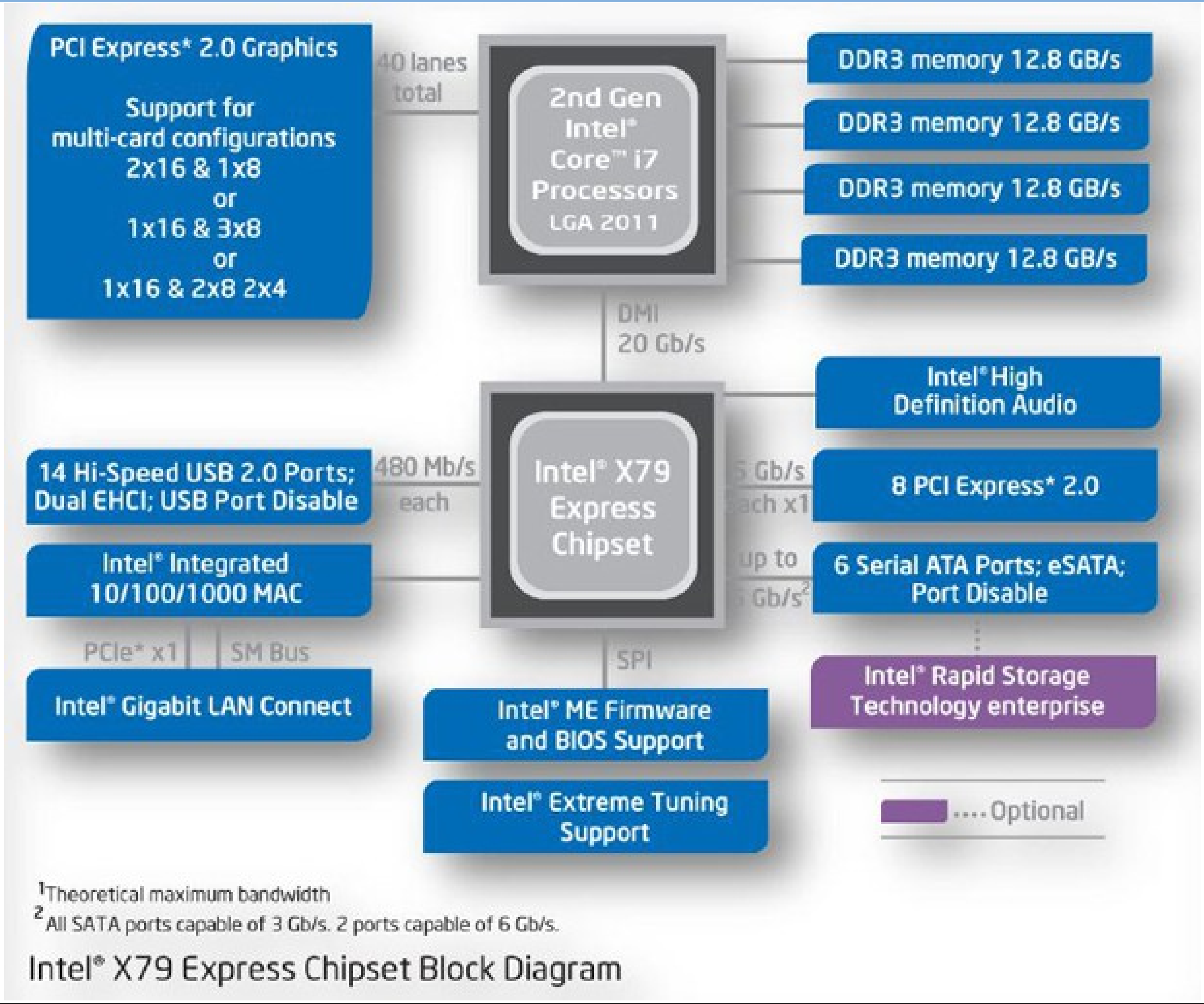
MC - Memory controller – contains circuitry responsible for SDRAM read and writes. It also takes care of refreshing each memory cell every 64 ms.

Intel Core 2 generation


Intel® Core™2 Duo Processor
Intel® Core™2 Quad Processor
10.6 GB/s
Display support for HDMI, DVI, HDCP, MEC, ADD2
Intel® Graphics Media Accelerator 3100 with Intel® Clear Video Technology
PCI Express x16 Graphics
8 GB/s
G33 GMCH
DDR2 or DDR3 6.4 GB/s or 8.5 GB/s
DDR2 or DDR3 6.4 GB/s or 8.5 GB/s
2 GB/s DMI
12 Hi-Speed USB 2.0 Ports; Dual EHCI; USB port disable
480 Mb/s each x2
6 PCI Express® x1
500 MB/s each x1
Intel® Integrated 10/100/1000 MAC
GLCI
LCI
Intel® Gigabit LAN Connect
ICH9 ICH9DH ICH9R
LPC or SPI
BIOS Support
Intel® High Definition Audio
Intel® Quiet System Technology
3 Gb/s each
6 Serial ATA Ports; eSATA; Port Disable
Intel® Matrix Storage Technology
Intel® Turbo Memory

Northbridge became Graphics and Memory Controller Hub (GMCH)

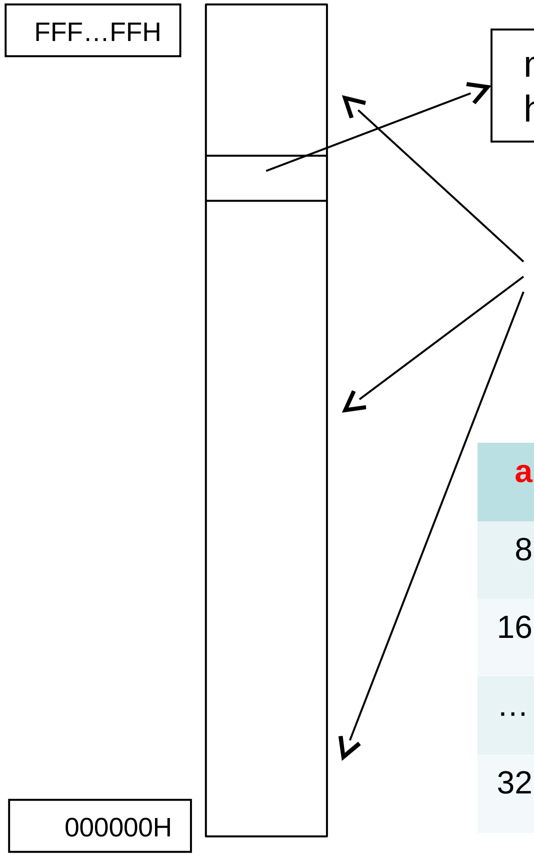
Intel i3/5/7 generation

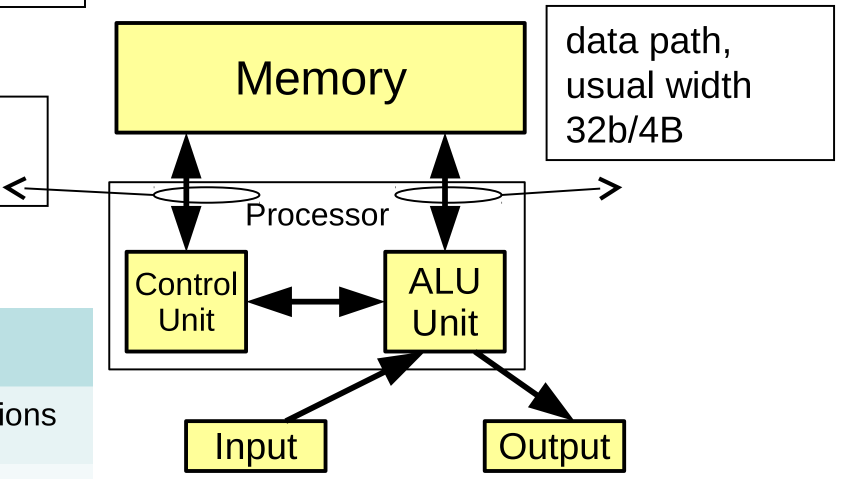

PCI Express* 2.0 Graphics
Support for multi-card configurations 2x16 & 1x8 or 1x16 & 3x8 or 1x16 & 2x8 2x4
40 lanes total
2nd Gen Intel® Core™ i7 Processors LGA 2011
DDR3 memory 12.8 GB/s
DDR3 memory 12.8 GB/s
DDR3 memory 12.8 GB/s
DDR3 memory 12.8 GB/s
DMI 20 Gb/s
Intel® High Definition Audio
14 Hi-Speed USB 2.0 Ports; Dual EHCI; USB Port Disable
480 Mb/s each
Intel® X79 Express Chipset
5 Gb/s each x1
8 PCI Express* 2.0
Intel® Integrated 10/100/1000 MAC
up to 6 Gb/s²
6 Serial ATA Ports; eSATA; Port Disable
PCIe* x1
SM Bus
SPI
Intel® Rapid Storage Technology enterprise
Intel® Gigabit LAN Connect
Intel® ME Firmware and BIOS Support
Intel® Extreme Tuning Support
.... Optional
¹Theoretical maximum bandwidth
²All SATA ports capable of 3 Gb/s. 2 ports capable of 6 Gb/s.

Intel® X79 Express Chipset Block Diagram

Memory address space

It is an array of addressable units (locations) where each unit can hold a data value.
Number/range of addresses same as addressable units/words are limited in size.





a	2↑a
8	256 distinct locations
16	64K (K=1024)
…	……
32	4G (4096M, M=K↑2)

The most common size of addressable memory unit is 1B (8 bits)

Memory subsystem – terms and definitions

- Memory address – fixed-length sequences of bits or index
- Data value – the visible content of a memory location

  Memory location can hold even more control/bookkeeping information
  - validity flag, parity and ECC bits etc.

- Basic memory parameters:
  - Access time – delay or latency between a request and the access being completed or the requested data returned
  - Memory latency – time between request and data being available (does not include time required for refresh and deactivation)
  - Throughput/bandwidth – main performance indicator. Rate of transferred data units per time.
  - Maximal, average and other latency parameters

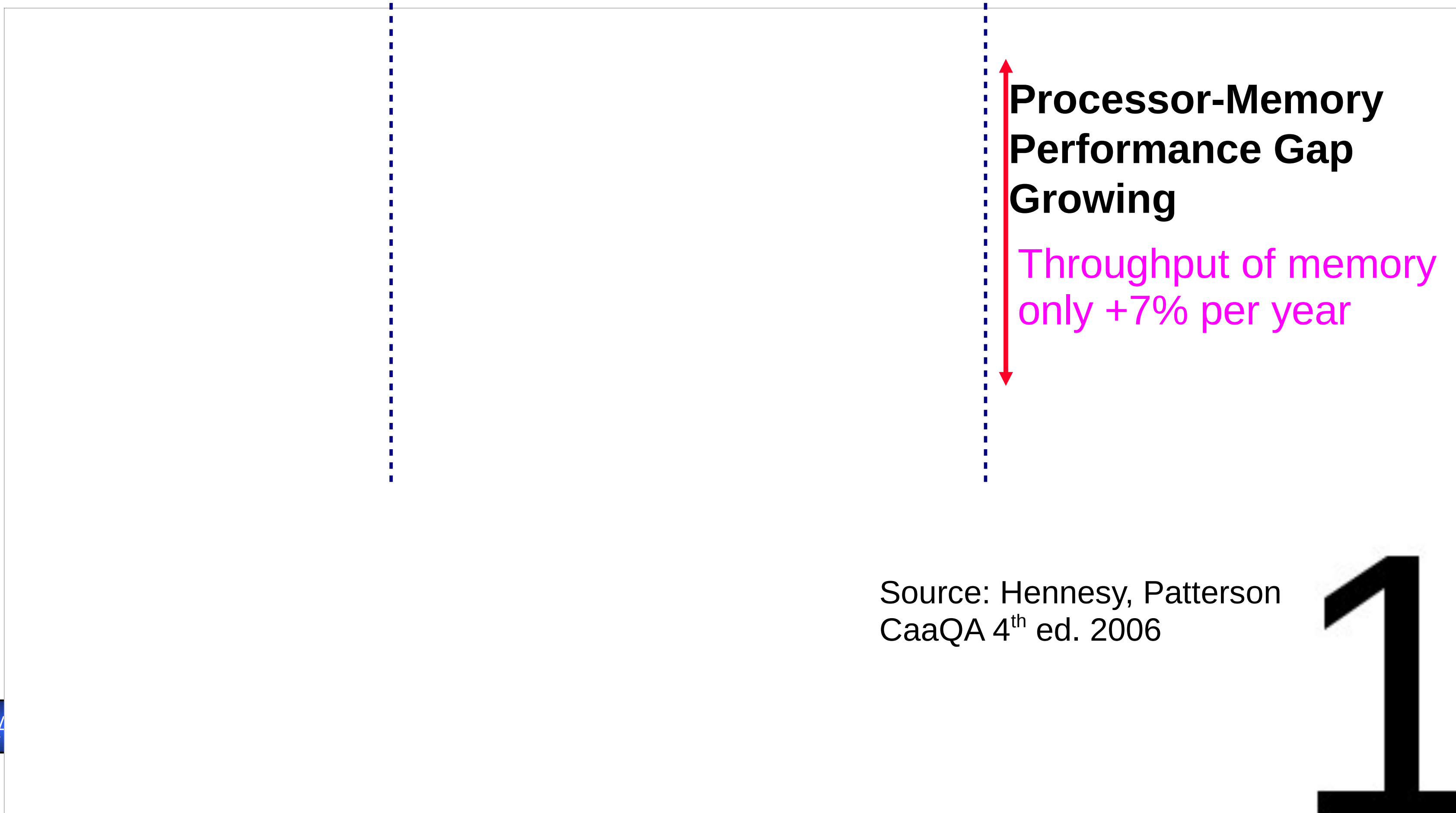
Memory types and maintenance

- Types: RWM (RAM), ROM, FLASH
- Implementation: SRAM, DRAM

- Data retention time and conditions (volatile/nonvolatile)
- Dynamic memories (DRAM, SDRAM) require specific care
  - Memory refresh – state of each memory cell has to be internally read, amplified and fed back to the cell once every refresh period (usually about 60 ms), even in idle state. Each refresh cycle processes one row of cells.
  - Precharge – necessary phase of access cycle to restore cell state after its partial discharge by read
  - Both contribute to maximal and average access time.

Memory and CPU speed – Moore's law

CPU performance | 25% per year | 52% per year | 20% per year



Processor-Memory Performance Gap Growing

Throughput of memory only +7% per year

Source: Hennesy, Patterson CaaQA 4th ed. 2006

Bubble sort – algorithm example from seminaries

```
int pole[5]={5,3,4,1,2};
int main()
{
    int N = 5,i,j,tmp;
    for(i=0; i<N; i++)
        for(j=0; j<N-1-i; j++)
            if(pole[j+1]<pole[j])
            {
                tmp = pole[j+1];
                pole[j+1] = pole[j];
                pole[j] = tmp;
            }
    return 0;
}
```

What we can consider and expect from our programs?

Think about some typical data access patterns and execution flow.

Memory hierarchy – principle of locality

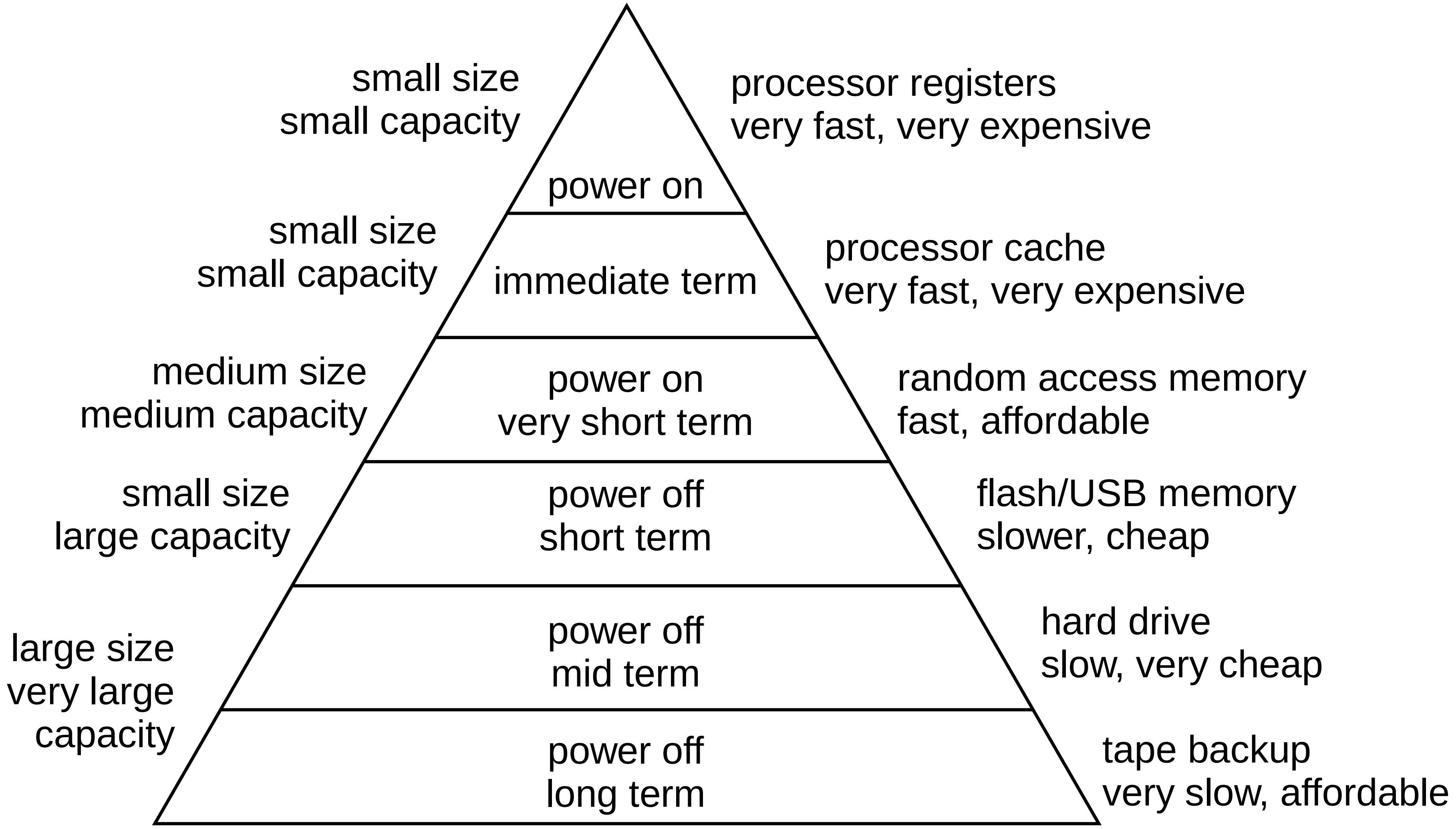
- Programs access a small proportion of their address space at any time
- Temporal locality
  - Items accessed recently are likely to be accessed again soon
  - e.g., instructions in a loop, induction variables
- Spatial locality
  - Items near those accessed recently are likely to be accessed soon
  - E.g., sequential instruction access, array data

Source: Hennesy, Patterson

Memory hierarchy introduced based on locality

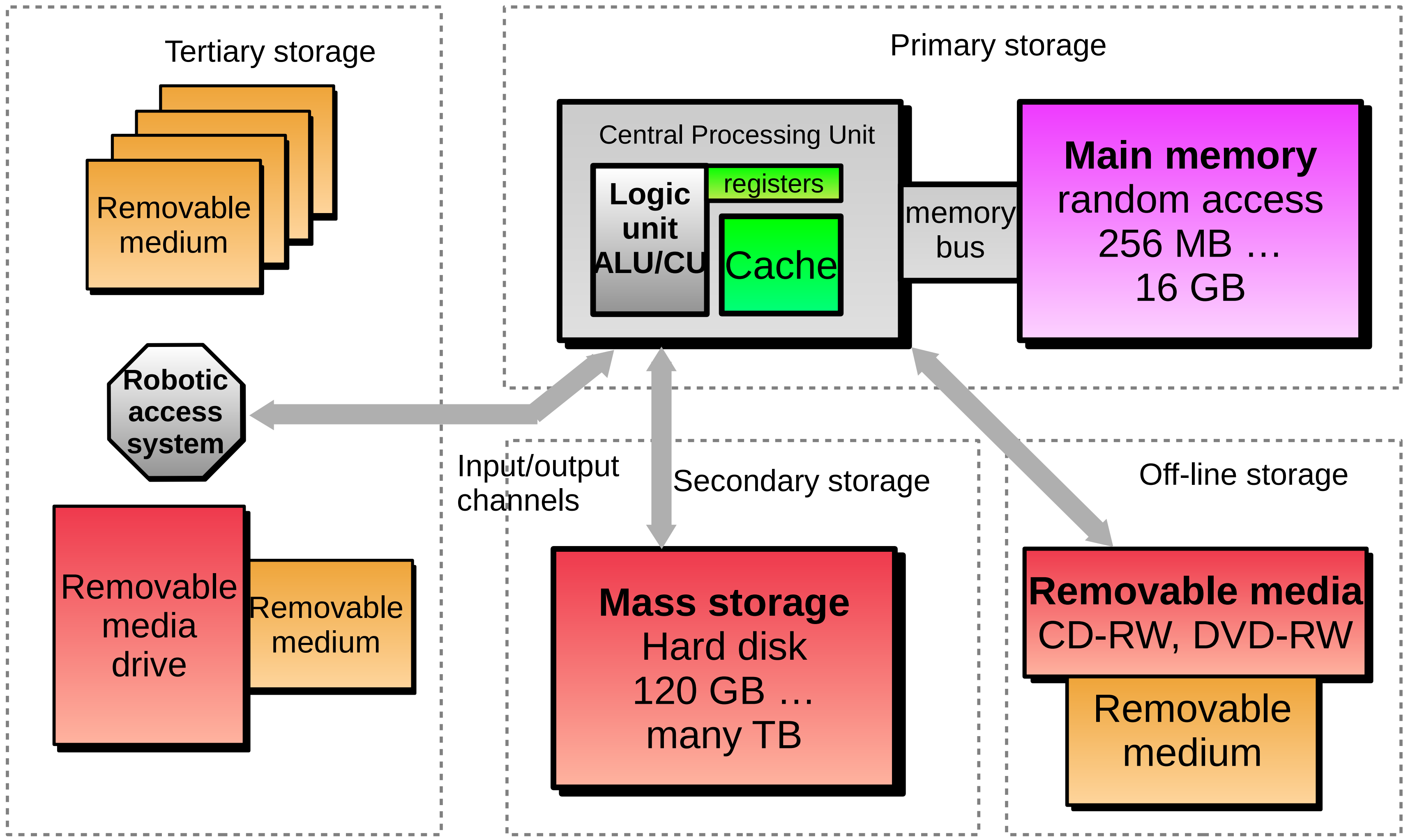
- The solution to resolve capacity and speed requirements is to build address space (data storage in general) as hierarchy of different technologies.
- Store input/output data, program code and its runtime data on large and cheaper secondary storage (hard disk)
- Copy recently accessed (and nearby) items from disk to smaller DRAM based main memory (usually under operating system control)
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory (cache) attached to CPU (hidden memory, transactions under HW control), optionally, tightly coupled memory under program's control
- Move currently processed variables to CPU registers (under machine program/compiler control)

Memory hierarchy – speed, capacity, price



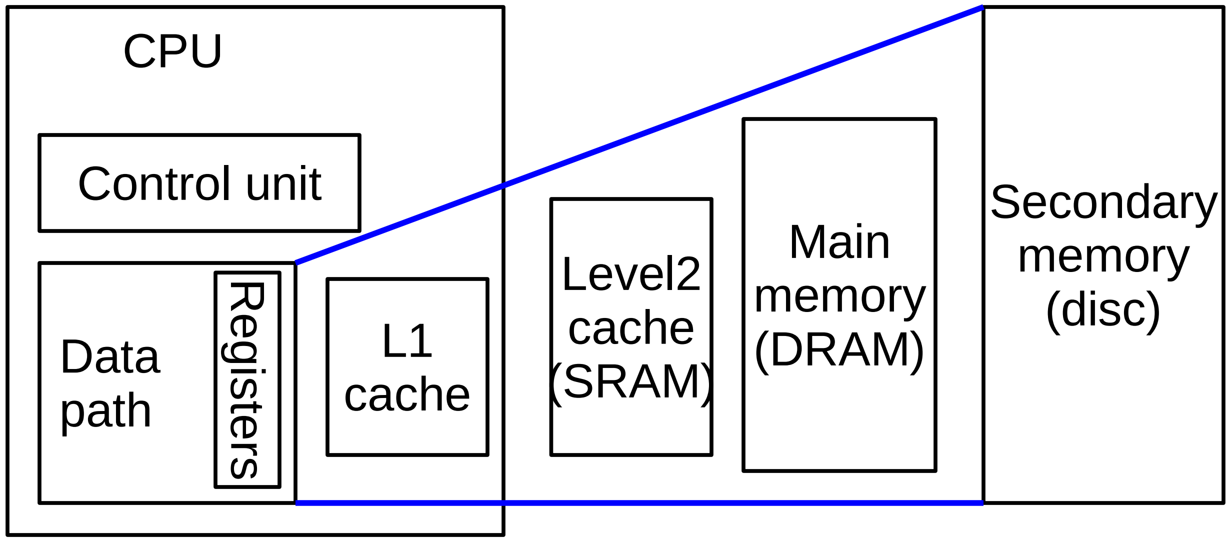
Source: Wikipedia.org

Memory/storage in computer system



Source: Wikipedia.org

Contemporary price/size examples



Type/ Size	L1 32kB	Sync SRAM	DDR3 16 GB	HDD 3TB
Price	10 kč/kB	300 kč/MB	123 kč/GB	1 kč/GB
Speed/ throughput	0.2...2ns	0.5...8 ns/word	15 GB/sec	100 MB/sec

Some data can be available in more copies (consider levels and/or SMP).
Mechanisms to keep consistency required if data are modified.

Mechanism to lookup demanded information?

- According to the address and other management information (data validity flags etc).
- The lookup starts at the most closely located memory level (local CPU L1 cache).
- Requirements:
  - Memory consistency/coherency.
- Used means:
  - Memory management unit to translate virtual address to physical and signal missing data on given level.
  - Mechanisms to free (swap) memory locations and migrate data between hierarchy levels
- Hit (data located in upper level – fast), miss (copy from lower level required)

Processor-memory performance gap solution – cache

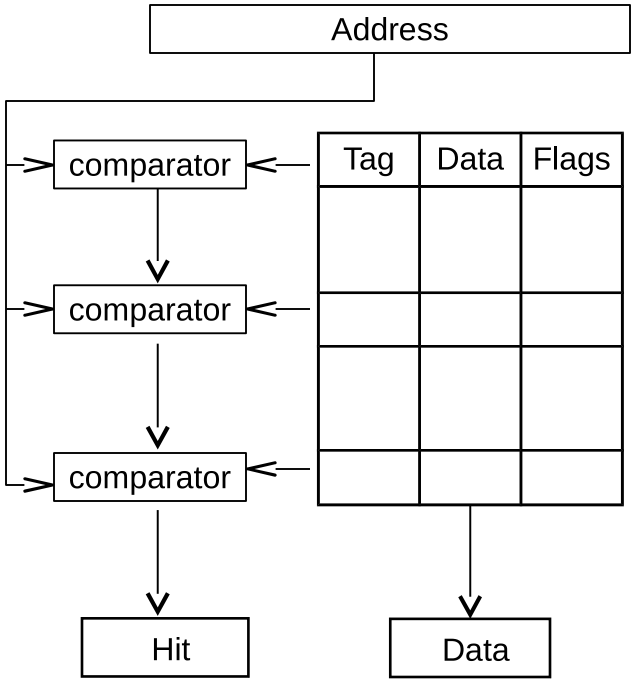
Performance gap between CPU and main memory

- Solution – **cache** memory
- Cache – component that (transparently) stores data so that future requests for that data can be served faster
- Transparent cache – hidden memory

- Placed between two subsystems with different data throughput. It speeds-up access to (recently) used data.
- This is achieved by maintaining copy of data on memory device faster than the original storage

Initial idea – fully associative cache

- **Tag** – the key to locate data (value) in the cache. The original address in the main memory for fully associative case. Size of this field is given by number of bits in an address – i.e. 32, 48 or 64
- **Data** – the stored information, basic unit – word – is usually 4 bytes
- **Flags** – additional bits to keep service information.

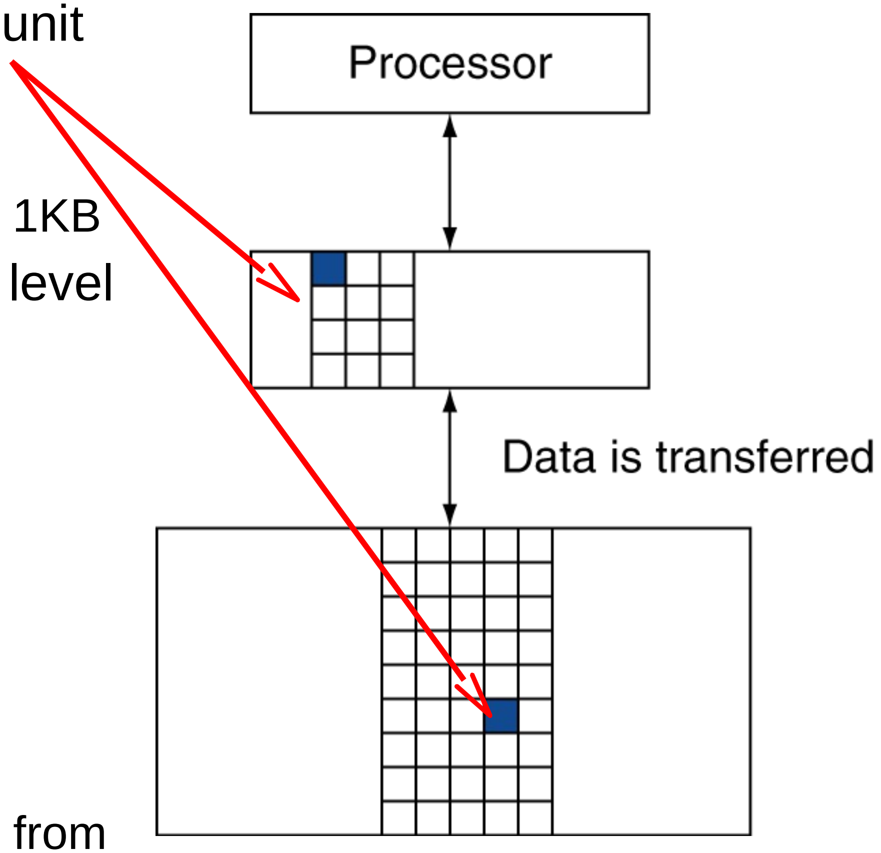


Cache line of fully associative cache

Tag	**Data**	**Flags**

Definitions for cache memory

- **Cache line** or **cache block** – basic unit copied between levels
  - May be multiple words
  - Usual cache line size from 8B up to 1KB
- If accessed data is present in upper level
  - **Hit**: access satisfied by upper level
    - **Hit rate**: hits/accesses
- If accessed data is absent
  - **Miss**: block copied from lower level
    - Time taken: **miss penalty**
    - **Miss rate**: misses/accesses
      = 1 – hit rate
  - Then the accessed data is supplied from upper level



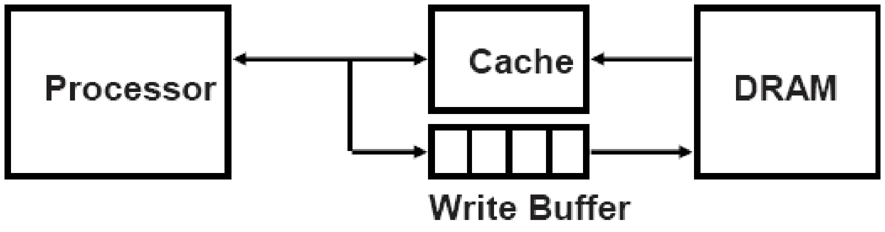
Fully associative cache implementation

- The **Tag** field width is equivalent to address width (not counting address bits equivalent to byte in word or line)
- Each cache line requires its own multiplexer input and same number of one-bit comparators as is size of the tag field.
- Cache line count determines capacity of the cache
- Cache requires complex replacement policy logic to find out which of all lines is the best candidate for new request.

- Such cache implementation is very expensive to implement in HW (ratio of gate count/capacity is high) and slow
- That is why other cache types are used in practice
  - Direct mapped
  - n-way associative – with limited associativity

CPU writes to main memory

- There is cache in the way
- **Data consistency** – requirement for data coherency for same address accessed through different paths
- **Write through** – data are written to the cache and write buffer/queue simultaneously
- **Write back** – data are written to the cache only and dirty bit is set. Write to the other level is delayed until cache line replacement time or to cache flush event
- **Dirty bit** – an additional flag for cache line. It Indicates that cached value is updated and does not correspond with the main memory.



V	Other bits, i.e. D	Tag	Data

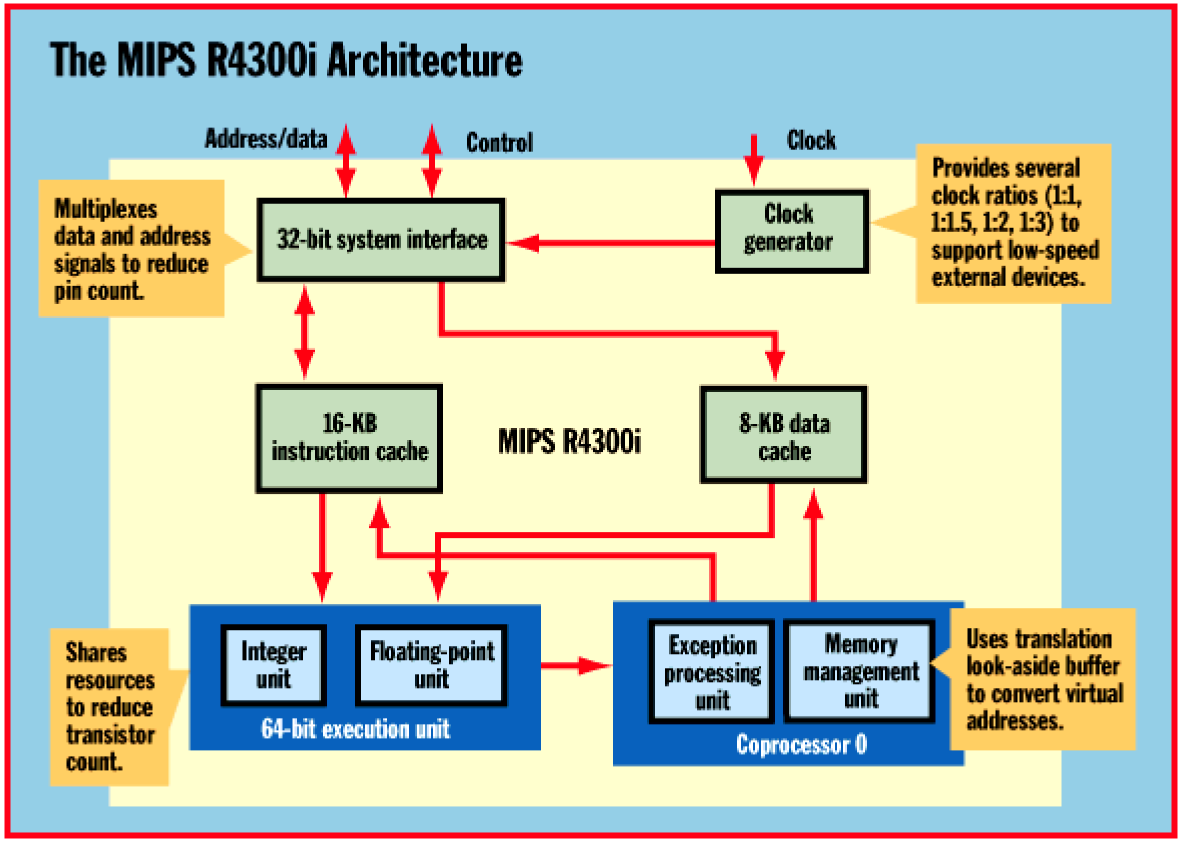
The process to resolve **cache miss**

- Data has to be filled from main memory, but quite often all available cache locations which address can be mapped to are allocated

- **Cache content replacement policy** (offending cache line is invalidated either immediately or after data are placed in the write queue/buffer)
- **Random** – random cache line is evicted
- **LRU** (Least Recently Used) – additional information is required to find cache line that has not been used for the longest time
- **LFU** (Least Frequently Used) – additional information is required to find cache line that is used least frequently – requires some kind of forgetting
- **ARC** (Adaptive Replacement Cache) – combination of LRU and LFU concepts
- **Write-back** – content of the modified (dirty) cache line is moved to the write queue

CPU including cache – Harvard cache architecture

Separated instruction and data cache
The concept of Von Neumann's CPU with Harvard cache is illustrated on a MIPS CPU family member, i.e. real CPU which is superset of the design introduced during lectures 2 and 4.


The MIPS R4300i Architecture
Address/data
Control
Clock
Multiplexes data and address signals to reduce pin count.
32-bit system interface
Clock generator
Provides several clock ratios (1:1, 1:1.5, 1:2, 1:3) to support low-speed external devices.
16-KB instruction cache
MIPS R4300i
8-KB data cache
Shares resources to reduce transistor count.
Integer unit
Floating-point unit
64-bit execution unit
Exception processing unit
Memory management unit
Coprocessor 0
Uses translation look-aside buffer to convert virtual addresses.

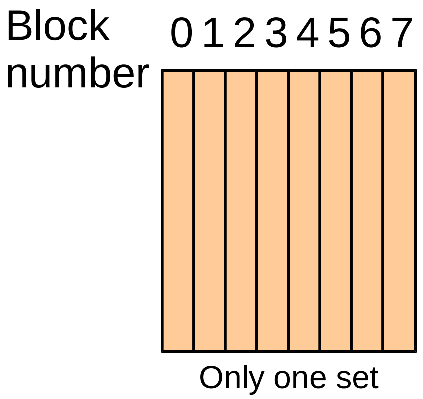
Example to illustrate base cache types

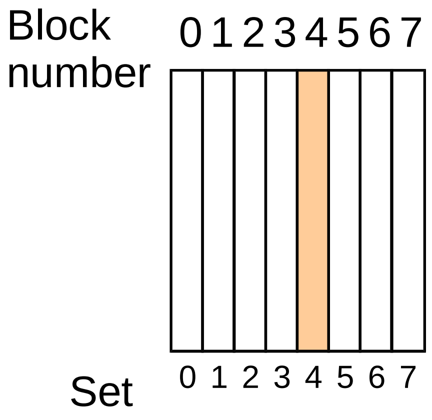
- The cache capacity 8 blocks. Where can be block/address 12 placed for
  - Fully associative
  - Direct mapped
  - N-way (set) associative – i.e. N=2 (2-way cache)

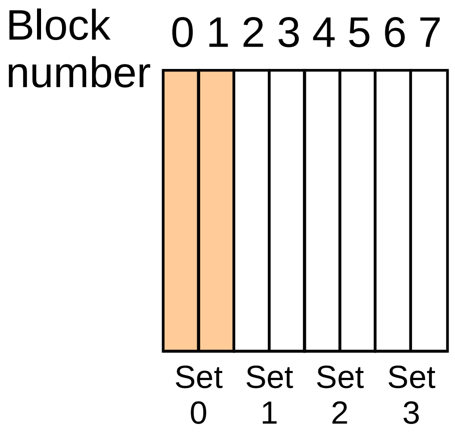
Fully associative:
Address 12 can be placed anywhere

Direct mapped:
Address 12 placed only to block 4 (12 mod 8)

2-way associative:
Address 12 is placed into set 0 (12 mod 4)

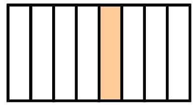






Direct mapped cache

direct mapped cache: one block in each set



Capacity – C

Number of sets – S

Block size – b

Number of blocks – B

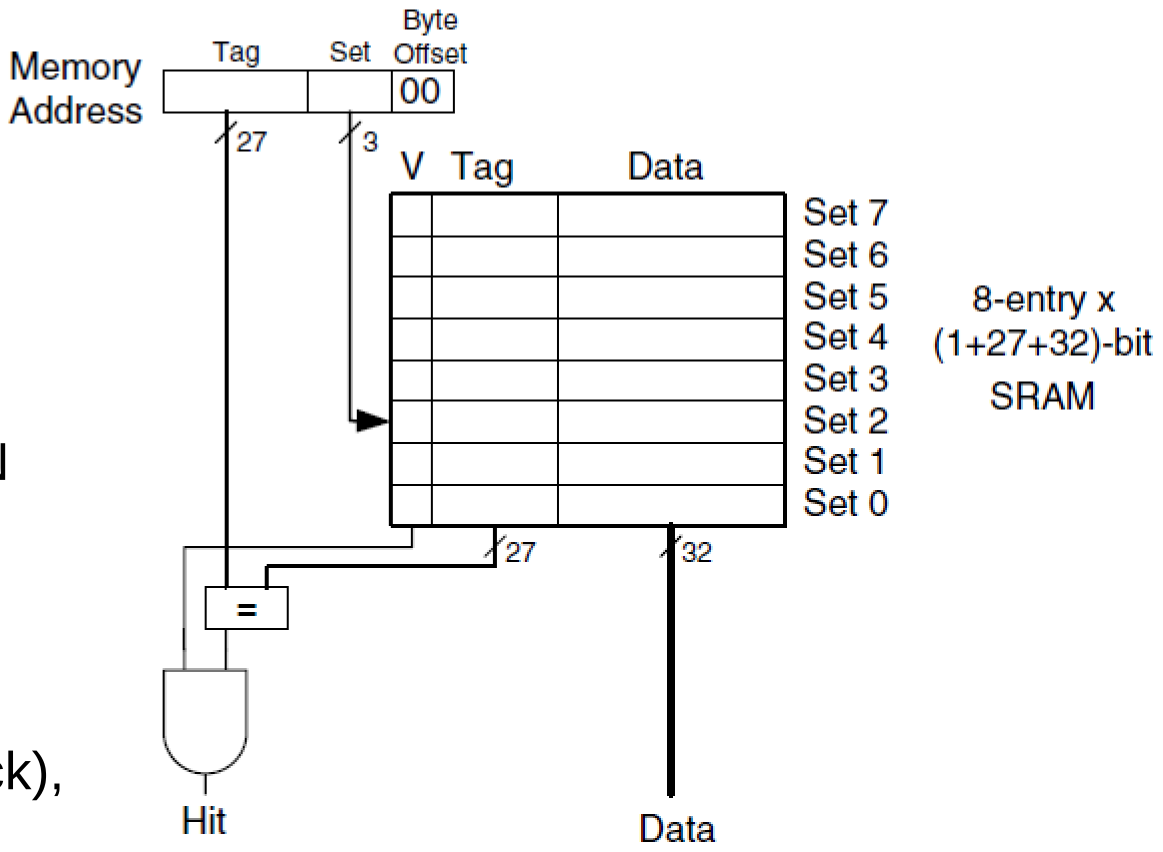
Degree of associativity – N

C = 8 (8 words),

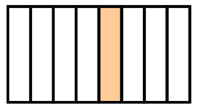
S = B = 8,

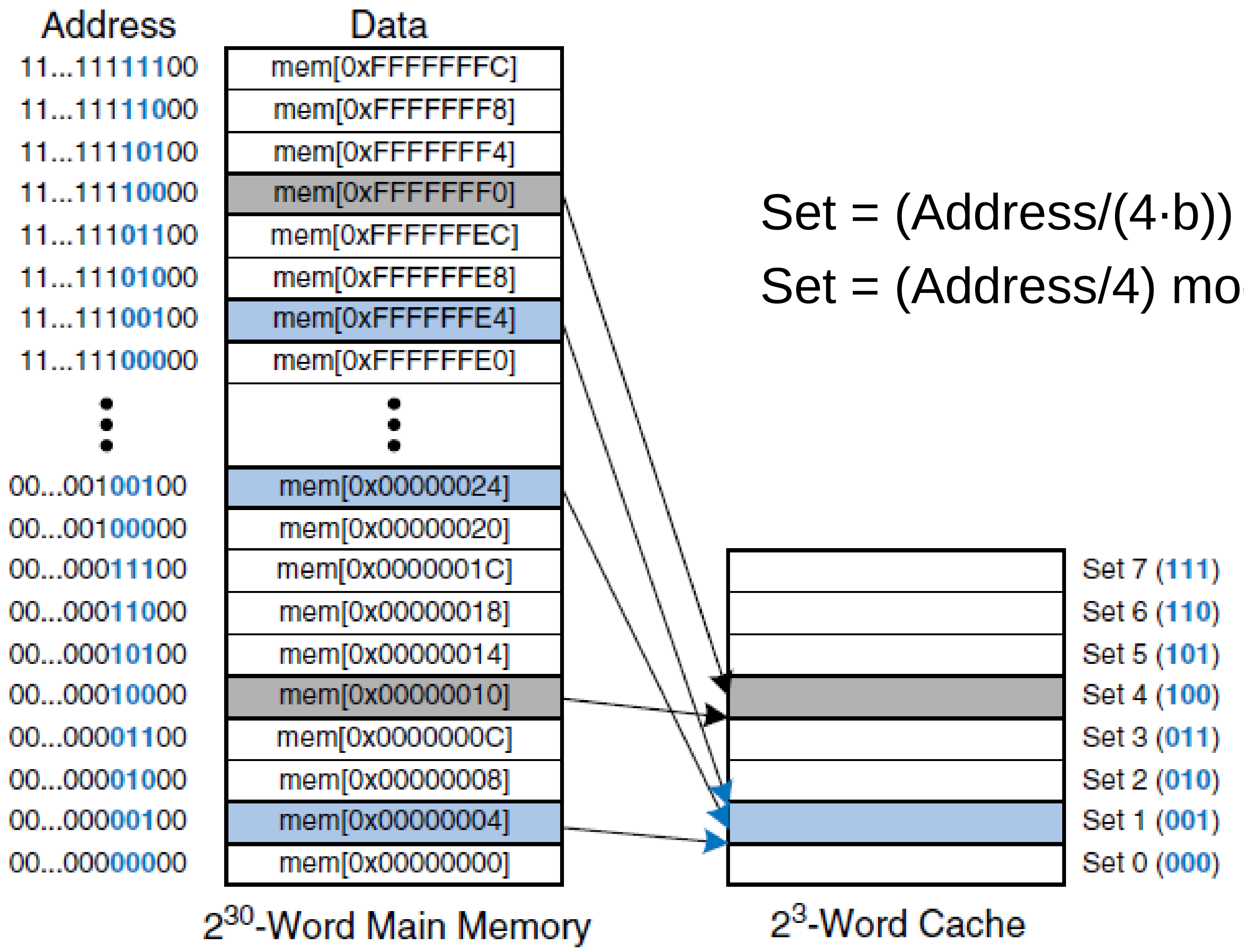
b = 1 (one word in the block),

N = 1



Direct mapped cache





Set = (Address/(4·b)) mod S

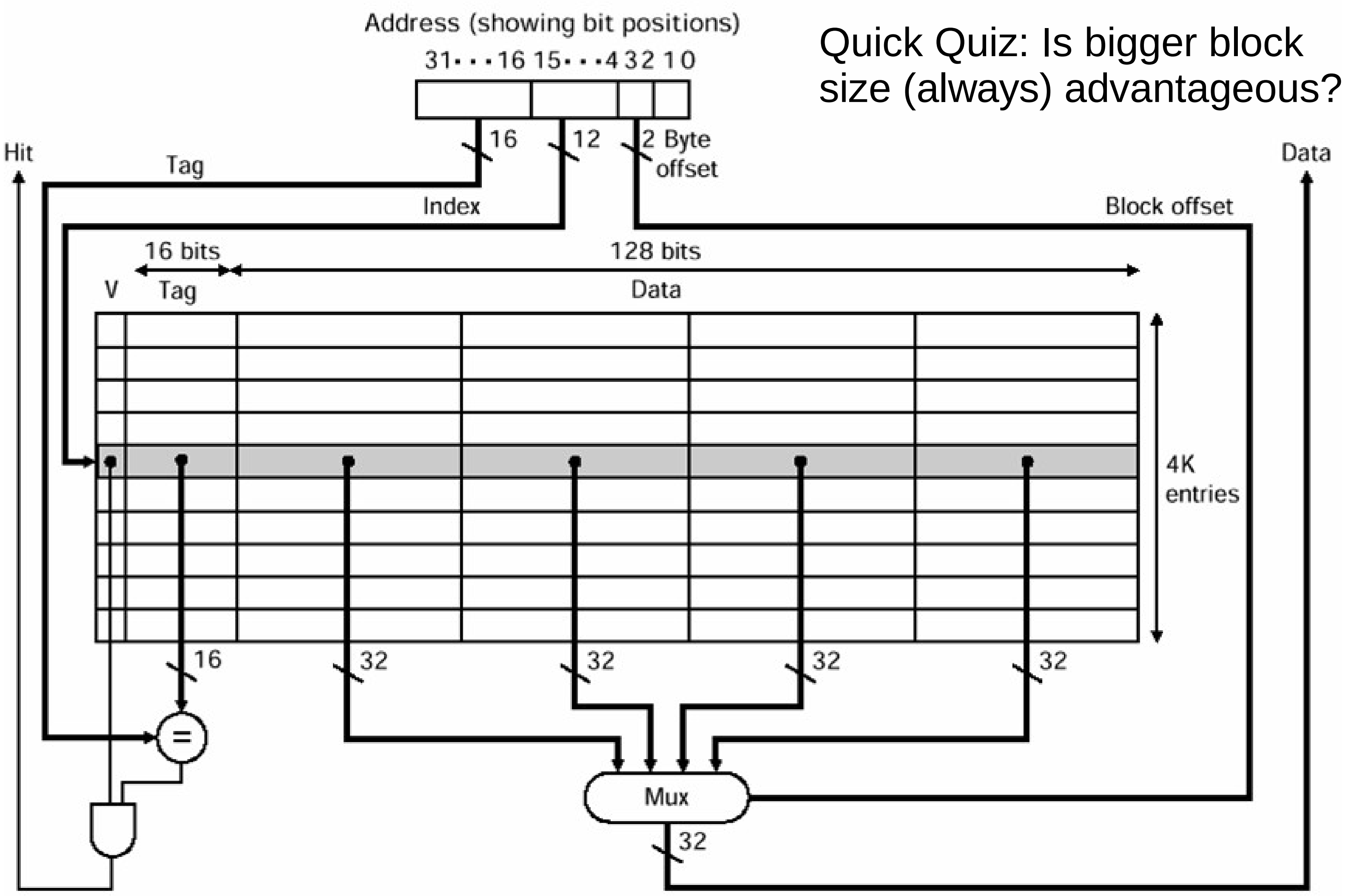
Set = (Address/4) mod 8

Real cache organization

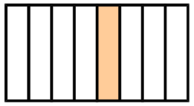
- **Tag** is index of the block corresponding to the cache set size in the main memory (that is address divided by block length and number or the cache lines in the set)
- **Data** are organized in cache line blocks, multiple words.
- **Valid bit** – marks line contents (or sometimes individual words) as valid.
- **Dirty bit** – corresponding cache line (word) was modified and write back will be required later
- Possible cache line states (for coherence protocols) – Invalid, Owned, Shared, Modified, Exclusive – out of the scope for this lecture

V	**Flags, i.e. dirty bit D**	**Tag**	**Data**

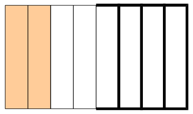
Direct mapped cache implementation

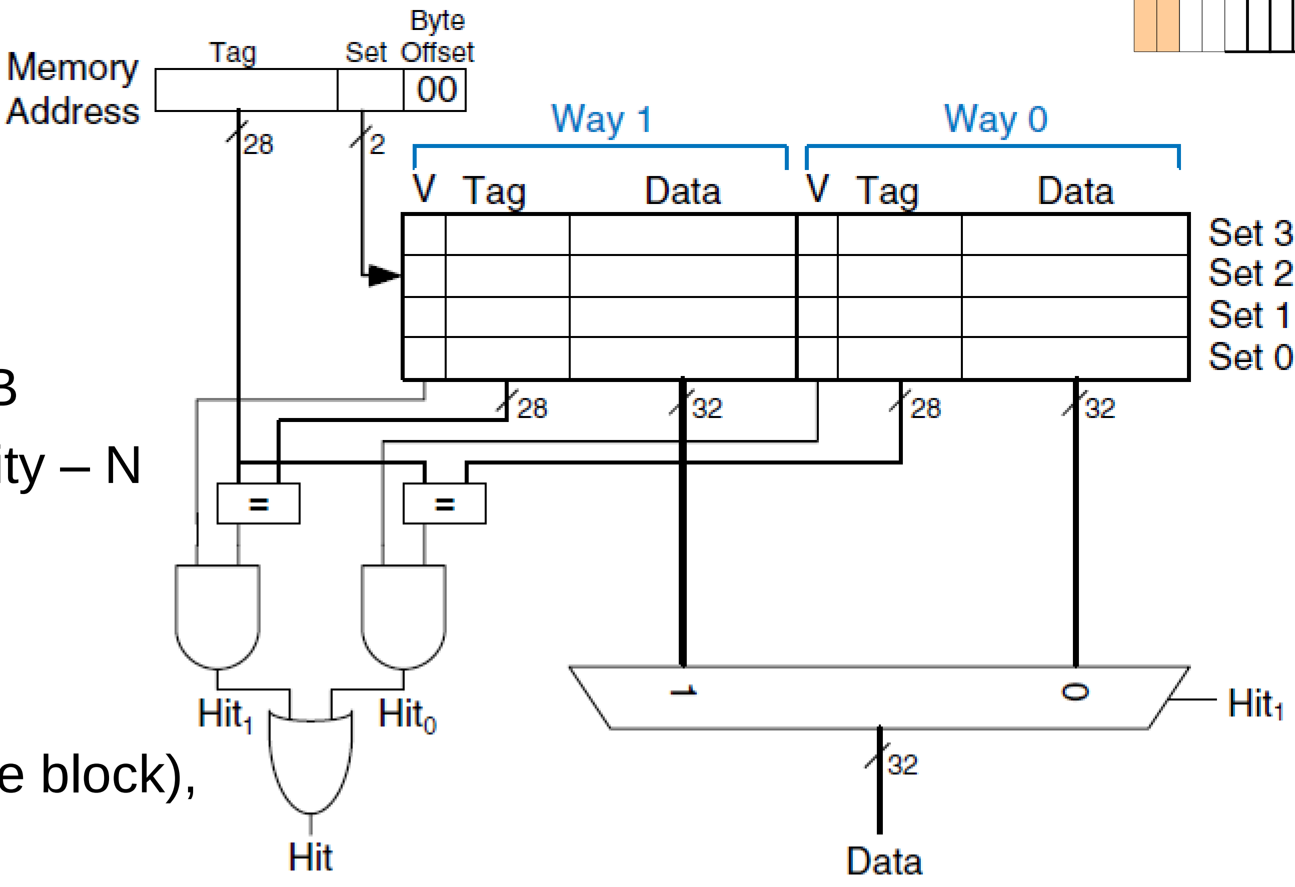


Quick Quiz: Is bigger block size (always) advantageous?



2-way set associative cache





Capacity – C

Number of sets – S

Block size – b

Number of blocks – B

Degree of associativity – N

C = 8 (8 words),

S = 4,

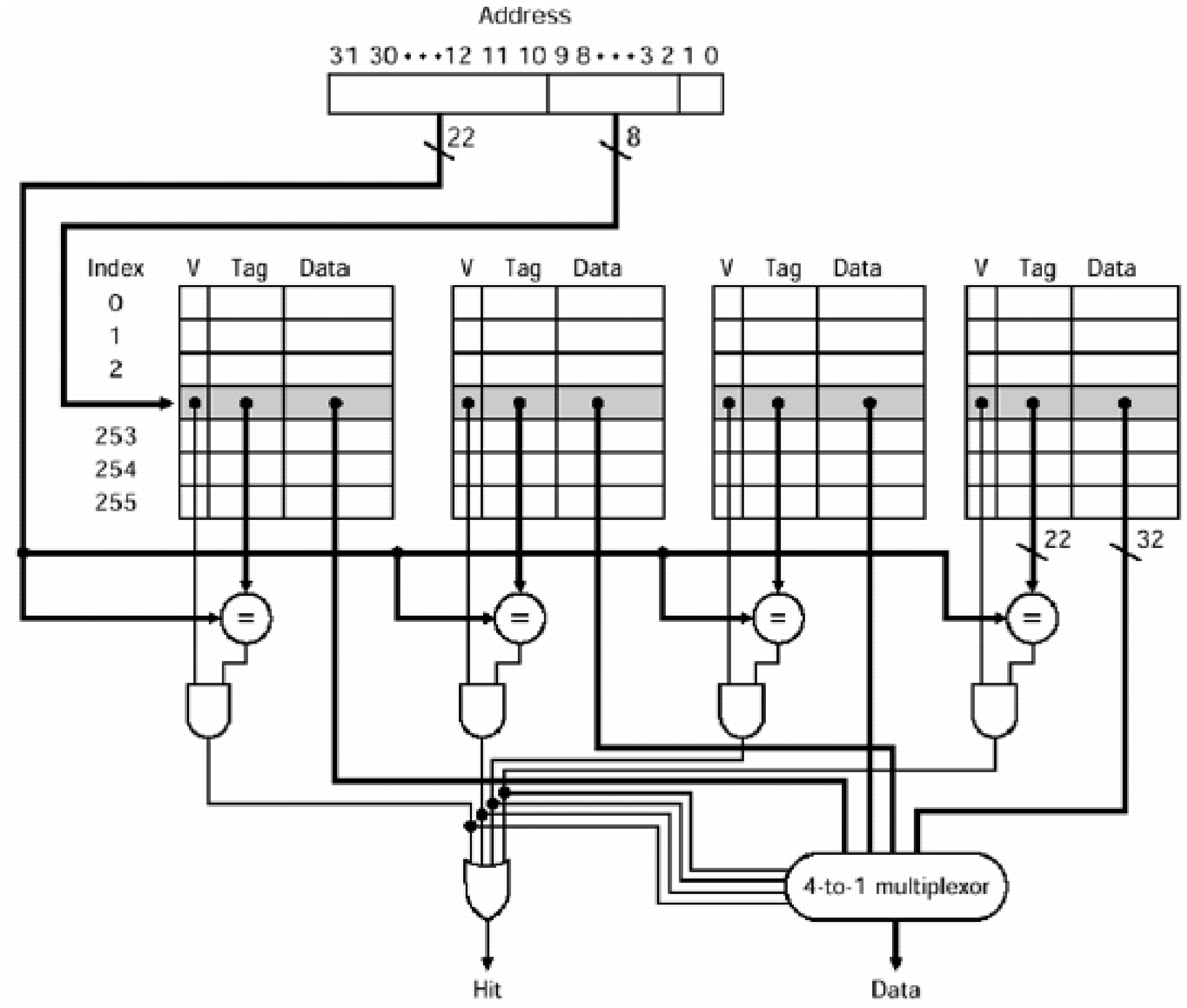
b = 1 (one word in the block),

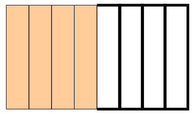
B = 8

N = 2

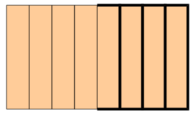
What is main advantage of higher associativity?

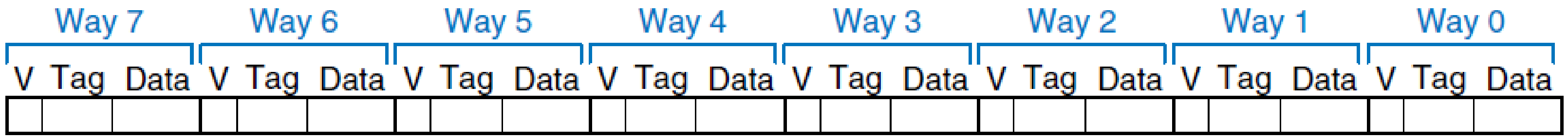
4-way set associative cache





Fully associative cache as special N-way case




Way 7
Way 6
Way 5
Way 4
Way 3
Way 2
Way 1
Way 0
V Tag Data V Tag Data V Tag Data V Tag Data V Tag Data V Tag Data V Tag Data V Tag Data

- From the above, a fully associative cache can be considered as N-way with only one set. N=B=C/(b·4)
- The same way we can define direct mapped cache as a special case where the number of ways is one.

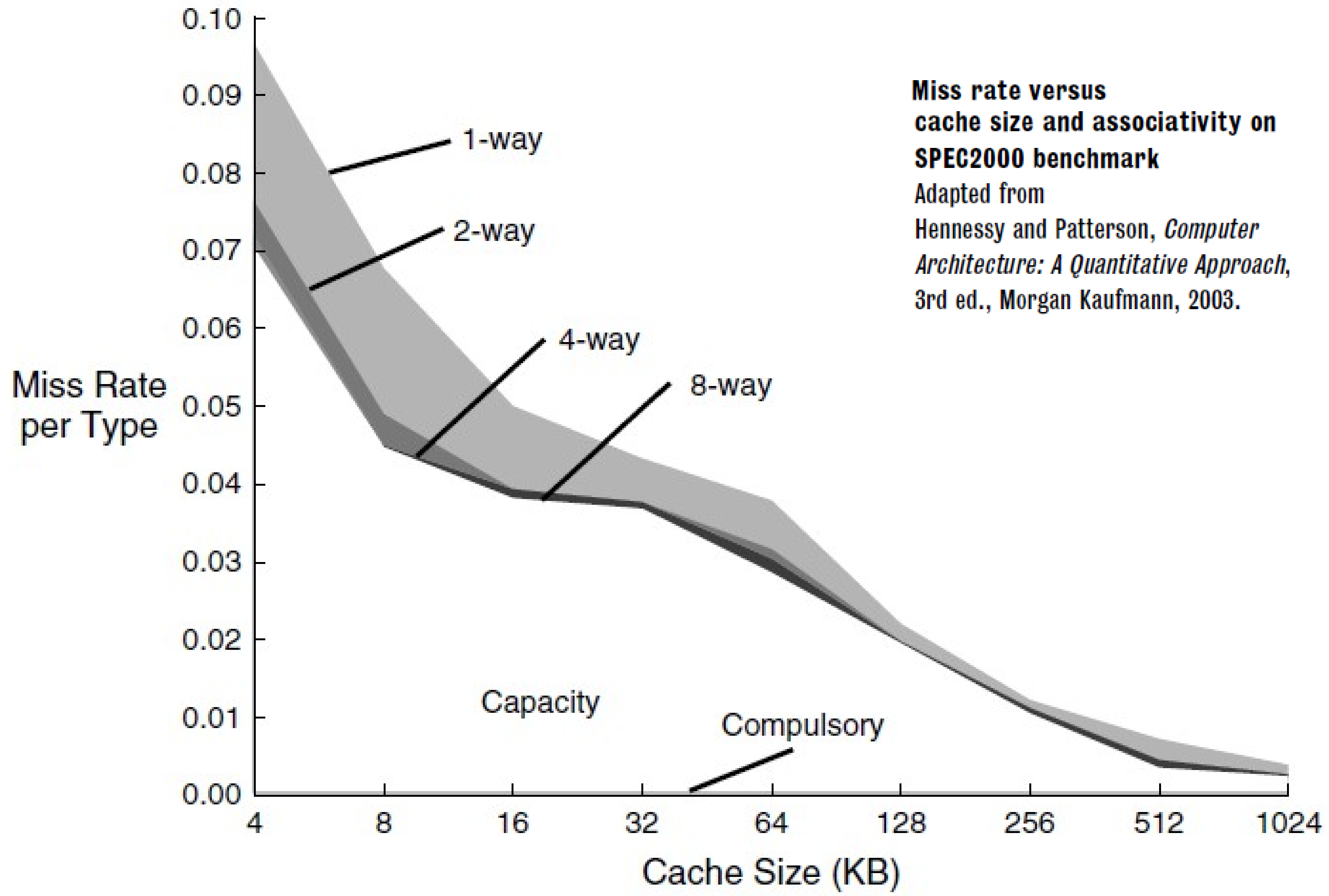
Important cache access statistical parameters

- **Hit Rate** – number of memory accesses satisfied by given level of cache divided by number of all memory accesses
- **Miss Rate** – same, but for requests resulting in access to slower memory = 1 – Hit Rate
- **Miss Penalty** – time required to transfer block (data) from lower/slower memory level
- Average Memory Access Time (AMAT)

$$\text{AMAT} = \text{Hit Time} + \text{Miss Rate} \times \text{Miss Penalty}$$

- Miss Penalty for multi-level cache can be computed by recursive application of AMAT formula

Comparison of different cache sizes and organizations

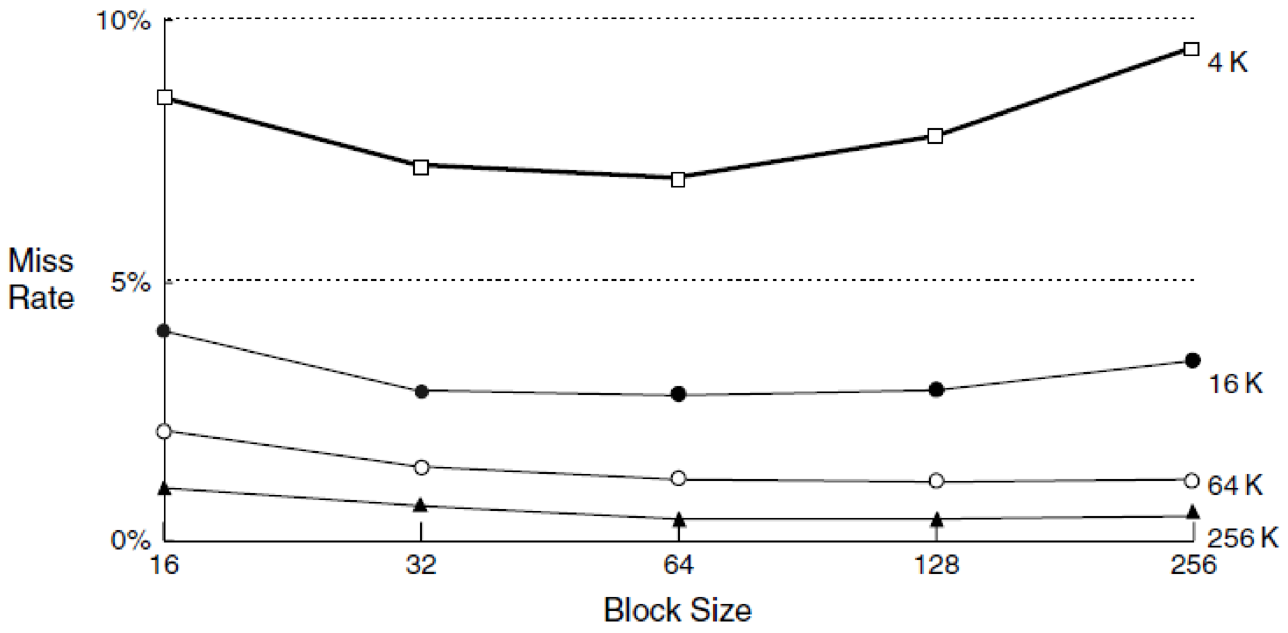


Miss rate versus cache size and associativity on SPEC2000 benchmark
Adapted from
Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed., Morgan Kaufmann, 2003.

Remember: 1. miss rate is not cache parameter/feature!
2. miss rate is not parameter/feature of the program!

What can be gained from spatial locality?

Miss rate of consecutive accesses can be reduced by increasing block size. On the other hand, increased block size for same cache capacity results in smaller number of sets and higher probability of conflicts (set number aliases) and then to increase of miss rate.



Miss rate versus block size and cache size on SPEC92 benchmark Adapted from Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed., Morgan Kaufmann, 2003.

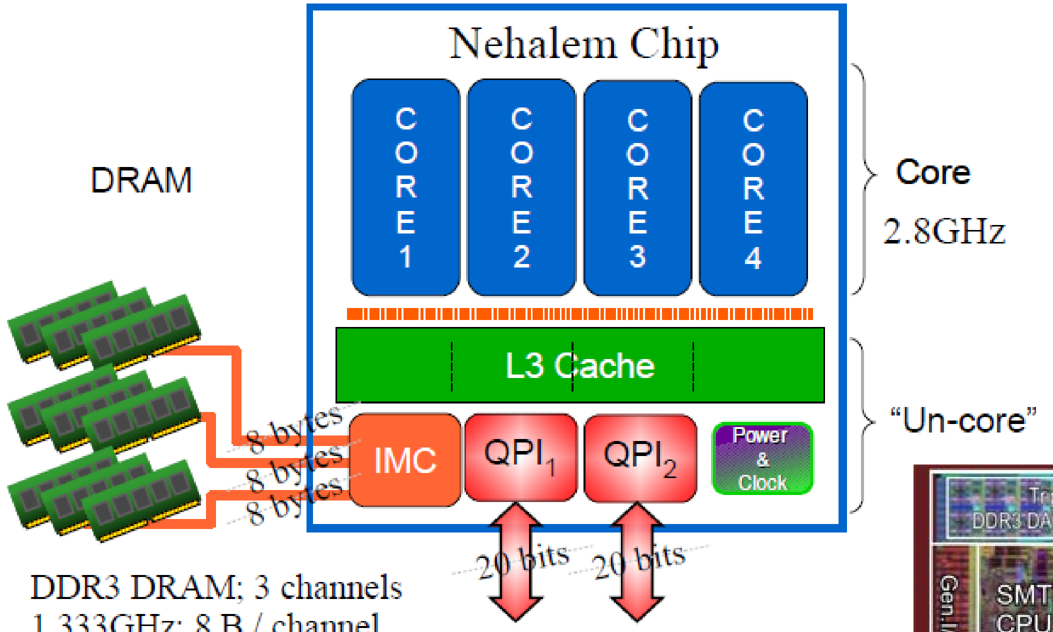
Multi-level cache organization

Multiple cache levels – development directions

- Primary/L1 cache – tightly coupled to the CPU
  - Fast but small. Main objective: minimal Hit Time/latency
  - Usually separated caches for instruction and for data
  - Size usually selected so that cache lines can be virtually tagged without aliasing. (set/way size is smaller than page size)
- L2 cache resolves cache misses of the primary cache
  - Much bigger and slower but still faster than main memory. Main goal: low Miss Rate
- L2 cache misses are resolved by main memory
- Trend to introduce L3 caches, inclusive versus exclusive cache

	Usual for L1	**Usual for L2**
Block count	250-2000	15 000-250 000
KB	16-64	2 000-3 000
Block size in bytes	16-64	64-128
Miss penalty (cycles)	10-25	100-1 000
Miss rates	2-5%	0,1-2%

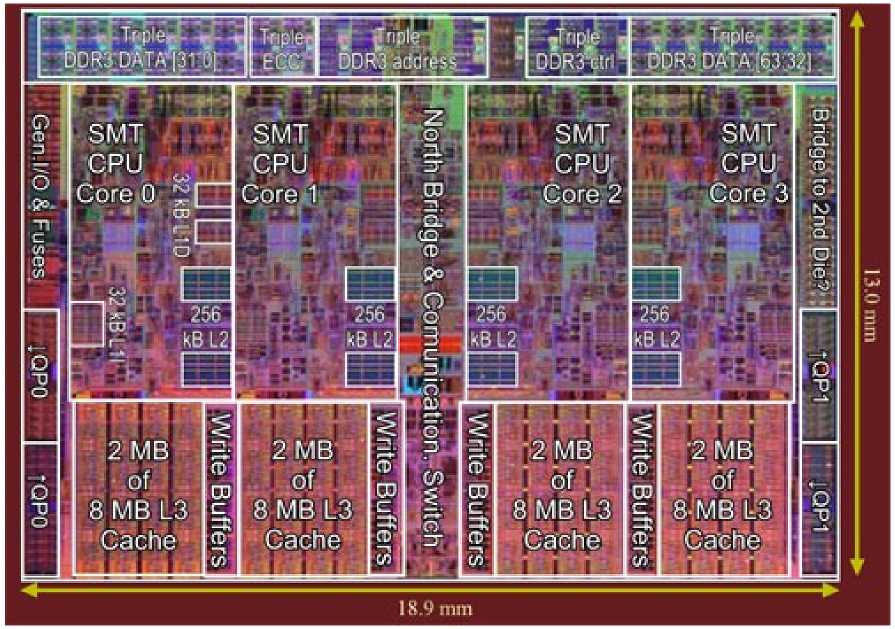
Intel Nehalem – example of Harvard three-level cache



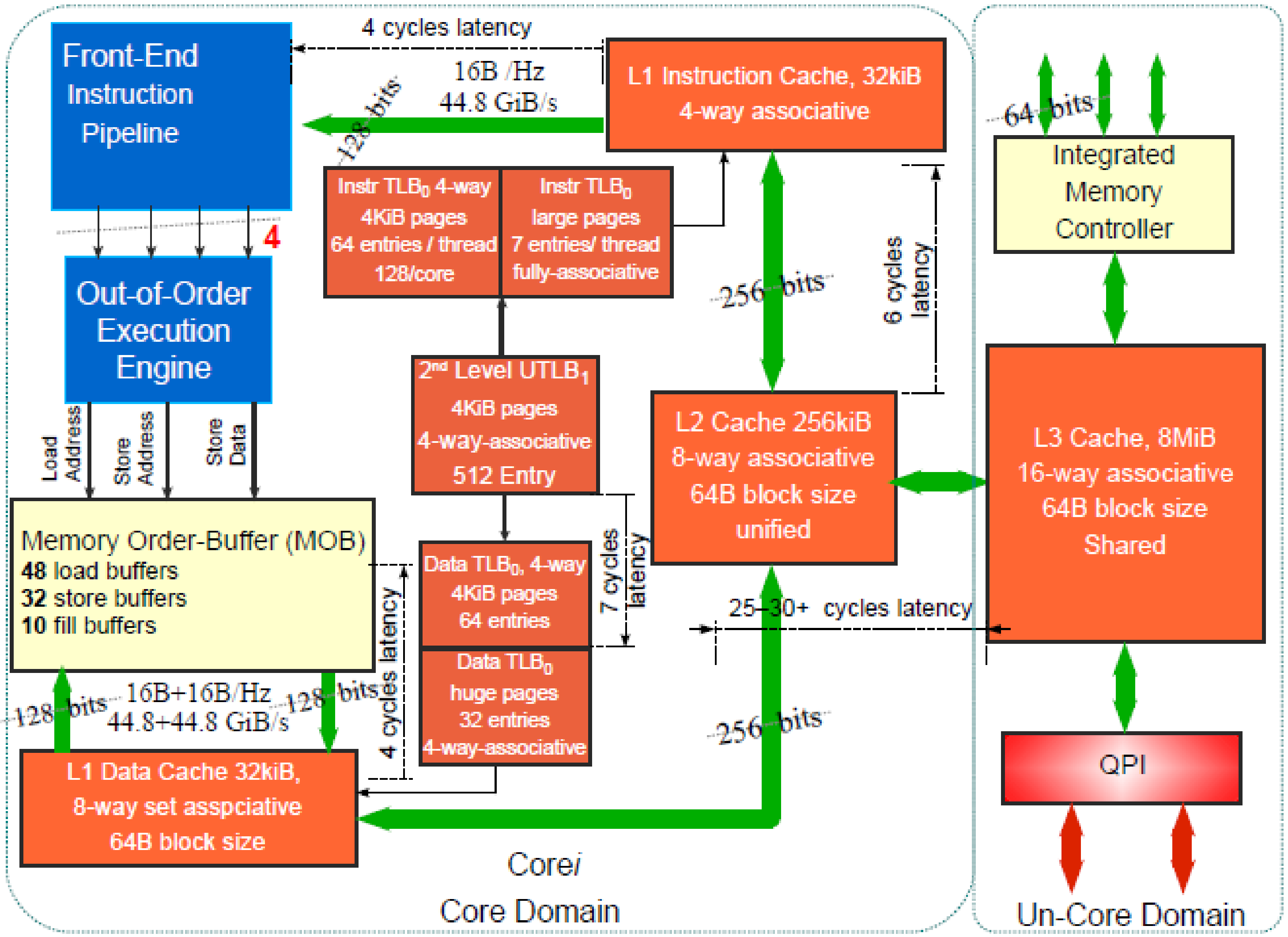
DDR3 DRAM; 3 channels
1.333GHz; 8 B / channel
31.992 GB/s aggregate
7.998 GB/s / core

Intel® QPI point-to-point link
6.4 GT/s; full-duplex;
12.8 GiB/s + **12.8**GiB/s

- IMC: integrated memory controller with 3 DDR3 memory channels,
- QPI: Quick-Path Interconnect ports

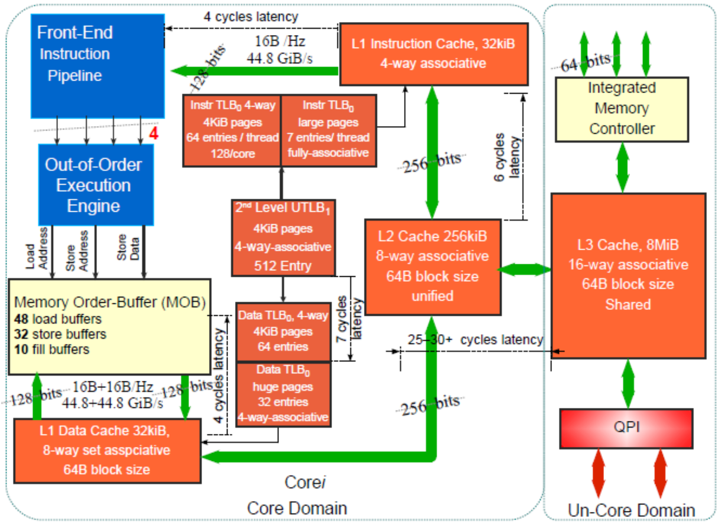


Intel Nehalem – memory subsystem structure



Front-End Instruction Pipeline
4 cycles latency
16B /Hz
44.8 GiB/s
128 bits
L1 Instruction Cache, 32kiB 4-way associative
Instr TLB0 4-way 4KiB pages 64 entries / thread 128/core
Instr TLB0 large pages 7 entries/ thread fully-associative
4
Out-of-Order Execution Engine
256 bits
6 cycles latency
64 bits
Integrated Memory Controller
2nd Level UTLB1 4KiB pages 4-way-associative 512 Entry
L2 Cache 256kiB 8-way associative 64B block size unified
L3 Cache, 8MiB 16-way associative 64B block size Shared
Load Address
Store Address
Store Data
Memory Order-Buffer (MOB)
48 load buffers
32 store buffers
10 fill buffers
Data TLB0, 4-way 4KiB pages 64 entries
7 cycles latency
25–30+ cycles latency
Data TLB0 huge pages 32 entries 4-way-associative
4 cycles latency
128 bits
16B+16B/Hz
44.8+44.8 GiB/s
128 bits
256 bits
QPI
L1 Data Cache 32kiB, 8-way set asspciative 64B block size
Corei
Core Domain
Un-Core Domain

Notes for Intel Nehalem example


Front-End Instruction Pipeline
4 cycles latency
16B /Hz 44.8 GiB/s
128 bits
L1 Instruction Cache, 32kiB 4-way associative
Instr TLB0 4-way 4KiB pages 64 entries / thread 128/core
Instr TLB0 large pages 7 entries/ thread fully-associative
Out-of-Order Execution Engine
256 bits
6 cycles latency
64 bits
Integrated Memory Controller
2nd Level UTLB1 4KiB pages 4-way-associative 512 Entry
L2 Cache 256kiB 8-way associative 64B block size unified
L3 Cache, 8MiB 16-way associative 64B block size Shared
Load Address
Store Address
Store Data
Memory Order-Buffer (MOB) 48 load buffers 32 store buffers 10 fill buffers
Data TLB0, 4-way 4KiB pages 64 entries
Data TLB0 huge pages 32 entries 4-way-associative
7 cycles latency
25-30+ cycles latency
4 cycles latency
128 bits
16B+16B/Hz 44.8+44.8 GiB/s
L1 Data Cache 32kiB, 8-way set asspciative 64B block size
QPI
Corei
Core Domain
Un-Core Domain

- Block size: 64B
- CPU reads whole cache line/block from
  main memory and each is 64B aligned
- (6 LS bits are zeros), partial line fills allowed
- L1 – Harvard. Shared by two (H)threads
  instruction – 4-way 32kB, data 8-way 32kB
- L2 – unified, 8-way, non-inclusive, WB
- L3 – unified, 16-way, inclusive (each line stored in L1 or L2 has copy in L3), WB
- Store Buffers – temporal data store for each write to eliminate wait for write to the cache or main memory. Ensure that final stores are in original order and solve “transaction” rollback or forced store for:

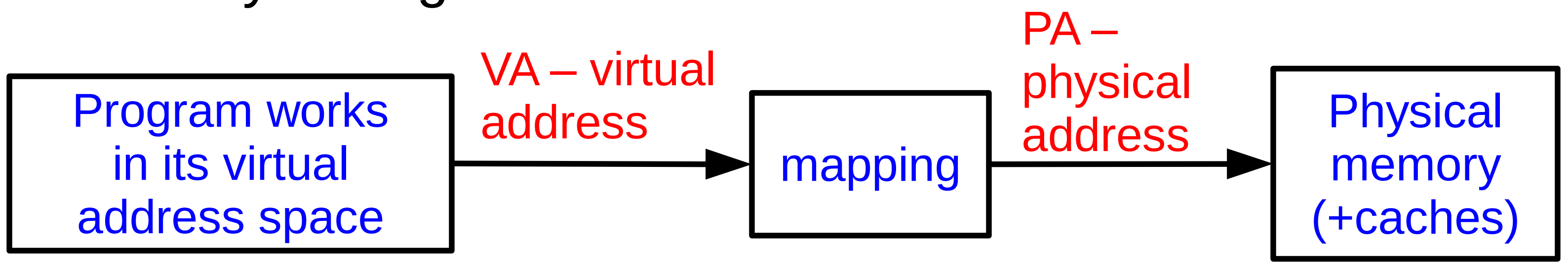
  - exceptions, interrupts, serialization/barrier instructions, lock prefix,..
- TLBs (Translation Lookaside Buffers) are separated for the first level

  Data L1 32kB/8-ways results in 4kB range (same as page) which allows to use 12 LSBs of virtual address to select L1 set in parallel with MMU/TLB

Virtual memory

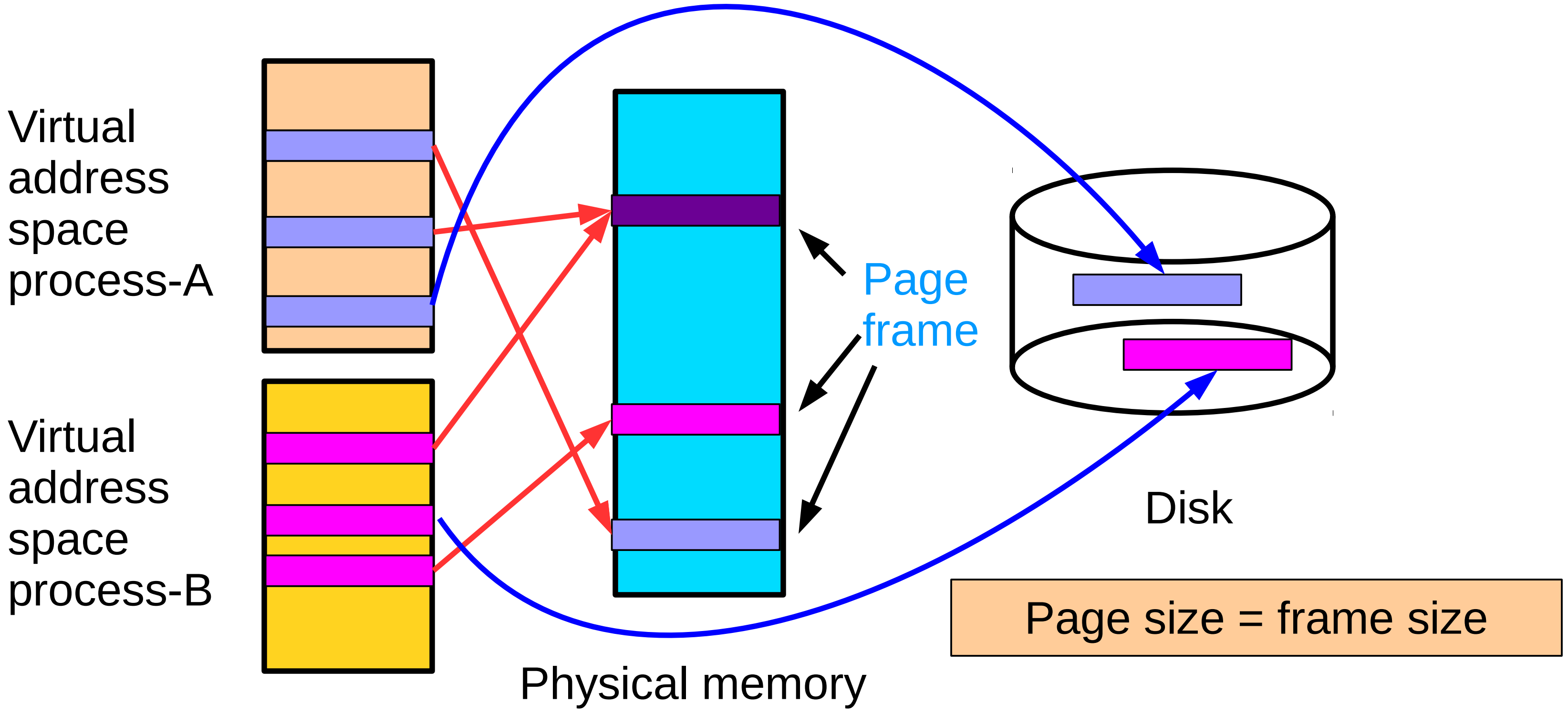
Virtual memory

- **Virtual memory (VM)** – a separate address space is provided to each process, it is (can be) organized independently on the physical memory ranges and can be even bigger than the whole physical memory
- Programs/instructions running on the CPU operate with data only through virtual addresses
- Translation from virtual address (VA) to physical address (PA) is implemented in HW (MMU, TLB).
- Common OSes implement virtual memory through paging which extends concept even to swapping memory content onto secondary storage

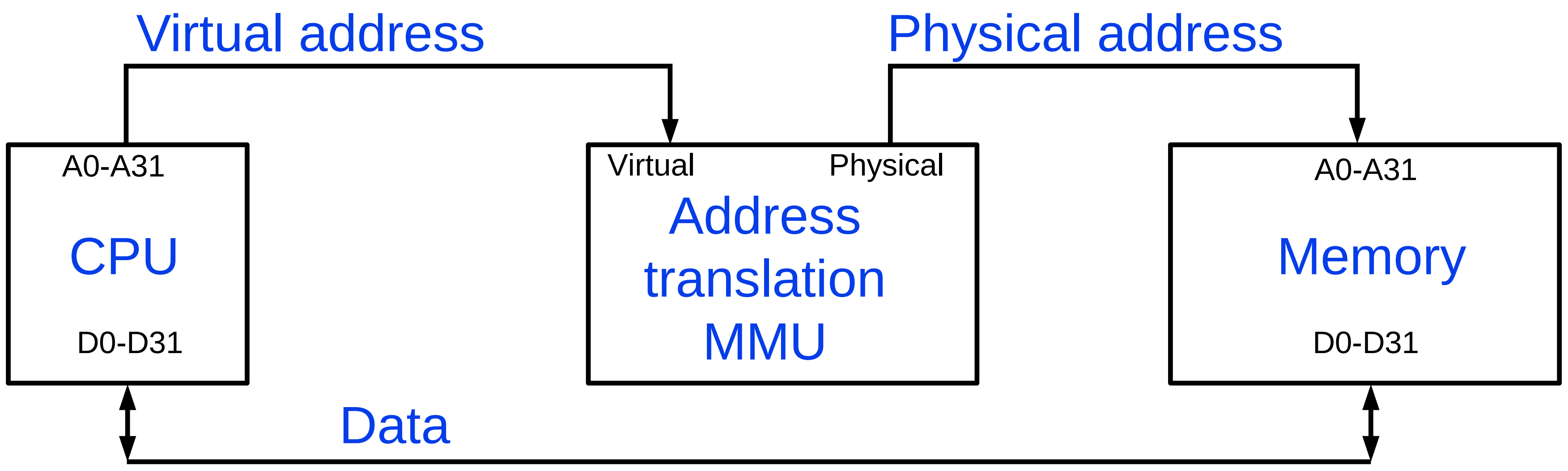

Program works in its virtual address space
VA – virtual address
mapping
PA – physical address
Physical memory (+caches)

Virtual memory – paging

- Process virtual memory content is divided into aligned pages of same size (power of 2, usually 4 or 8 kB)
- Physical memory consists of page frames of the same size
- Note: huge pages option on modern OS and HW – 2^n pages


Virtual address space process-A
Virtual address space process-B
Page frame
Disk
Physical memory
Page size = frame size

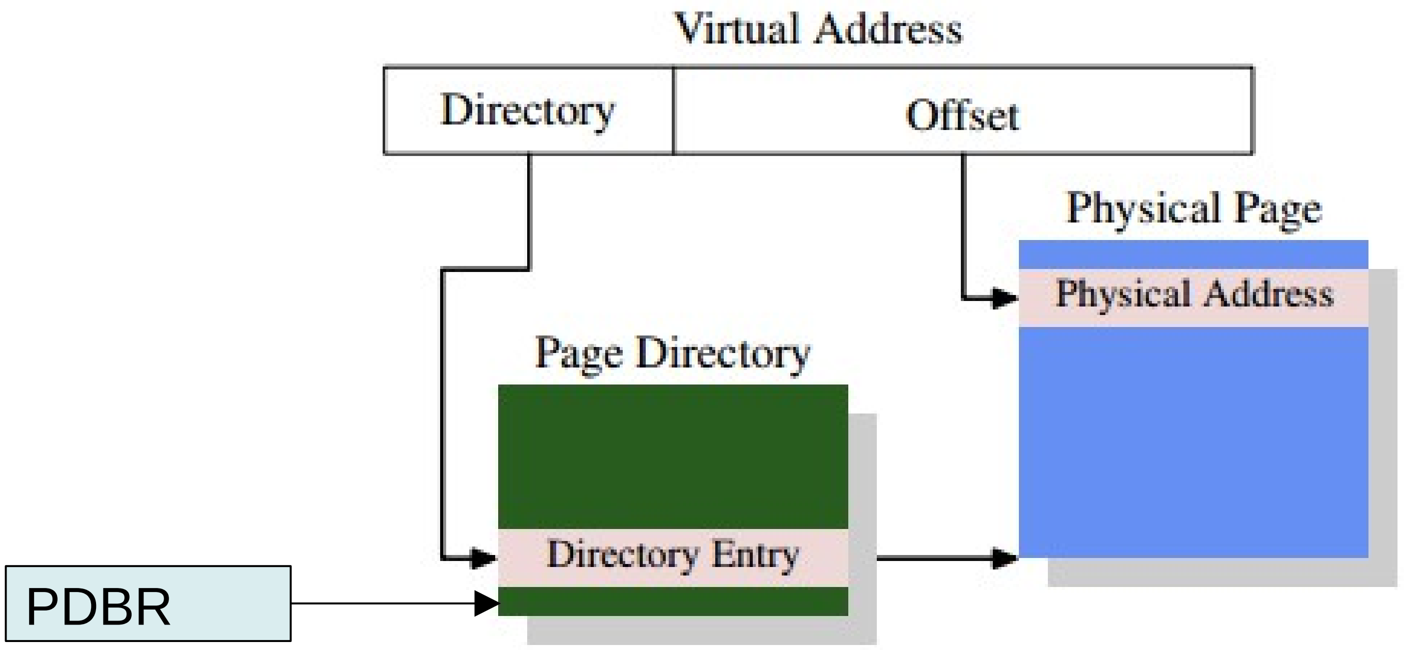
Virtual/physical address and data



Address translation

- Page Table
  - Root pointer/page directory base register (x86 CR3=PDBR)
  - Page table directory PTD
  - Page table entries PTE
- Basic mapping unit is a page (page frame)
- Page is basic unit of data transfers between main memory and secondary storage
- Mapping is implemented as look-up table in most cases
- Address translation is realized by **Memory Management Unit** (MMU)
- Example follows on the next slide:

Single-level page table (MMU)

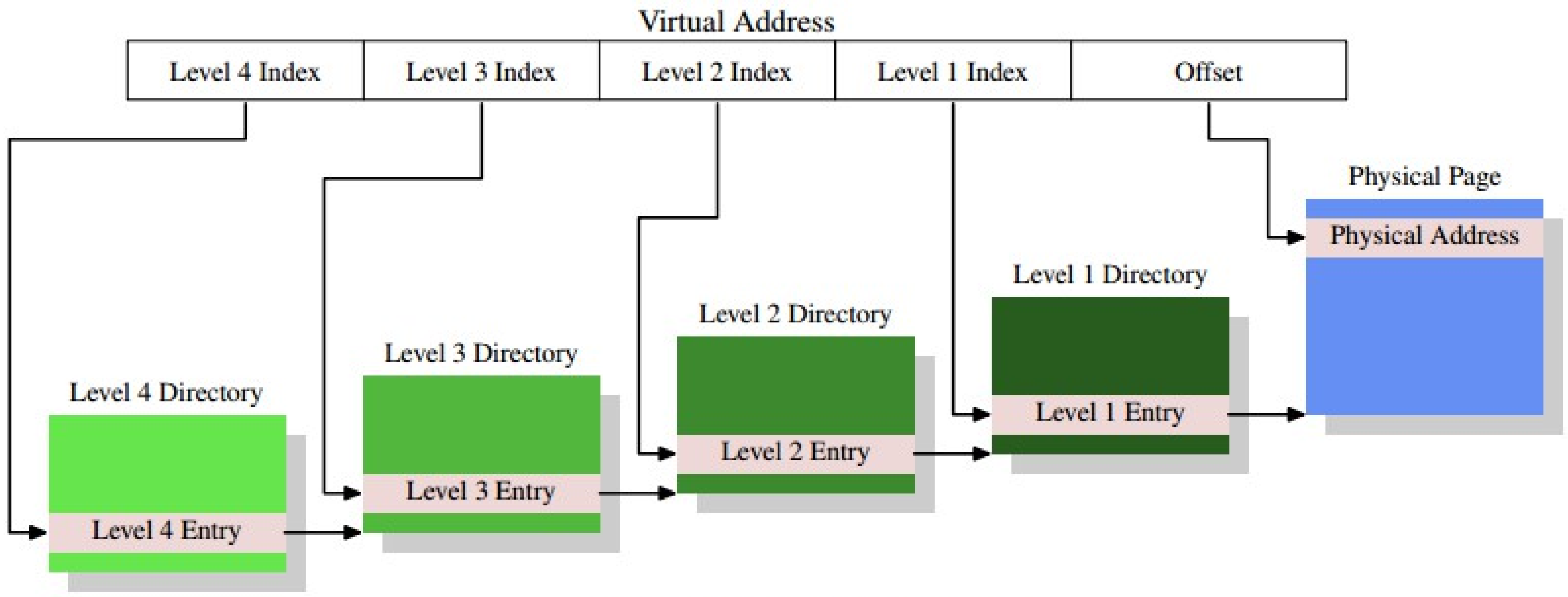


- Page directory is represented as data structure stored in main memory. OS task is to allocate physically continuous block of memory (for each process/memory context) and assign its start address to special CPU/MMU register.
- PDBR - page directory base register – for x86 register CR3 – holds physical address of page directory start, alternate names PTBR - page table base register – the same thing, page table root pointer URP, SRP on m68k

But consider memory consumed by page table …

- Typical page size is 4 kB = 2^12
- 12 bits (offset) are enough to address data in page (frame). There are 20 bits left for address translation on 32-bit address/architecture.
- The fastest map/table look-up is indexing ⇒ use array structure
- The page directory is an array of 2^20 entries (PTE). That is big overhead for processes that do not use whole virtual address range. There are another problems as well (physical space allocation fragmentation when large compact table is used for each process, etc.)
- Solution: multi-level page table – lower levels populated only for used address ranges

Multi-levels page table

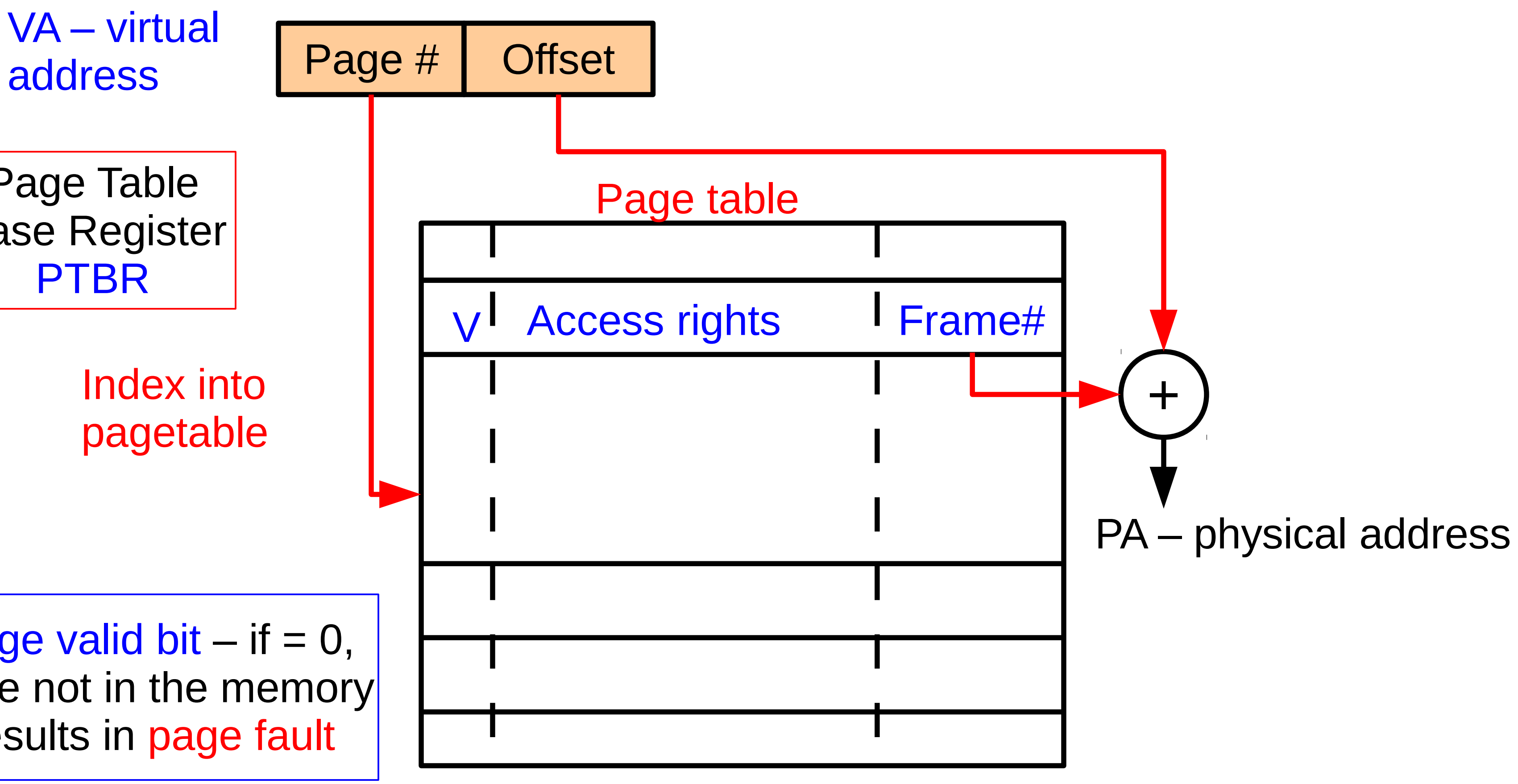


4-Level Address Translation

What is in page table entries?

Page Table Base Register PTBR

Page valid bit – if = 0, page not in the memory results in page fault



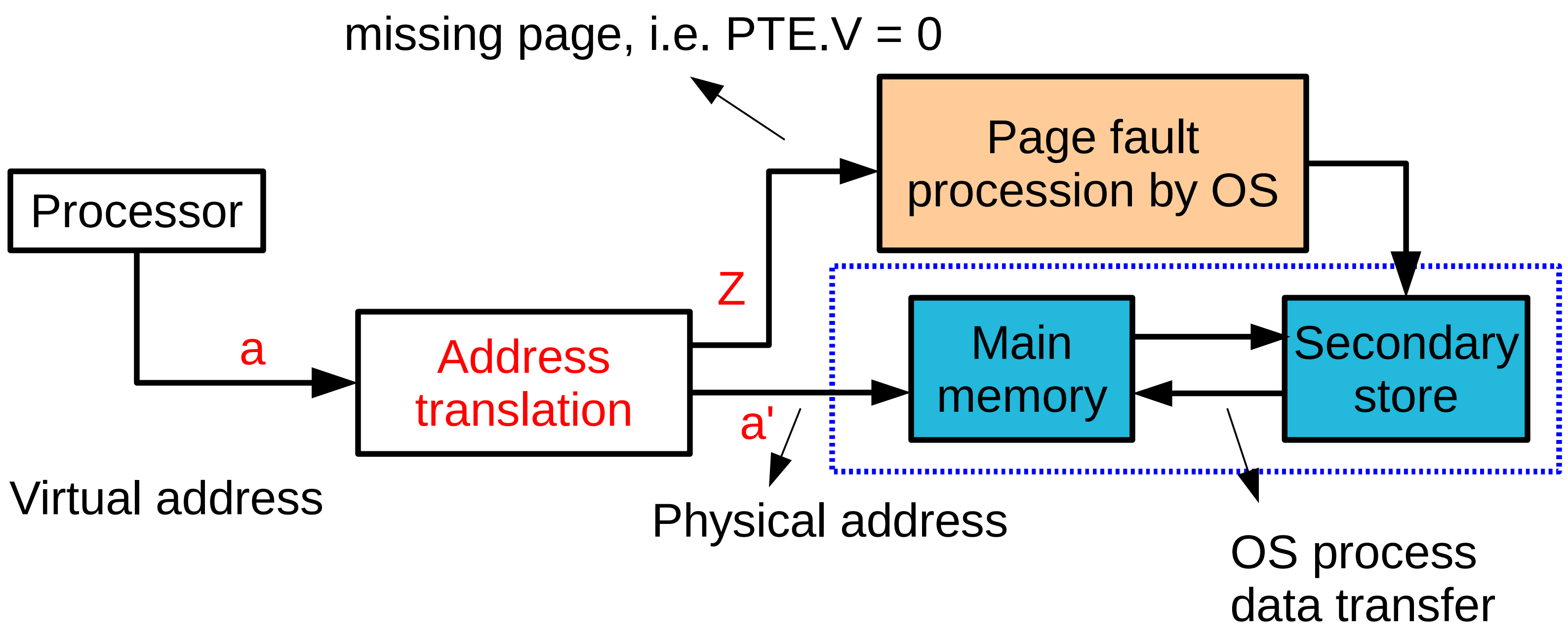
Page table placed in physical memory

Remarks

- Each process has its own page table
- Process specific value of CPU PTBRT register is loaded by OS when given process is scheduled to run
- This ensures memory separation and protection between processes
- Page table entry format
  - V – Validity Bit. V=0 $\rightarrow$ page is not valid (is invalid)
  - AR – Access Rights (Read Only, Read/Write, Executable, etc.),
  - Frame# - page frame number (location in physical memory)
  - Other management information, Modified/Dirty, (more bits discussed later, permission, system, user etc.).

V	AR	Frame#

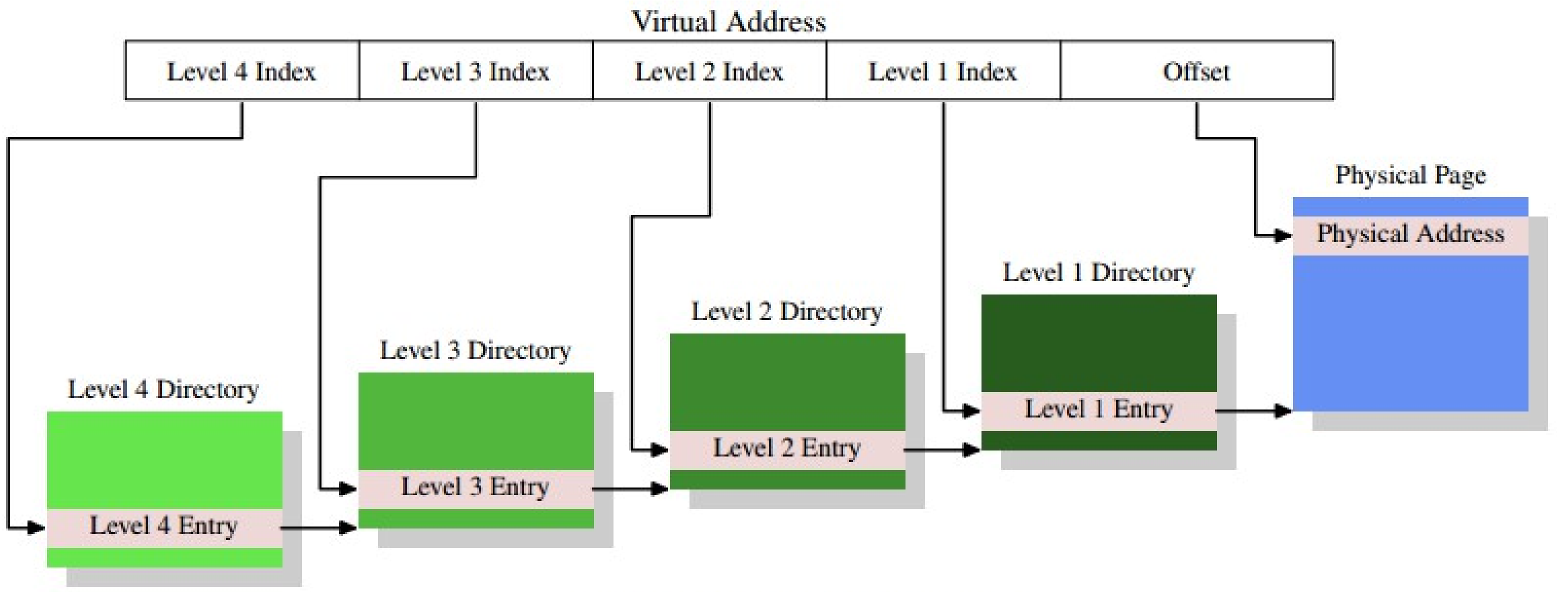
Virtual memory – Hardware and software interaction



How to resolve page-fault

- Check first that fault address belongs to process mapped areas
- If free physical frame is available
  - The missing data are found in the backing store (usually swap or file on disk)
  - Page content is read (usually through DMA, Direct Memory Access, part of some future lesson) to the allocated free frame. If read blocks, the OS scheduler switches to another process.
  - End of the DMA transfer raises interrupt, OS updates page table of original process.
  - Scheduler switches to (resumes) original process.
- If no free frame is available, some frame has to be released
  - The LRU algorithm finds (unpinned – not locked in physical memory by OS) frame, which can be released.
  - If the Dirty bit is set, frame content is written to the backing store (disc). If store is a swap – store to the PTE or other place block nr.
  - Then continue with gained free physical frame.

Multi-level page table – translation overhead

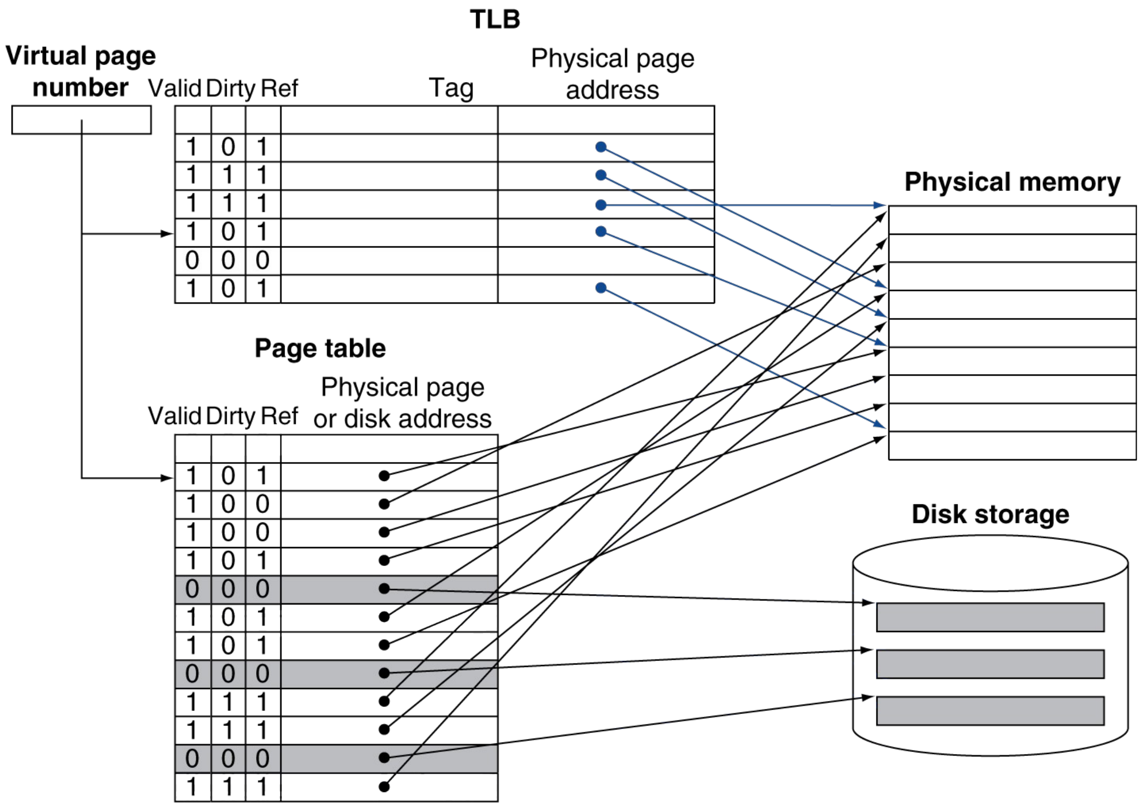


4-Level Address Translation

- Translation would take long time, even if entries for all levels were present in cache. (One access per level, they cannot be done in parallel.)
- The solution is to cache found/computed physical addresses
- Such cache is labeled as Translation Look-Aside Buffer
- Even multi-level translation caching are in use today

Fast MMU/address translation using TLB

- Translation-Lookaside Buffer, or may it be, more descriptive name – Translation-Cache
- Cache of frame numbers where key is page virtual addresses



Typical sizes of today I/D and TLB caches comparison

	Typical paged memory parameters	Typical TLB
Size in blocks	16 000-250 000	40-1024
Size	500-1 000 MB	0,25-16 KB
Block sizes in B	4 000-64 000	4-32
Miss penalty (clock cycles)	10 000 000 – 100 000 000	10-1 000
Miss rates	0,00001-0,0001%	0,01-2
Backing store	Pages on the disk	Page table in the main memory
Fast access location	Main memory frames	TLB

Hierarchical memory caveats

Some problems to be aware of

- Memory coherence – definition on next slide
- Single processor (single core) systems
  - Solution: D-bit and Write-back based data transactions
  - Even in this case, consistency with DMA requited (SW or HW)
- Multiprocessing (symmetric) SMP with common and shared memory – more complicated. Solutions:
  - Common memory bus: Snooping, MESI, MOESI protocol
  - Broadcast
  - Directories
- More about these advanced topics in A4M36PAP

Coherency definition

- Memory coherence is an issue that affects the design of computer systems in which two or more processors, cores or bus master controllers share a common area of memory.
- Intuitive definition: The memory subsystem is coherent if the value returned by each read operation is always the same as the value written by the most recent write operation to the same address.
- More formal: P – set of CPU's. $\mathbf{x_m} \in \mathbf{X}$ locations. $\forall \mathbf{p_i}, \mathbf{p_k} \in \mathbf{P}$: $\mathbf{p_i} \neq \mathbf{p_k}$. Memory system is coherent if
  1. $\mathbf{p_i}$ read after $\mathbf{p_i}$ write value **a** to $\mathbf{x_m}$ returns **a** if there is no $\mathbf{p_i}$ or $\mathbf{p_k}$ write between these read and write operations
  2. if $\mathbf{p_i}$ reads $\mathbf{x_m}$ after $\mathbf{p_k}$ write **b** to $\mathbf{x_m}$ and there is no other $\mathbf{p_i}$ or $\mathbf{p_k}$ write to $\mathbf{x_m}$ then $\mathbf{p_i}$ reads **b** if operations are separated by enough time (in other case previous value of $\mathbf{x_m}$ can be read) or architecture specified operations are inserted after write and before read.
  3. writes by multiple CPU's to the given location are serialized such than no CPU reads older value when it already read recent one

Comparison of virtual memory and cache memory

Virtual memory	Cache memory
Page	Block/cache line
Page Fault	Read/Write Miss
Page size: 512 B – 8 KB	Block size: 8 – 128 B
Fully associative	DM, N-way set associative
Victim selection: LRU	LRU/Random
Write Back	Write Thru/Write Back

- Remarks.: TLB for address translation can be fully associative, but for bigger sizes is 4-way.
- Do you understand the terms?
  - What does victim represent?
- Important: adjectives cache and virtual mean different things.

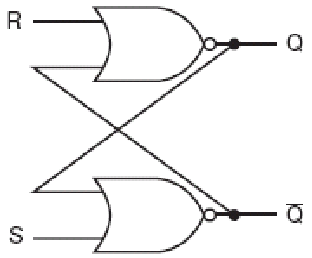
Inclusive versus exclusive cache/data backing store

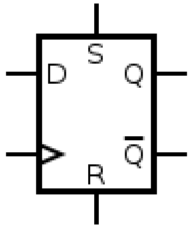
- Mapping of contents of the main memory to the cache memory is **inclusive**, i.e. main memory location cannot be reused for other data when corresponding or updated contents is held in the cache
- If there are more cache levels it can be waste of the space to keep stale/old data in the previous cache level. Snoop cycle is required anyway. The **exclusive** mechanism is sometimes used in such situation.
- **Inclusive** mapping is the rule for secondary storage files mapped into main memory.
- But for swapping of physical contents to swap device/file exclusive or mixed approach is quite common.

Memory realization – memory chips

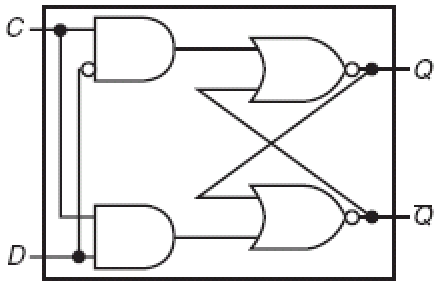
Flip-flop circuits – reminder from previous courses

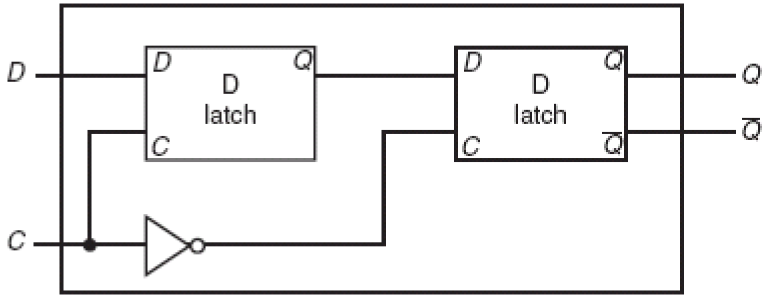
RS

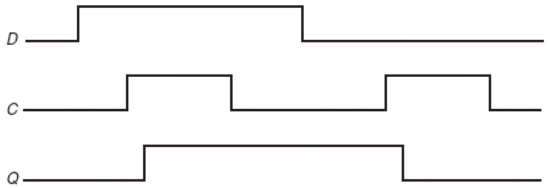


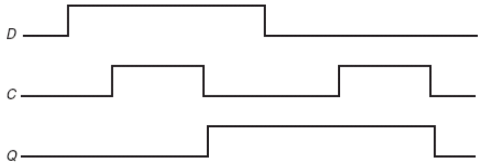


D latch, level-controlled flip-flop **D flip-flop**, edge-controlled flip-flop



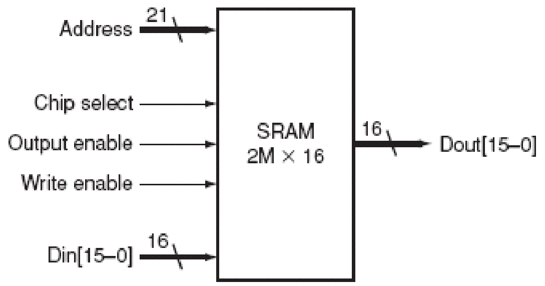




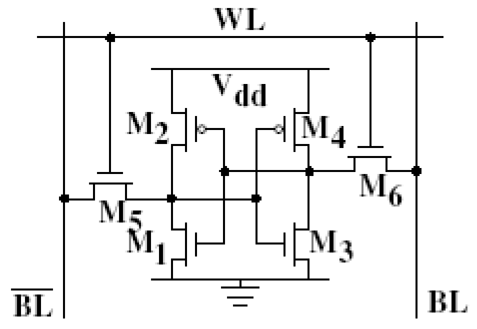


Usual SRAM chip and SRAM cell

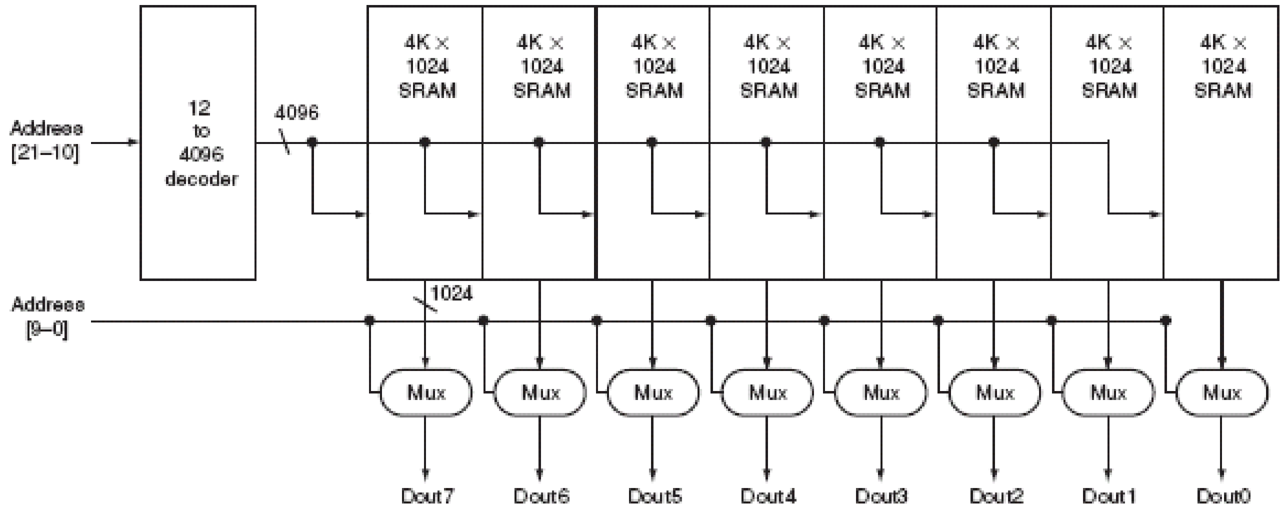
Usual SRAM chip



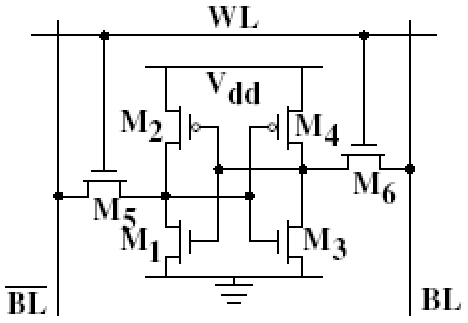
SRAM memory cell
CMOS technology



Bigger memory size?

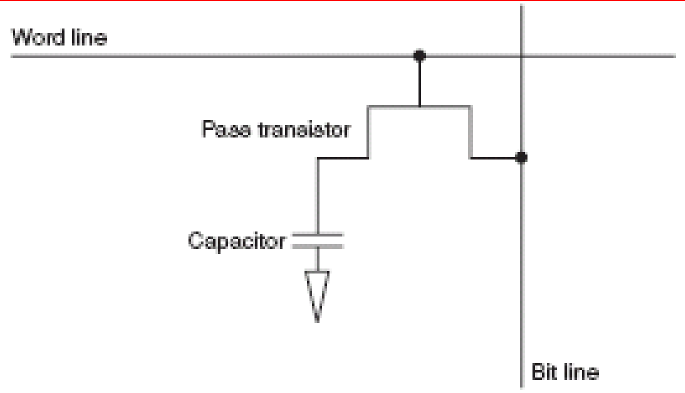


Detail of static and dynamic memory bit cell

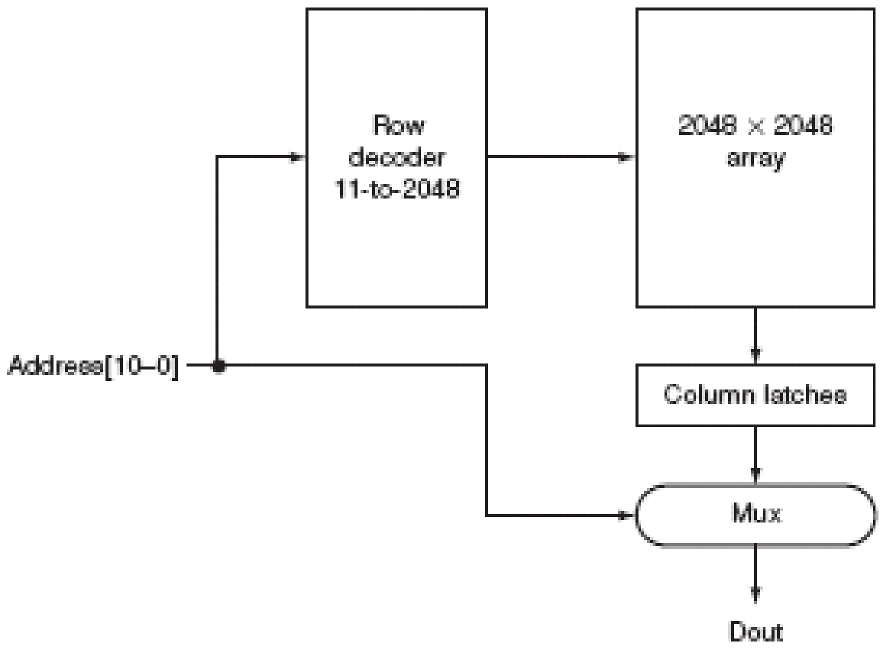


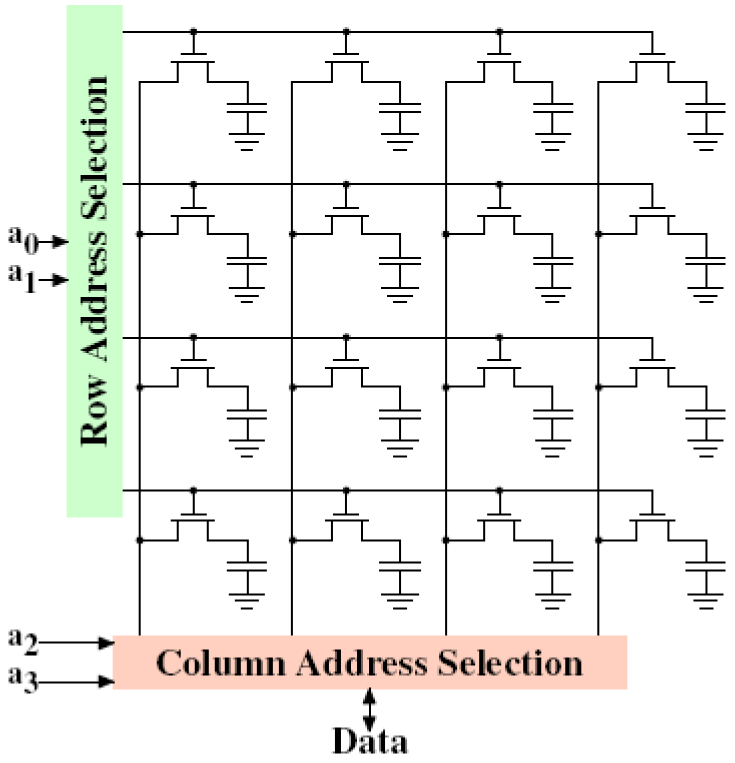
6 transistor static memory cell (single bit)

Single transistor cell of dynamic memory



Internal architecture of the DRAM memory chip

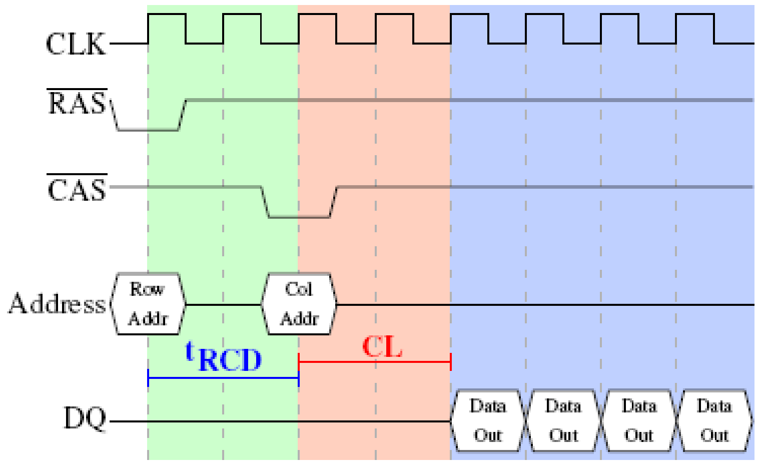




This 4M × 1 DRAM is internally realized as an 2048x2048 array of 1b memory cells

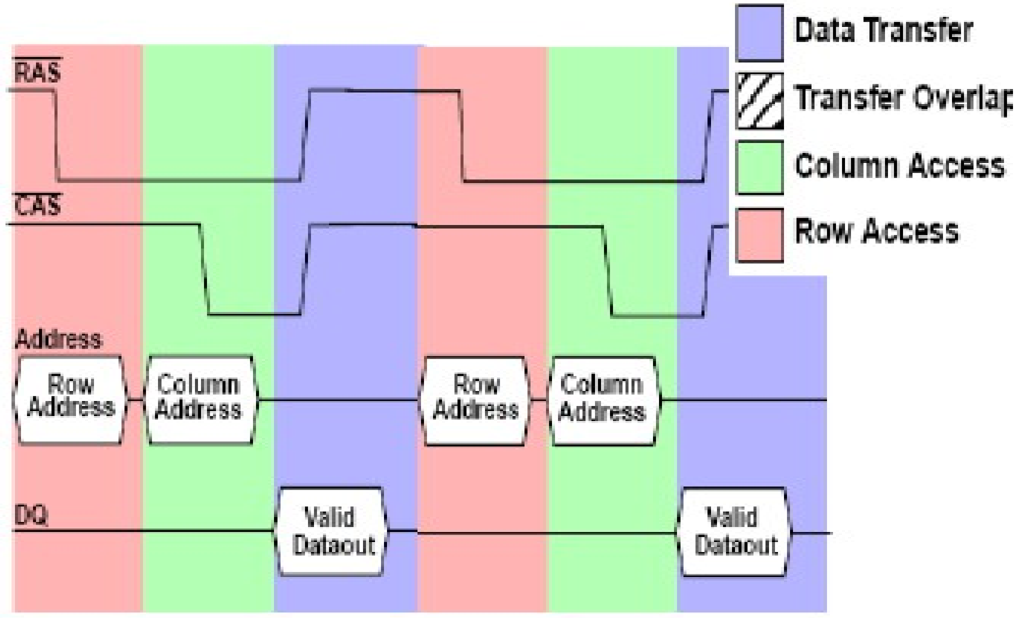
History of DRAM chips development

Year	Capacity	Price[$]/GB	Access time [ns]
1980	64 Kb	1 500 000	250
1983	256 Kb	500 000	185
1985	1 Mb	200 000	135
1989	4 Mb	50 000	110
1992	16 Mb	15 000	90
1996	64 Mb	10 000	60
1998	128 Mb	4 000	60
2000	256 Mb	1 000	55
2004	512 Mb	250	50
2007	1 Gb	50	40



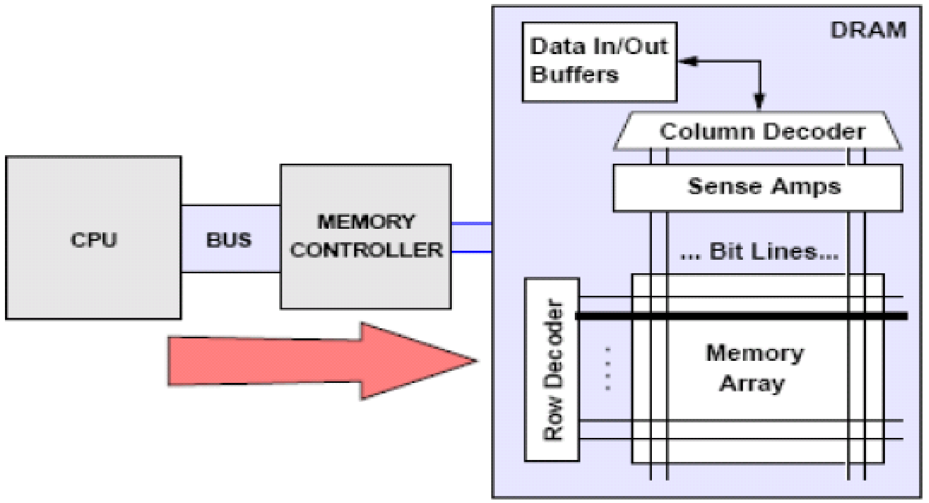
Old school DRAM – asynchronous access

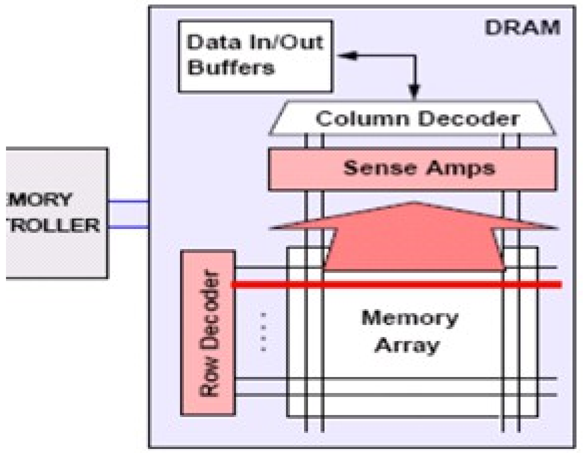
- The address is transferred in two phases – reduces number of chip module pins and is natural for internal DRAM organization
- This method is preserved even for today chips

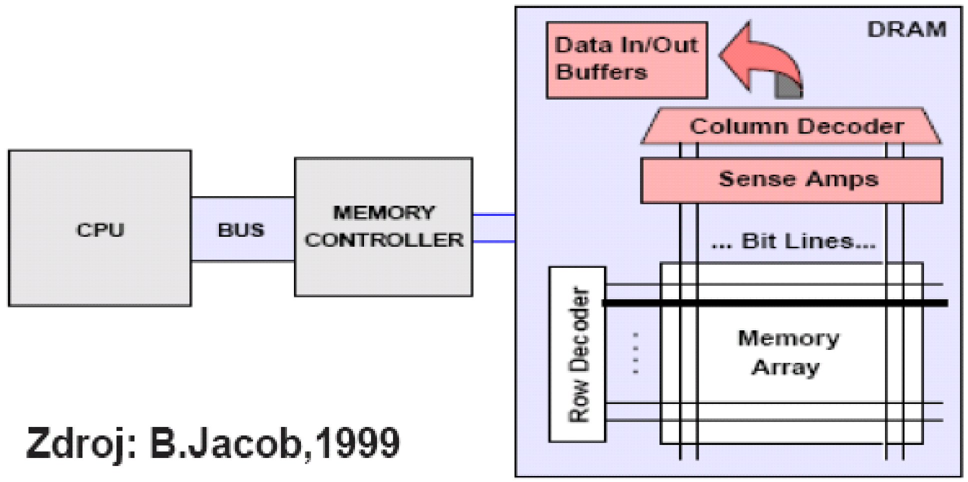


RAS – Row Address Strobe,
CAS – Column Address Strobe

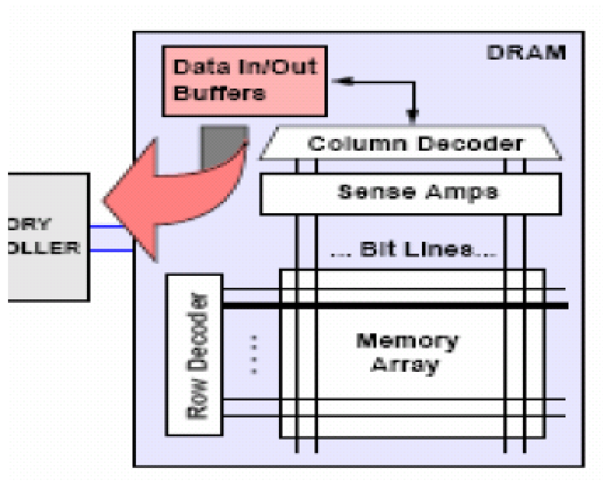
Phases of DRAM memory read







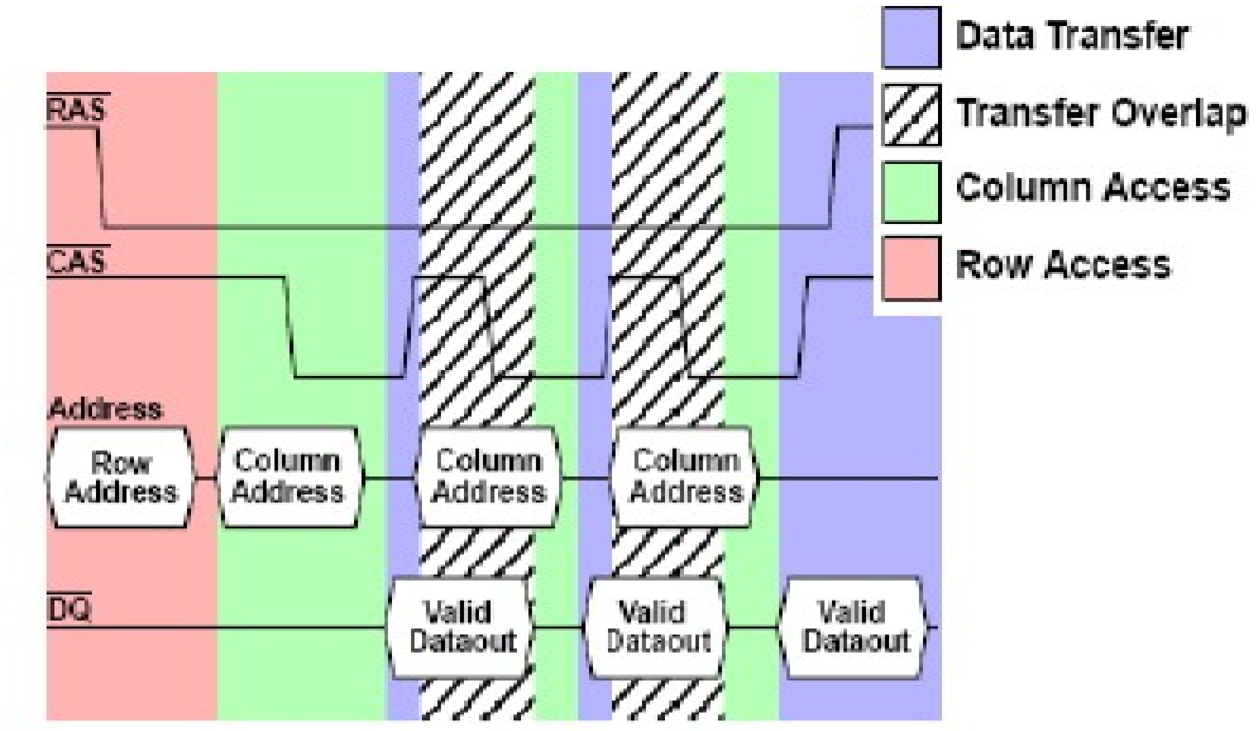
Zdroj: B.Jacob,1999



EDO-RAM – about 1995

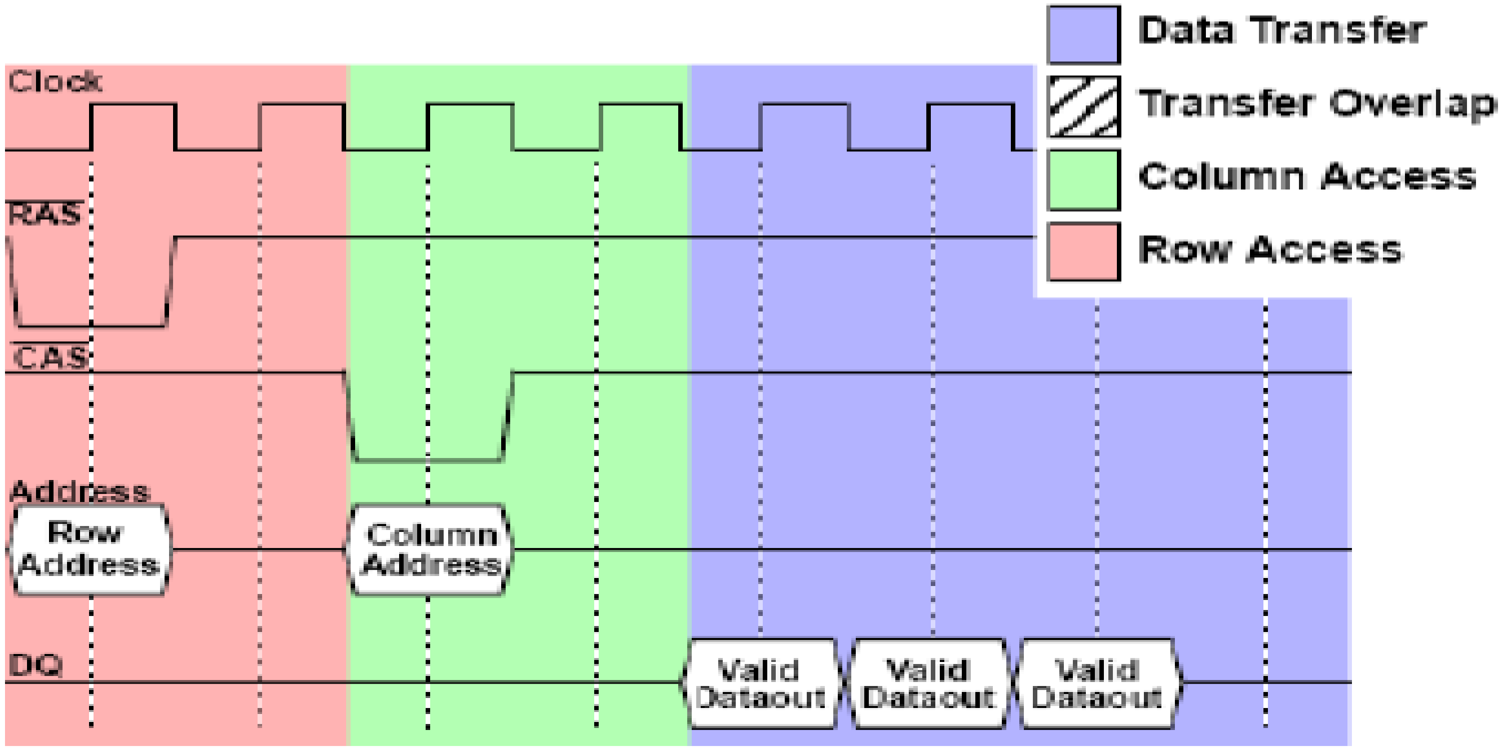
- Output register holds data during overlap of next read CAS phase with previous access data transfer

  this overlap (“pipelining”) increases throughput



SDRAM – end of 90-ties – synchronous DRAM

- SDRAM chip is equipped by counter that can be used to define continuous block length (burst) which is read together



SDRAM – the most widely used main memory technology

- SDRAM – clock frequency up to 100 MHz, 2.5V.
- DDR SDRAM – data transfer at both CLK edges, 2.5V.
- DDR2 SDRAM – lower power consumption 1.8V, frequency up to 400 MHz.
- DDR3 SDRAM – even lower power consumption at 1.5V, frequency up to 800 MHz.
- DDR4 SDRAM …
- There are also other dynamic memory types, I.e. RAMBUS, that use entirely different concept
- All these innovations are focused mainly on throughput, not on the random access latency.

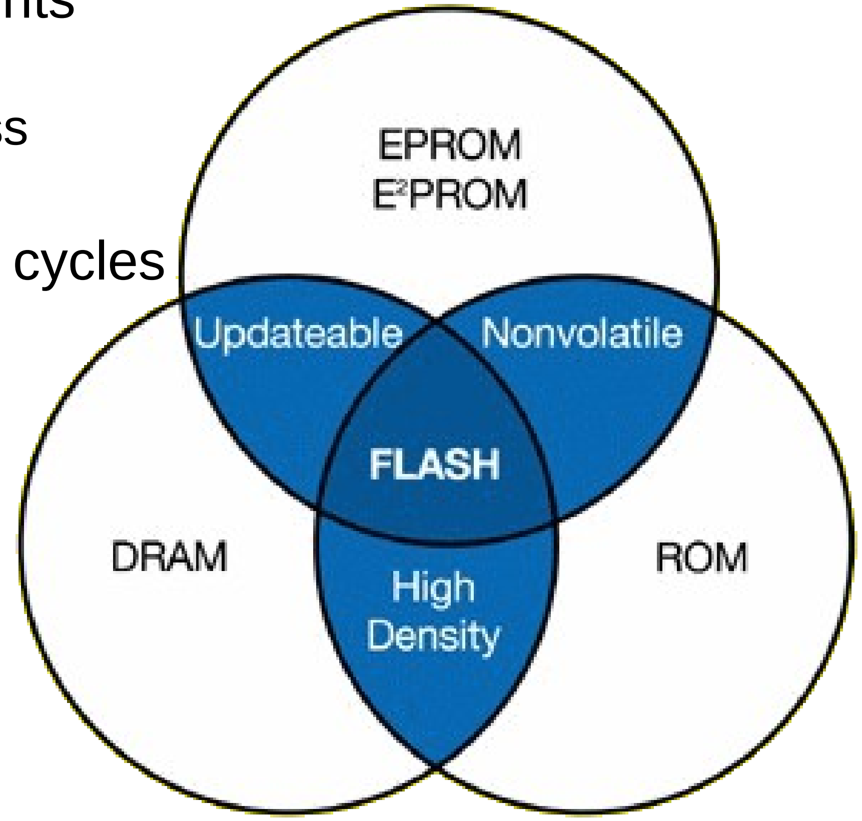
Notes for todays SDRAMs and slides

- Use of the banked architecture that enables throughput to be increased by hiding latency of the opening and closing rows. These operations can proceed in parallel on different banks (sequential and interleaved banks mapping). The change result in a minimal pin count increase that is critical for price and density.
- FIXME: More information about DDR2/3 should be added

Other memory technologies – secondary storage

Flash

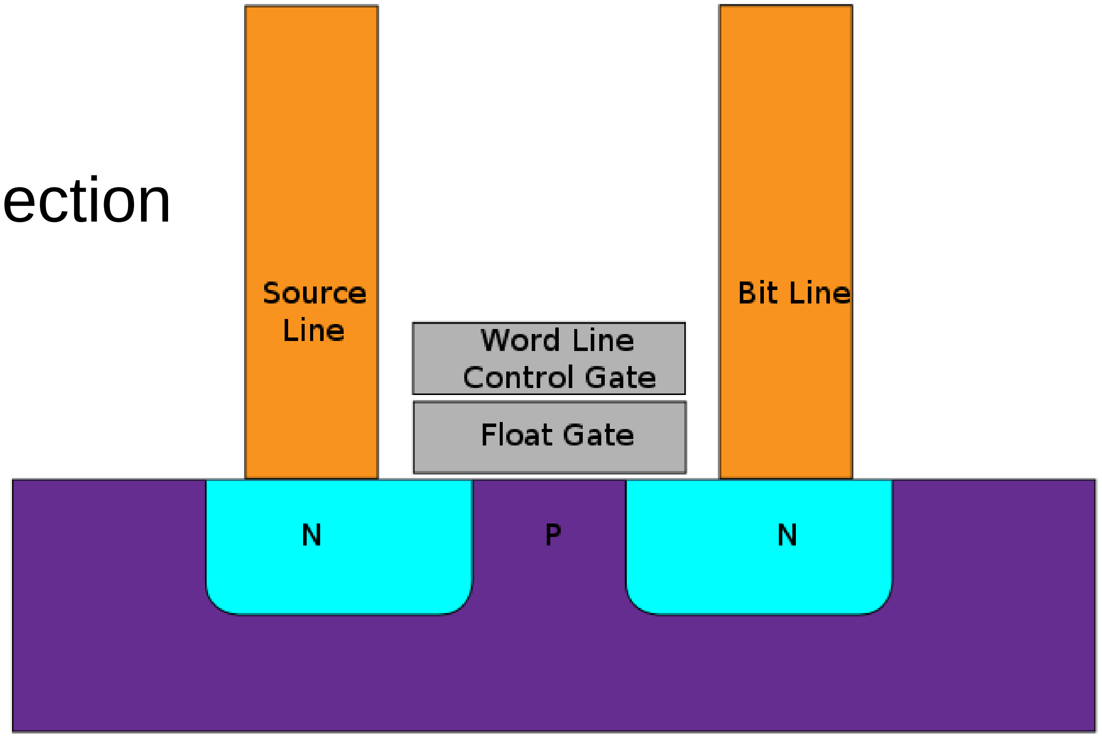
- Combines properties of E2PROM, DRAM, ROM
- Data are stored in transistor (floating gates) array (cells)
- Each block can be programmed separately
- But be aware of large erase segments
  - Nor type RAM access
  - NAND block addressing and access
- Nonvolatile computer memory
- Endures about 100 000 erase-write cycles
- Read access time (50 - 110 ns)
- Writes are slow, erase even slower
- Data retention is 10 or more years
- Uses:
  - memory cards
  - USB flash disk
  - memory chips
  - SSD disk



Flash memory cell

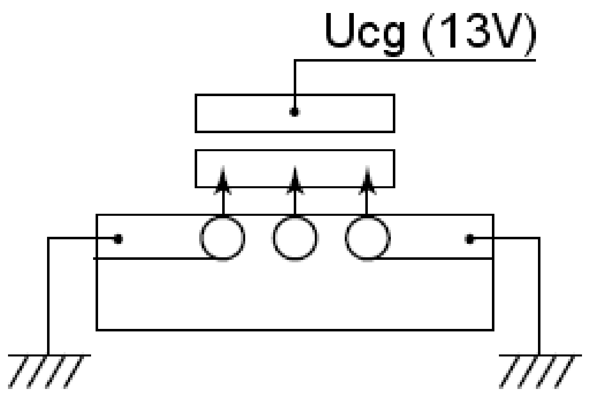
- Modified MOSFET transistor with electrically isolated floating gate

- Memory cell operations:
  - Programming
    - F-N tunneling
    - Hot-carrier injection
  - Erase
  - Read



Programming – Fowler-Nordheim's tunneling

- The U_{cg}voltage is applied to the control gate
- This voltage creates an electric field that creates a potential barrier
- This barrier simplifies the way for electrons in the substrate to the floating gate
- Alternative to programing by Fowler-Nordheim's tunneling is Drain-side tunneling



Hot-carrier injection programming

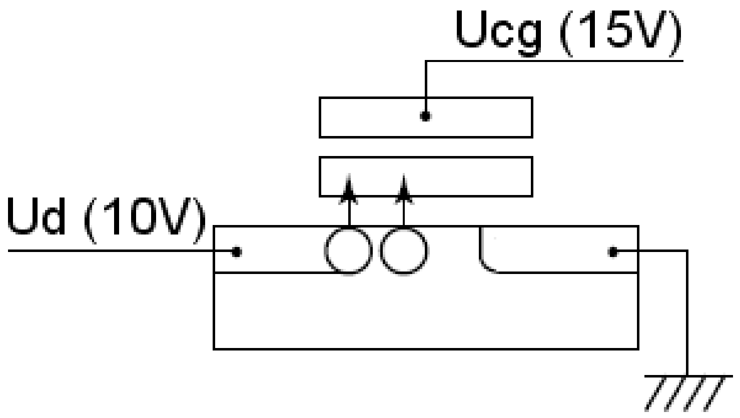
- There are two methods of Hot-carrier injection
  - a) hot-electron injection (for N type MOSFET placed on P-substrate) – used in practice for higher speed
  - b) hot-hole injection (for P type on N-substrate)

•Hot-electron injection :

1)The U_{cg} - U_d voltage creates an electric field in the semiconductor

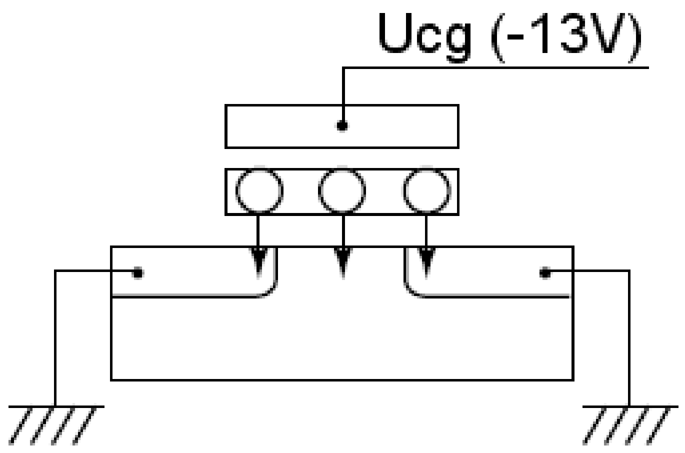
2)This field accelerates electrons from source electrode to drain electrode

3)The do not land to the drain electrode because of they have enough kinetic energy to cross isolation layer to the floating gate with higher potential



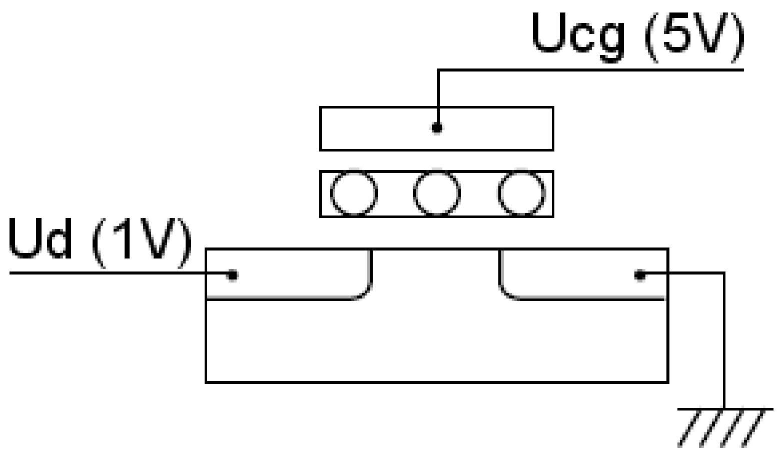
Flash cell erase

- Erase can be realized by Fowler-Nordheim's tunneling as well
- Electrons are expelled from floating point gate by opposite polarity of U_{cg} than polarity used for programming

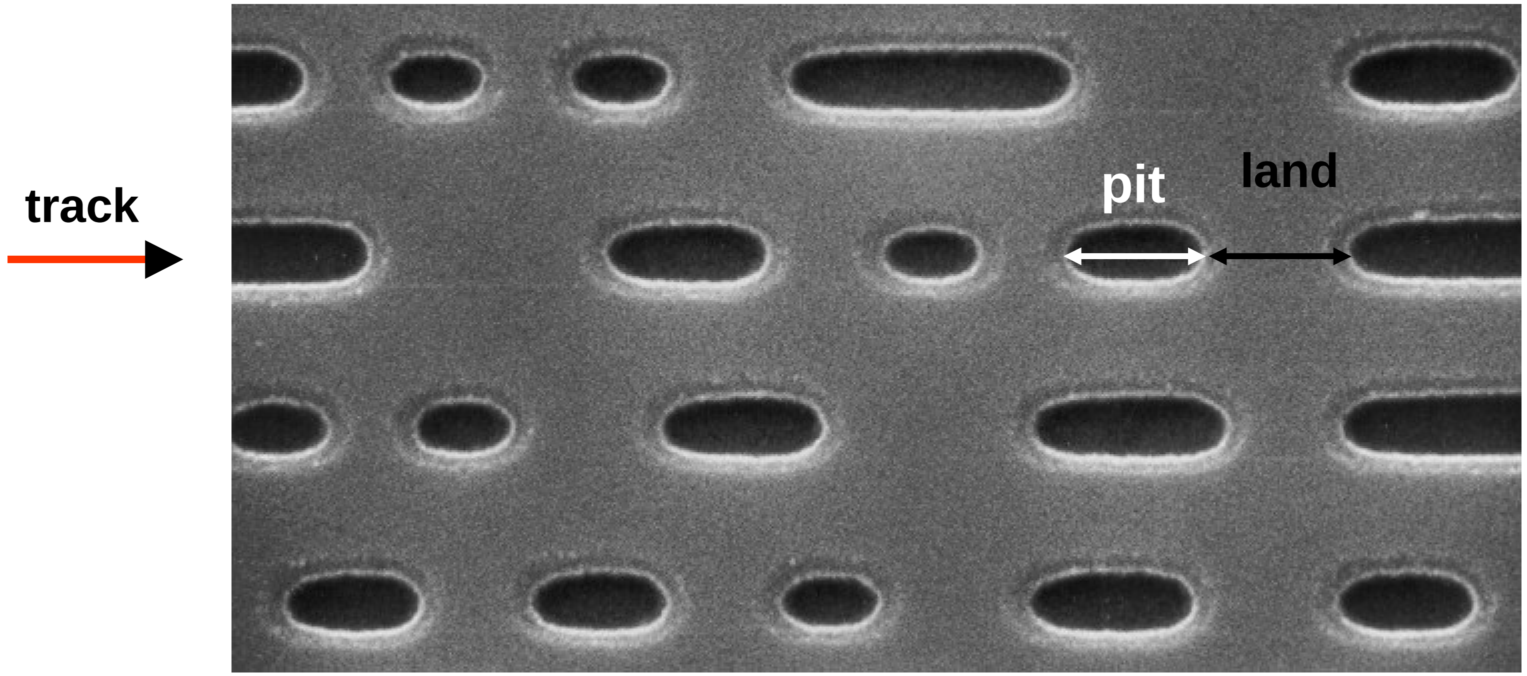


Flash cell read operation

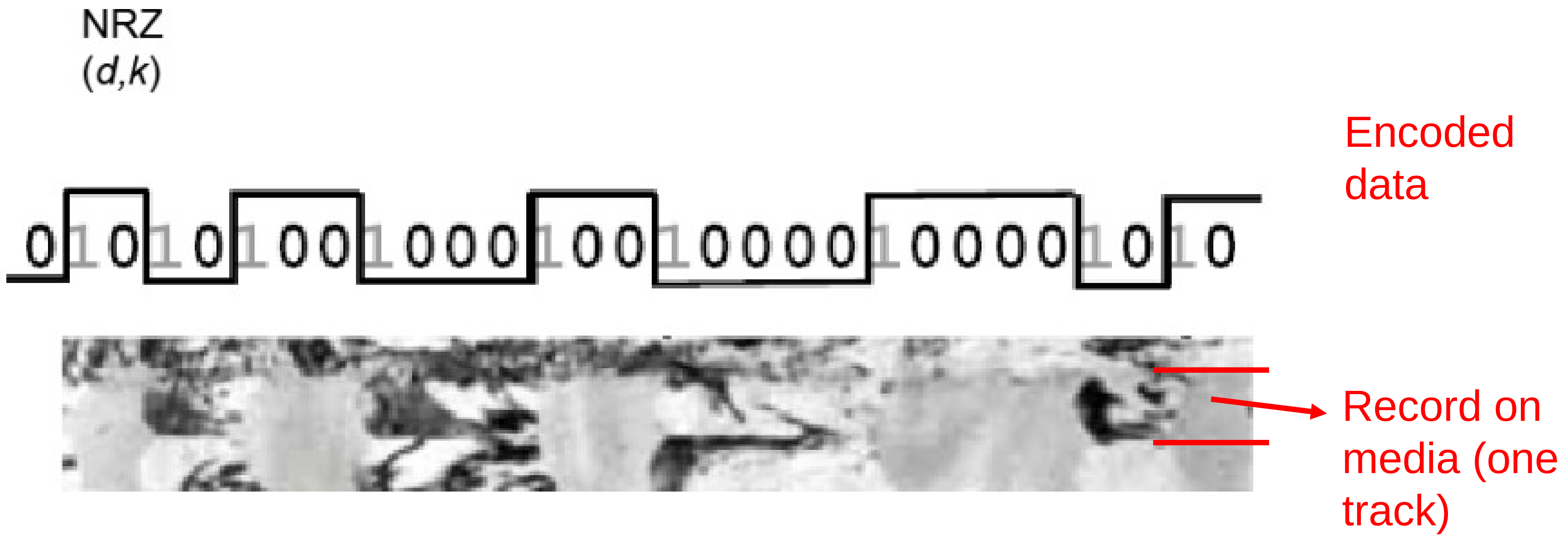
- The current flowing through transistor depends on floating gate charge value (in combination with word selector U_{cg}).
- The current on common rail is compared to same threshold(s) and converted to digital bit(s) value



Optic storage – CD-ROM – detail

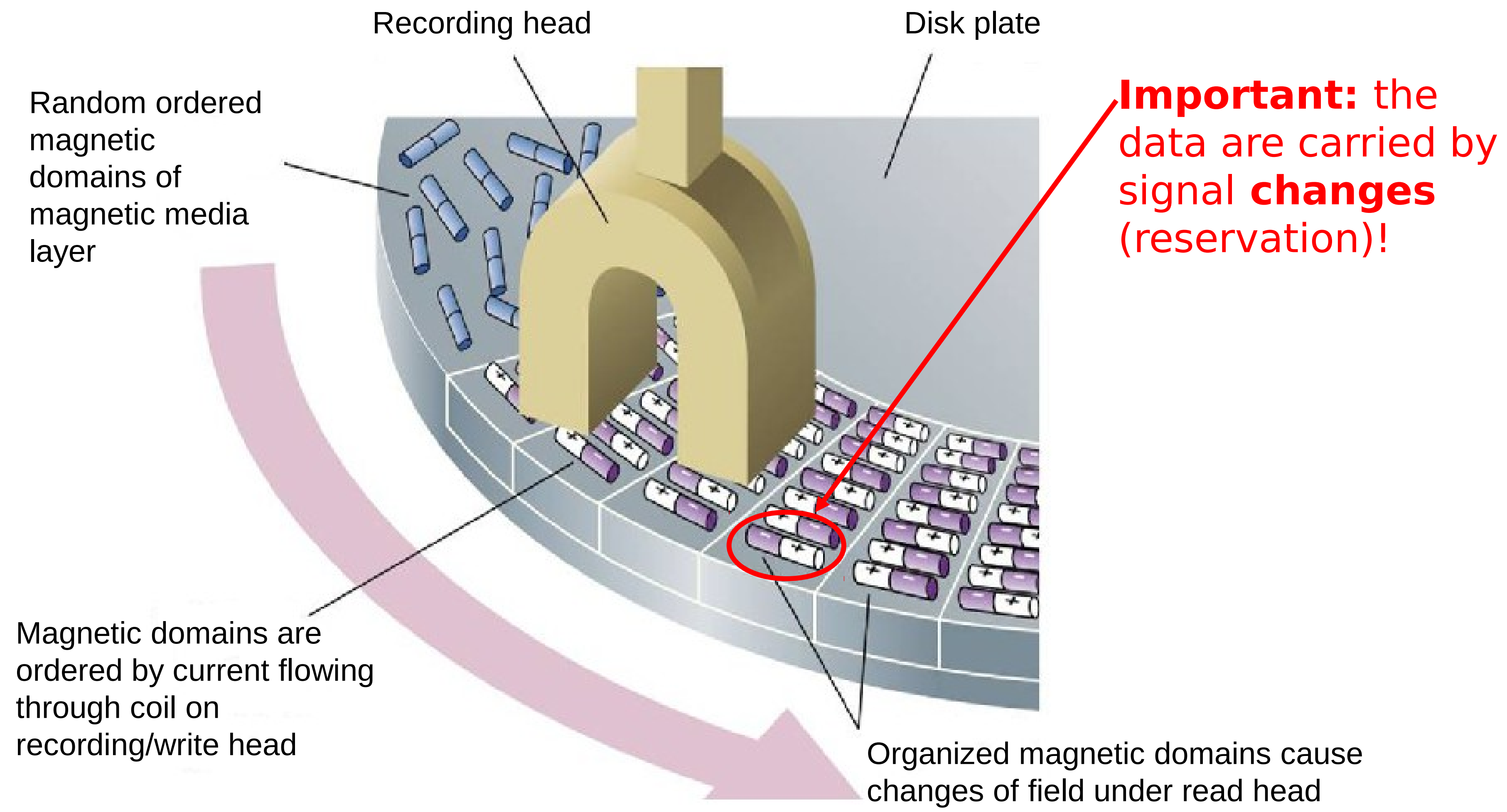


How to record „0“ and „1“?



- Ones are encoded by signal change!
- Zeros as no change. Bit stuffing etc.

Physical principle of magnetic media record


Recording head
Disk plate
Random ordered magnetic domains of magnetic media layer
Magnetic domains are ordered by current flowing through coil on recording/write head
Organized magnetic domains cause changes of field under read head
Important: the data are carried by signal **changes** (reservation)!

Quick Quiz

- Are associative memory and cache memory synonymous?

Literature to read

Read:

- What Every Programmer Should Know About Memory by Ulrich Drepper, Red Hat, Inc. http://www.akkadia.org/drepper/cpumemory.pdf
- Chapter 5 (Large and Fast: Exploiting memory hierarchy) from Hennesy, Patterson CaaQA

For brave ones

- Memory Ordering in Modern Microprocessors by Paul McKenney http://www.rdrop.com/users/paulmck/scalability/paper/ordering.2007.09.19a.pdf