Mainstream Computer System Components

(Desktop/Low-end Server)

CPU Core 2 GHz - 3.5 GHz 4-way Superscaler (RISC or RISC-core (x86):
Dynamic scheduling, Hardware speculation
Multiple FP, integer FUs, Dynamic branch prediction ...

One core or multi-core (2-8) per chip

Double Date Rate (DDR) SDRAM
One channel = 8 bytes = 64 bits wide

Current DDR3 SDRAM Example:

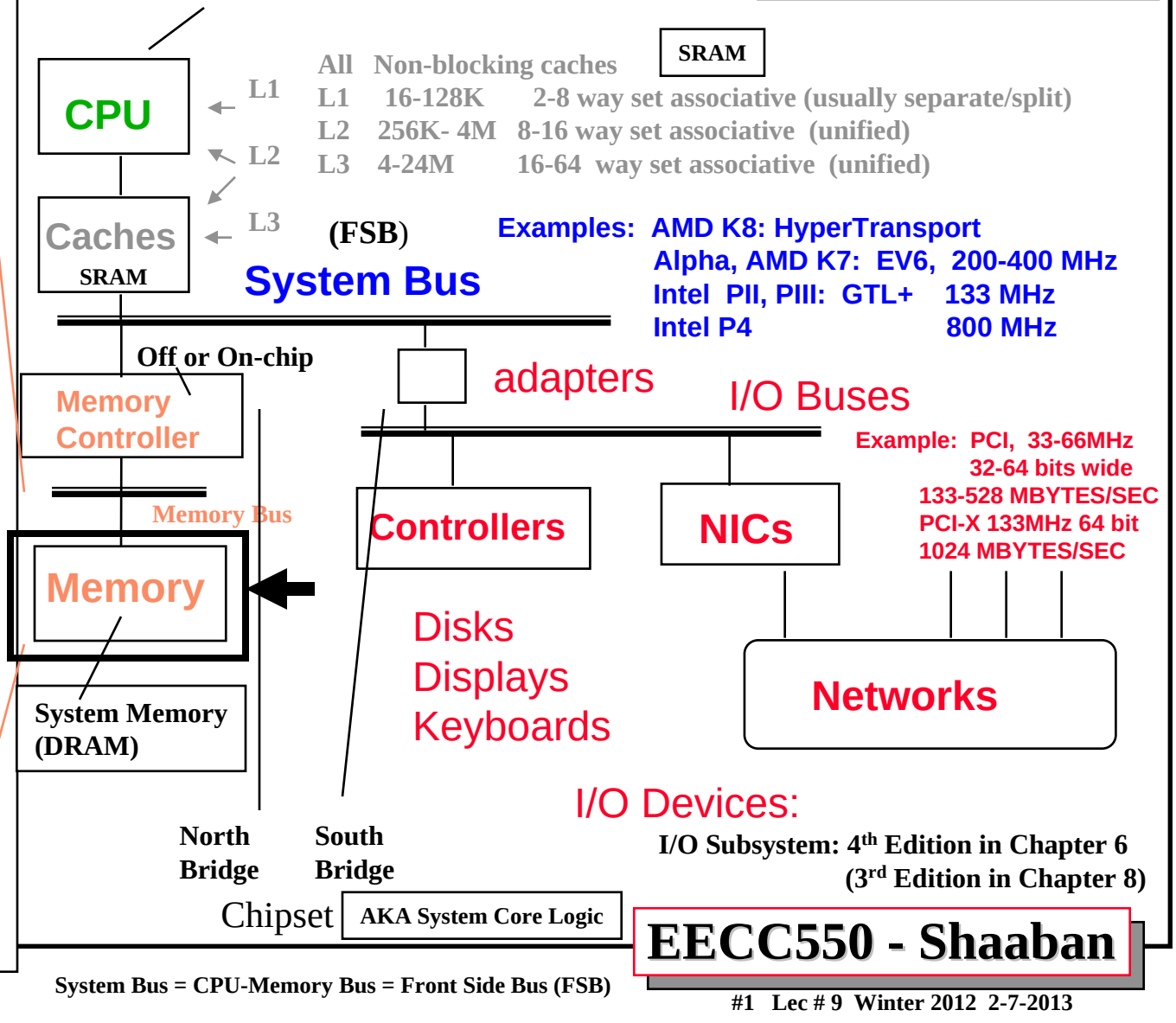
PC3-12800 (DDR3-1600)
200 MHz (internal base chip clock)
8-way interleaved (8-banks)
~12.8 GBYTES/SEC (peak)
(one 64bit channel)
~25.6 GBYTES/SEC (peak)
(two 64bit channels – e,g AMD x4, x6)
~38.4 GBYTES/SEC (peak)
(three 64bit channels – e.g Intel Core i7)

PC2-6400 (DDR2-800)
200 MHz (internal base chip clock)
64-128 bits wide
4-way interleaved (4-banks)
~6.4 GBYTES/SEC (peak)
(one 64bit channel)
~12.8 GBYTES/SEC (peak)
(two 64bit channels)

DDR SDRAM Example:

PC3200 (DDR-400)
200 MHz (base chip clock)
4-way interleaved (4-banks)
~3.2 GBYTES/SEC (peak)
(one 64bit channel)
~6.4 GBYTES/SEC
(two 64bit channels)

Single Date Rate SDRAM
PC100/PC133
100-133MHz (base chip clock)
64-128 bits wide
2-way inteleaved (2-banks)
~ 900 MBYTES/SEC peak (64bit)



I/O Subsystem: 4th Edition in Chapter 6
(3rd Edition in Chapter 8)

System Bus = CPU-Memory Bus = Front Side Bus (FSB)

The Memory Hierarchy: Main &Virtual Memory

- The Motivation for The Memory Hierarchy:
  - CPU/Memory Performance Gap
  - The Principle Of Locality → Cache $$$$$
- Cache Concepts:
  - Organization, Replacement, Operation
  - Cache Performance Evaluation: Memory Access Tree

Cache exploits access locality to:
- Lower AMAT by hiding long main memory access latency.
- Lower demands on main memory bandwidth.

(Desktop/Low-end Server)

- Main Memory:
  - Performance Metrics: Latency & Bandwidth
    - Key DRAM Timing Parameters
  - DRAM System Memory Generations
  - Basic Techniques for Memory Bandwidth Improvement/Miss Penalty (M) Reduction

4th Edition in 5.2 (3rd Edition in 7.3)

- Virtual Memory
  - Benefits, Issues/Strategies
  - Basic Virtual → Physical Address Translation: Page Tables
  - Speeding Up Address Translation: Translation Look-aside Buffer (TLB)

4th Edition in 5.4 (3rd Edition in 7.4)

Addressing The CPU/Memory Performance Gap:

Memory Access Latency Reduction & Hiding Techniques

Memory Latency Reduction Techniques:

Reduce it!

- **Faster Dynamic RAM (DRAM) Cells:** Depends on VLSI processing technology.
- **Wider Memory Bus Width:** Fewer memory bus accesses needed (e.g 128 vs. 64 bits)
- **Multiple Memory Banks:**
  - At DRAM chip level (SDR, DDR, DDR2 SDRAM), module or channel levels.

Basic Memory Bandwidth Improvement/Miss Penalty Reduction Techniques

- **Integration of Memory Controller with Processor:** e.g AMD's current processor architecture
- **New Emerging Faster RAM Technologies:** e.g. Magnetoresistive Random Access Memory (MRAM)

Memory Latency Hiding Techniques:

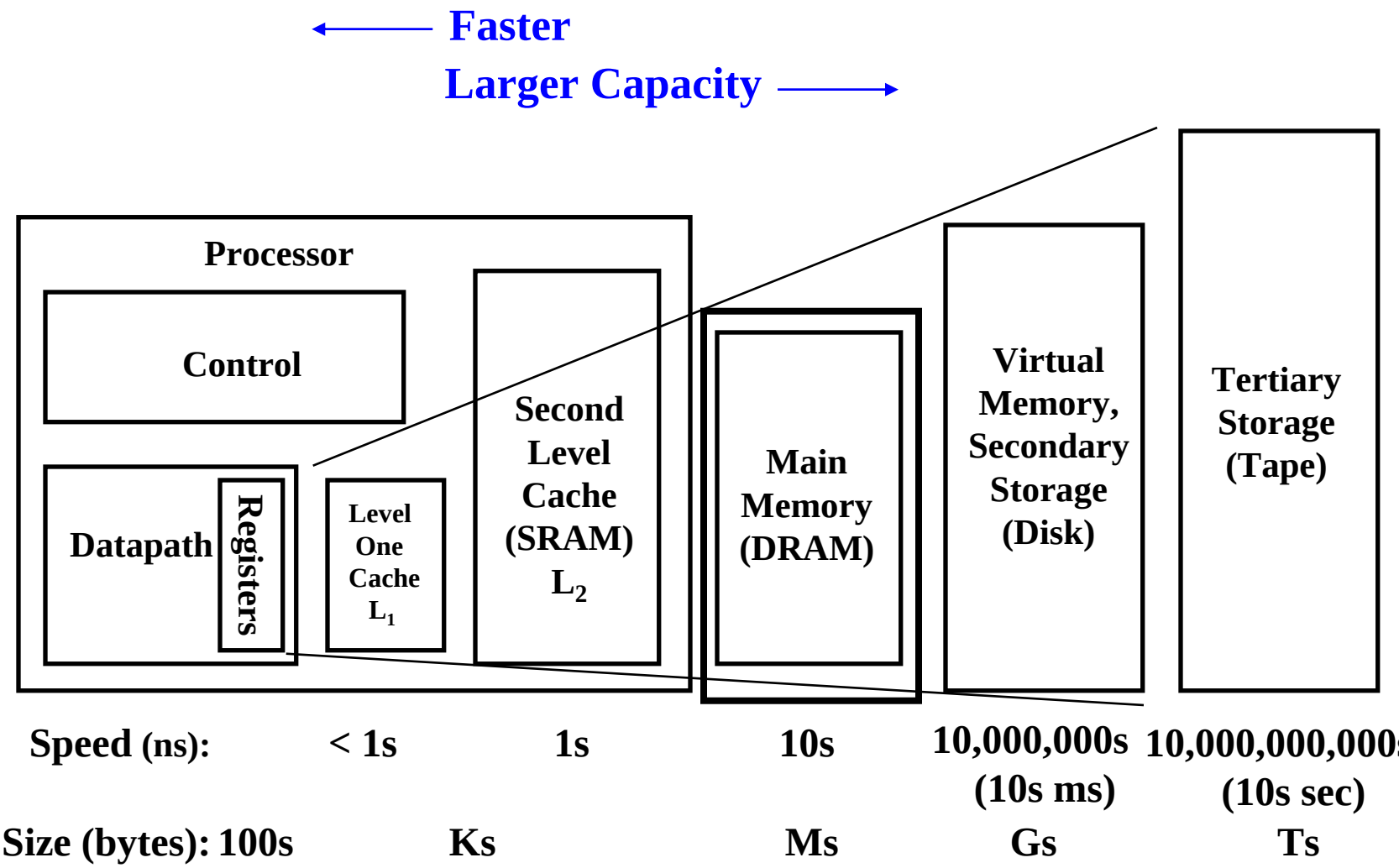
Hide it!

- **Memory Hierarchy**: One or more levels of smaller and faster memory (SRAM-based cache) on- or off-chip that exploit program access locality to hide long main memory latency.

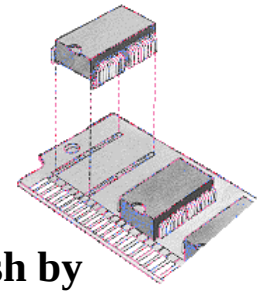
Lecture 8

- **Pre-Fetching:** Request instructions and/or data from memory before actually needed to hide long memory access latency.

A Typical Memory Hierarchy



Main Memory



- **Main memory generally utilizes Dynamic RAM (DRAM),**
  which use a single transistor to store a bit, but require a periodic data refresh by reading every row increasing cycle time.

DRAM: Slow but high density

- **Static RAM may be used for main memory if the added expense, low density, high power consumption, and complexity is feasible (e.g. Cray Vector Supercomputers).**

SRAM: Fast but low density

- **Main memory performance is affected by:**
  - **Memory latency: Affects cache miss penalty, M. Measured by:**
    - **Memory Access time: The time it takes between a memory access request is issued to main memory and the time the requested information is available to cache/CPU.**
    - **Memory Cycle time: The minimum time between requests to memory**

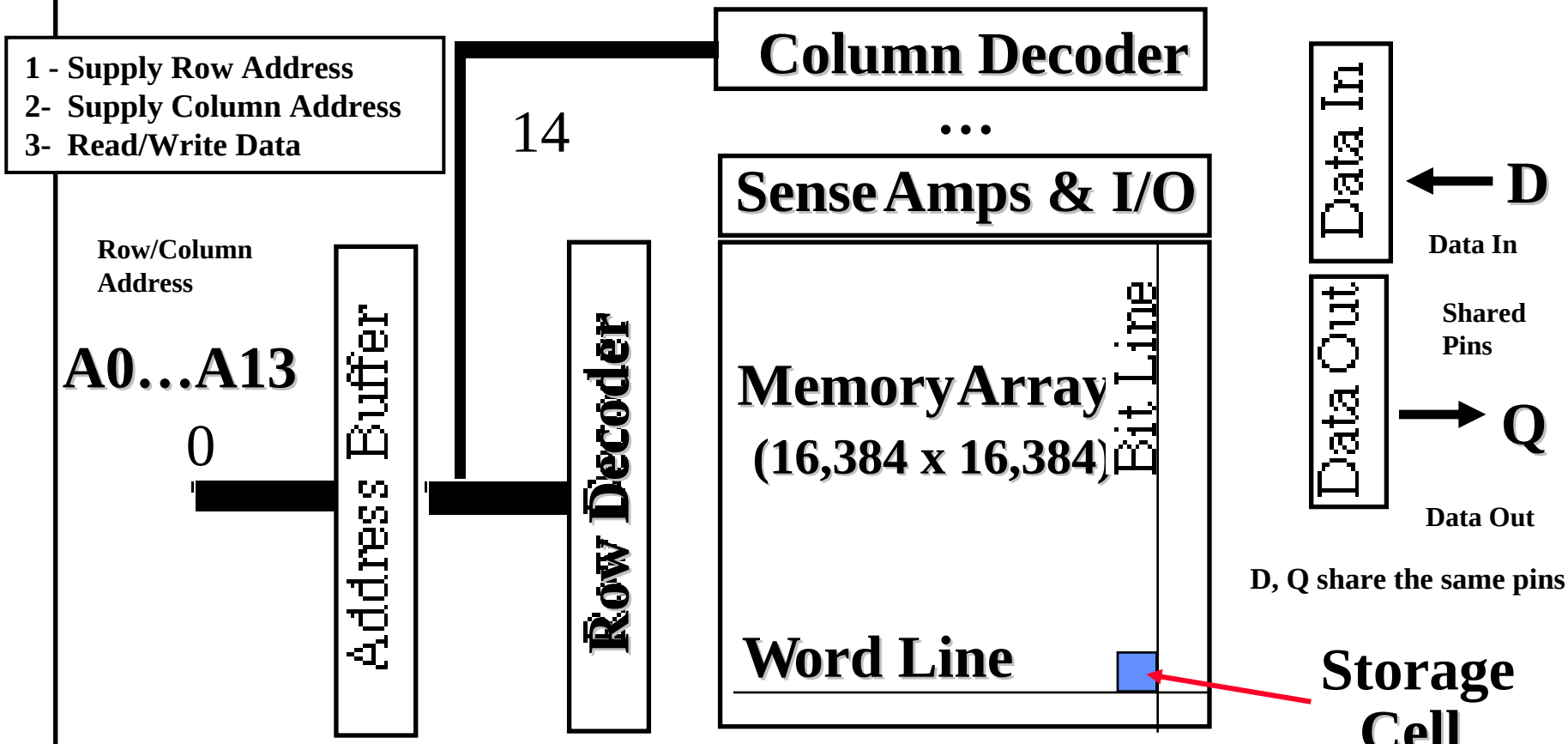
    (greater than access time in DRAM to allow address lines to be stable)
  - **Peak Memory bandwidth: The maximum sustained data transfer rate between main memory and cache/CPU.**
    - In current memory technologies (e.g Double Data Rate SDRAM) published peak memory bandwidth does not take account most of **the memory access latency**.
    - This leads to achievable realistic memory bandwidth < peak memory bandwidth

Or maximum effective memory bandwidth

4th Edition: Chapter 5.2
3rd Edition: Chapter 7.3

Logical Dynamic RAM (DRAM) Chip Organization (16 Mbit)

Typical DRAM access time = 80 ns or more (non ideal)



Basic Steps:

Control Signals:

1 - Row Access Strobe (RAS): Low to latch row address
2- Column Address Strobe (CAS): Low to latch column address
3- Write Enable (WE) or Output Enable (OE)
4- Wait for data to be ready

A periodic data refresh is required by reading every bit

1 - Supply Row Address 2- Supply Column Address 3- Get Data

Four Key DRAM Timing Parameters

1 • t_{RAC}: Minimum time from RAS (Row Access Strobe) line falling (activated) to the valid data output.
  - Used to be quoted as the nominal speed of a DRAM chip
  - For a typical 64Mb DRAM t_{RAC} = 60 ns

2 • t_{RC}: Minimum time from the start of one row access to the start of the next (memory cycle time).
  - $t_{RC} = t_{RAC}$ + RAS Precharge Time
  - t_{RC} = 110 ns for a 64Mbit DRAM with a t_{RAC} of 60 ns

3 • t_{CAC}: Minimum time from CAS (Column Access Strobe) line falling to valid data output.
  - 12 ns for a 64Mbit DRAM with a t_{RAC} of 60 ns

4 • t_{PC}: Minimum time from the start of one column access to the start of the next.
  - $t_{PC} = t_{CAC}$ + CAS Precharge Time
  - About 25 ns for a 64Mbit DRAM with a t_{RAC} of 60 ns

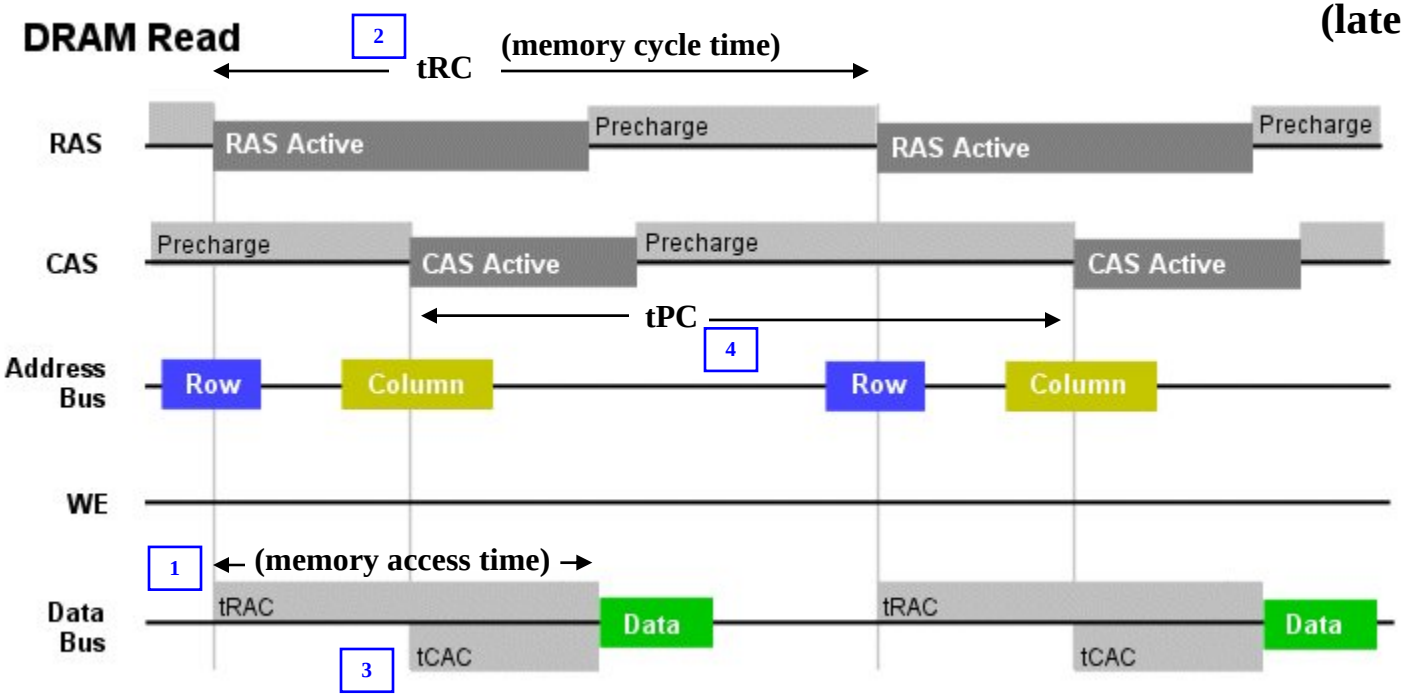
1 - Supply Row Address 2- Supply Column Address 3- Get Data

Simplified Asynchronous DRAM Read Timing

Memory Cycle Time = tRC = tRAC + RAS Precharge Time

Non-burst Mode Memory Access Example

(late 70s)



1 t_{RAC}: Minimum time from RAS (Row Access Strobe) line falling to the valid data output.

2 t_{RC}: Minimum time from the start of one row access to the start of the next (memory cycle time).

3 t_{CAC}: minimum time from CAS (Column Access Strobe) line falling to valid data output.

4 t_{PC}: minimum time from the start of one column access to the start of the next.

Peak Memory Bandwidth = Memory bus width / Memory cycle time

Example: Memory Bus Width = 8 Bytes Memory Cycle time = 200 ns

Peak Memory Bandwidth = 8 / 200 x 10^{-9} = 40 x 10^{6} Bytes/sec

Source: http://arstechnica.com/paedia/r/ram_guide/ram_guide.part2-1.html

Simplified DRAM Speed Parameters

- **Row Access Strobe (RAS)Time: (similar to t_{RAC}):**
  - Minimum time from RAS (Row Access Strobe) line falling (activated) to the first valid data output.
  - A major component of memory latency.
  - Only improves ~ 5% every year.

And cache miss penalty M

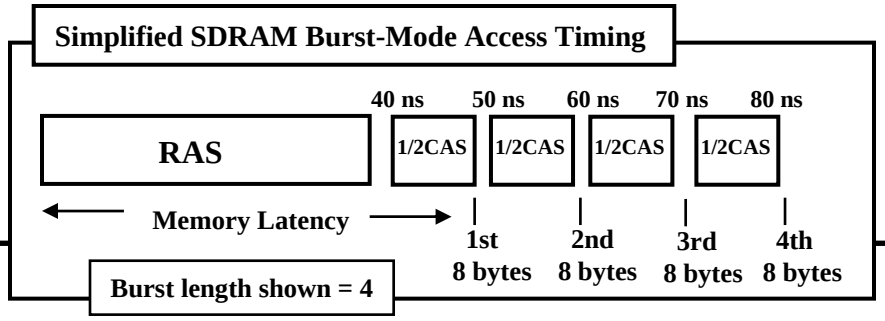
Effective

- **Column Access Strobe (CAS) Time/data transfer time: (similar to t_{CAC})**
  - The minimum time required to read additional data by changing column address while keeping the same row address.
  - Along with memory bus width, determines peak memory bandwidth.
    - e.g For SDRAM Peak Memory Bandwidth = Bus Width /(0.5 x t_{CAC})

Example

For PC100 SDRAM Memory bus width = 8 bytes t_{CAC} = 20ns

Peak Bandwidth = 8 x 100x10^6 = 800 x 10^6 bytes/sec



For PC100 SDRAM: Clock = 100 MHz

DRAM Generations

Year	Size	RAS (ns)	Effective CAS (ns)	~ RAS+ Cycle Time	Memory Type	
1980	64 Kb	150-180	75	250 ns	Page Mode	Asynchronous DRAM
1983	256 Kb	120-150	50	220 ns	Page Mode	
1986	1 Mb	100-120	25	190 ns		
1989	4 Mb	80-100	20	165 ns	Fast Page Mode	
1992	16 Mb	60-80	15	120 ns	EDO	
1996	64 Mb	50-70	12	110 ns	PC66 SDRAM	Synchronous DRAM
1998	128 Mb	50-70	10	100 ns	PC100 SDRAM	
2000	256 Mb	45-65	7	90 ns	PC133 SDRAM	
2002	512 Mb	40-60	5	80 ns	PC2700 DDR SDRAM	
	8000:1 (Capacity)		15:1 (~bandwidth)	3:1 (Latency)		
2012	2 Gb		Peak			

PC3200 DDR (2003)

↓

DDR2 SDRAM (2004)

↓

DDR3 SDRAM (2008-?)

A major factor in cache miss penalty M

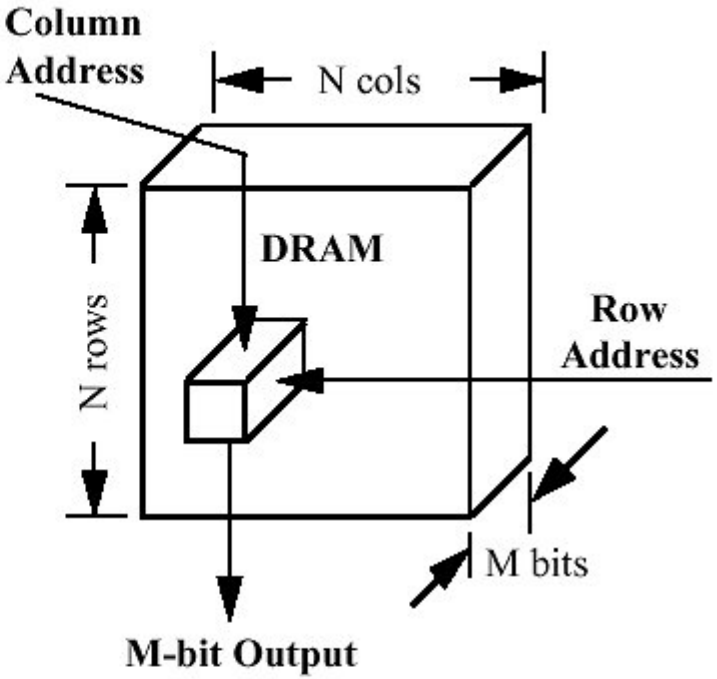
Asynchronous DRAM:

Page Mode DRAM (Early 80s)

Last system memory type to use non-burst access mode

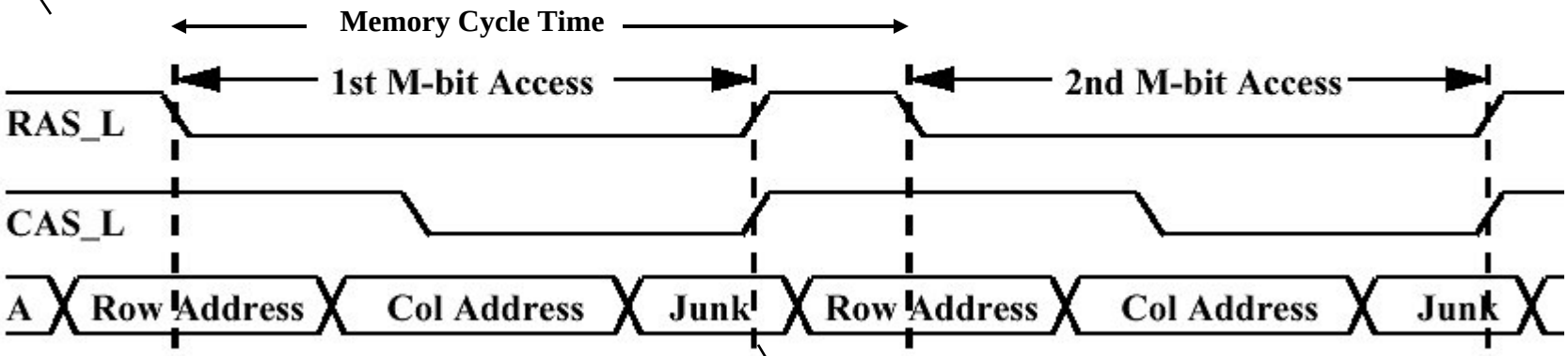
° Regular DRAM Organization:

- N rows x N column x M-bit
- Read & Write M-bit at a time
- Each M-bit access requires a RAS / CAS cycle



1 - Supply Row Address
2- Supply Column Address
3- Read/Write Data

Non-burst Mode Memory Access



1 - Supply Row Address 2- Supply Column Address 3- Get Data

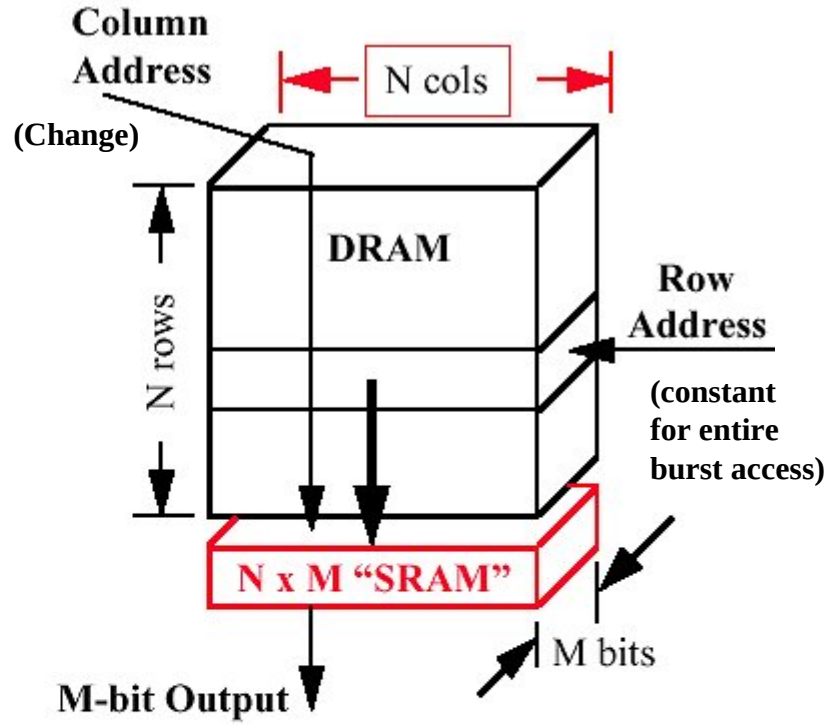
Asynchronous DRAM:

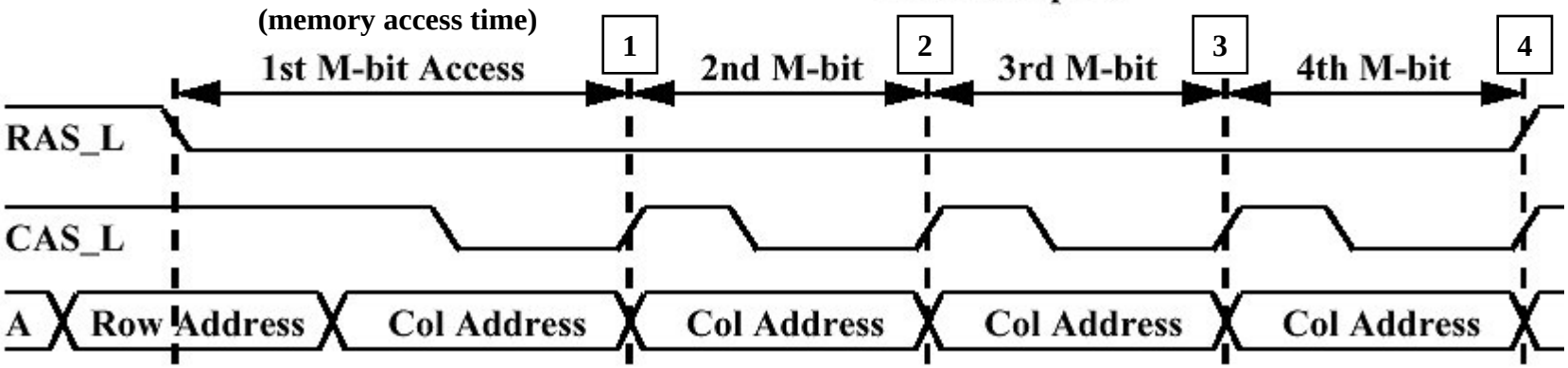
FPM

Fast Page Mode DRAM (late 80s)

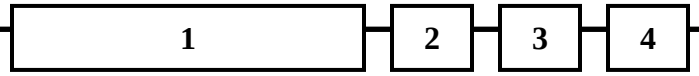
- Fast Page Mode DRAM
  - N x M "SRAM" to save a row
- After a row is read into the register
  - Only CAS is needed to access other M-bit blocks on that row
  - RAS_L remains asserted while CAS_L is toggled

- The first "burst mode" DRAM





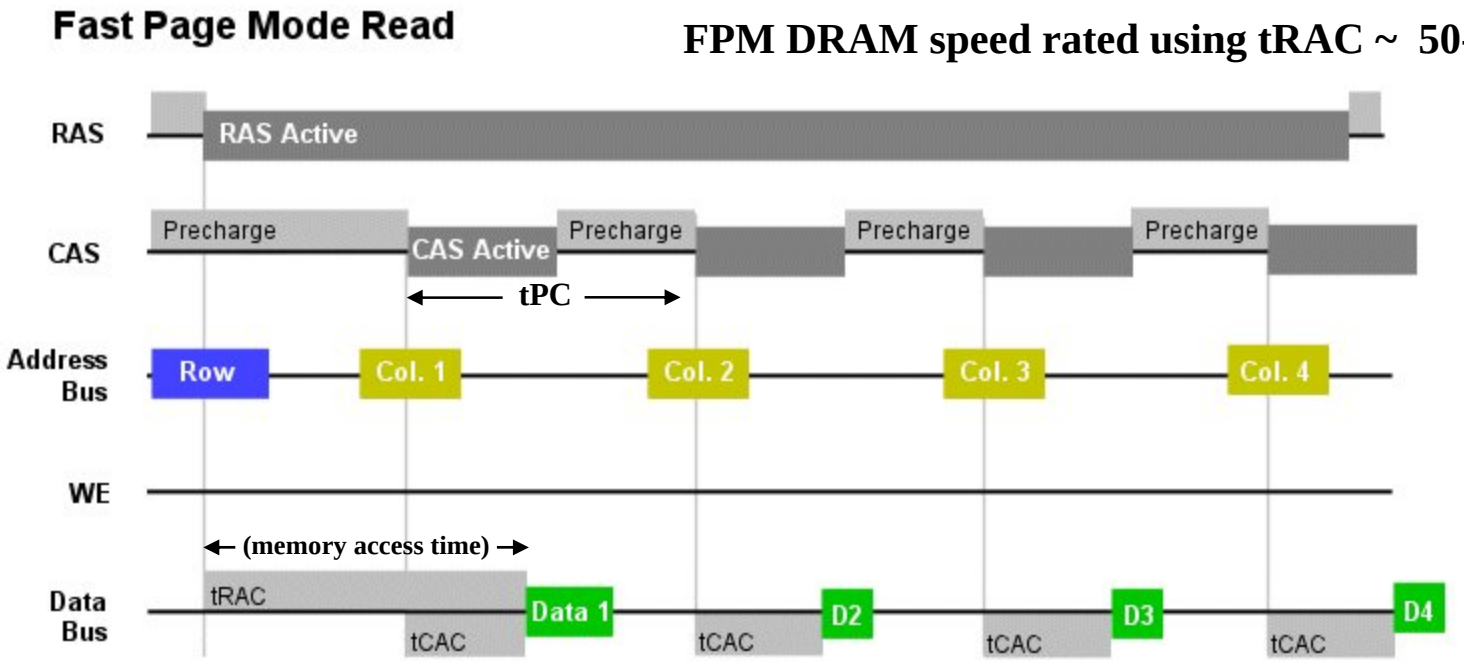
A read burst of length 4 shown



Burst Mode Memory Access

Simplified Asynchronous Fast Page Mode (FPM) DRAM Read Timing

(late 80s)



A read burst of length 4 shown

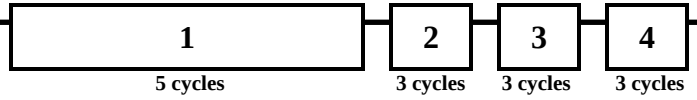
Typical timing at 66 MHz : 5-3-3-3 (burst of length 4)

For bus width = 64 bits = 8 bytes cache block size = 32 bytes

It takes = 5+3+3+3 = 14 memory cycles or 15 ns x 14 = 210 ns to read 32 byte block

Miss penalty for CPU running at 1 GHz = M = 15 x 14 = 210 CPU cycles

One memory cycle at 66 MHz = 1000/66 = 15 CPU cycles at 1 GHz



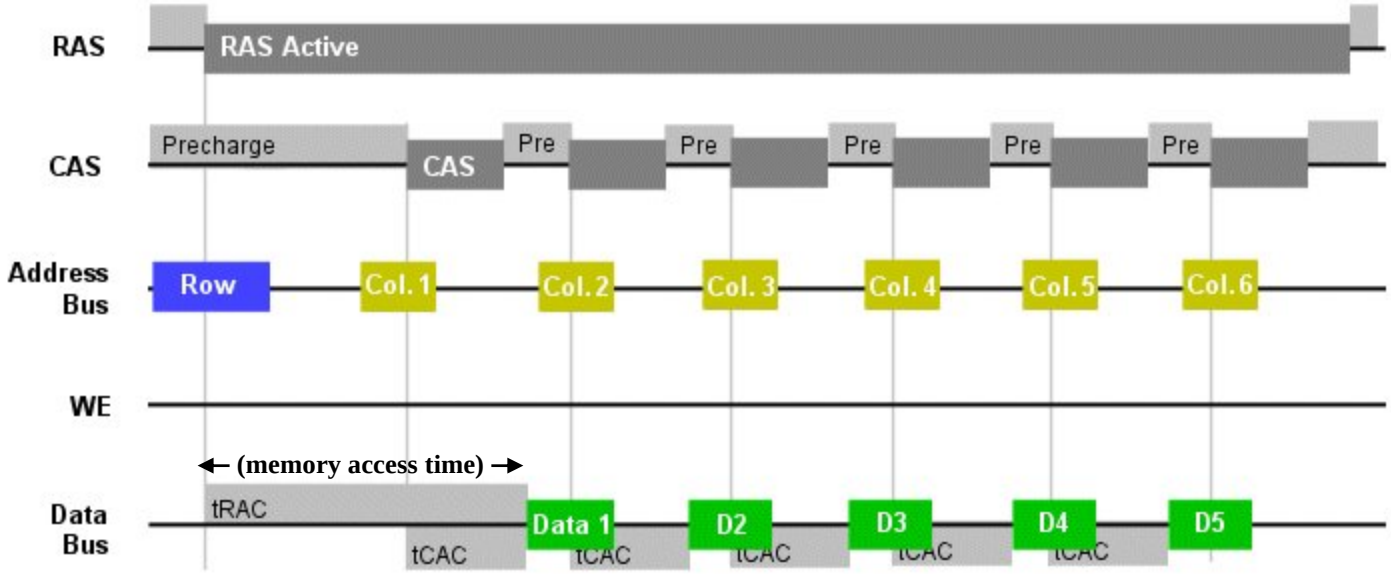
Simplified Asynchronous Extended Data Out (EDO) DRAM Read Timing

(early 90s)

- **Extended Data Out DRAM operates in a similar fashion to Fast Page Mode DRAM except putting data from one read on the output pins at the same time the column address for the next read is being latched in.**

EDO Read

EDO DRAM speed rated using tRAC ~ 40-60ns



Typical timing at 66 MHz : 5-2-2-2 (burst of length 4)

For bus width = 64 bits = 8 bytes Max. Bandwidth = 8 x 66 / 2 = 264 Mbytes/sec

It takes = 5+2+2+2 = 11 memory cycles or 15 ns x 11 = 165 ns to read 32 byte cache block

Minimum Read Miss penalty for CPU running at 1 GHz = M = 11 x 15 = 165 CPU cycles

One memory cycle at 66 MHz = 1000/66 = 15 CPU cycles at 1 GHz

Source: http://arstechnica.com/paedia/r/ram_guide/ram_guide.part2-1.html

Basic Memory Bandwidth Improvement/Miss Penalty (M) Latency Reduction Techniques

1 **Wider Main Memory (CPU-Memory Bus):** i.e wider FSB

Memory bus width is increased to a number of words (usually up to the size of a cache block).

- Memory bandwidth is proportional to memory bus width.
  - e.g Doubling the width of cache and memory doubles potential memory bandwidth available to the CPU. e.g 128 bit (16 bytes) memory bus instead of 64 bits (8 bytes) – now 24 bytes (192 bits)
- The miss penalty is reduced since fewer memory bus accesses are needed to fill a cache block on a miss.

2 **Interleaved (Multi-Bank) Memory:**

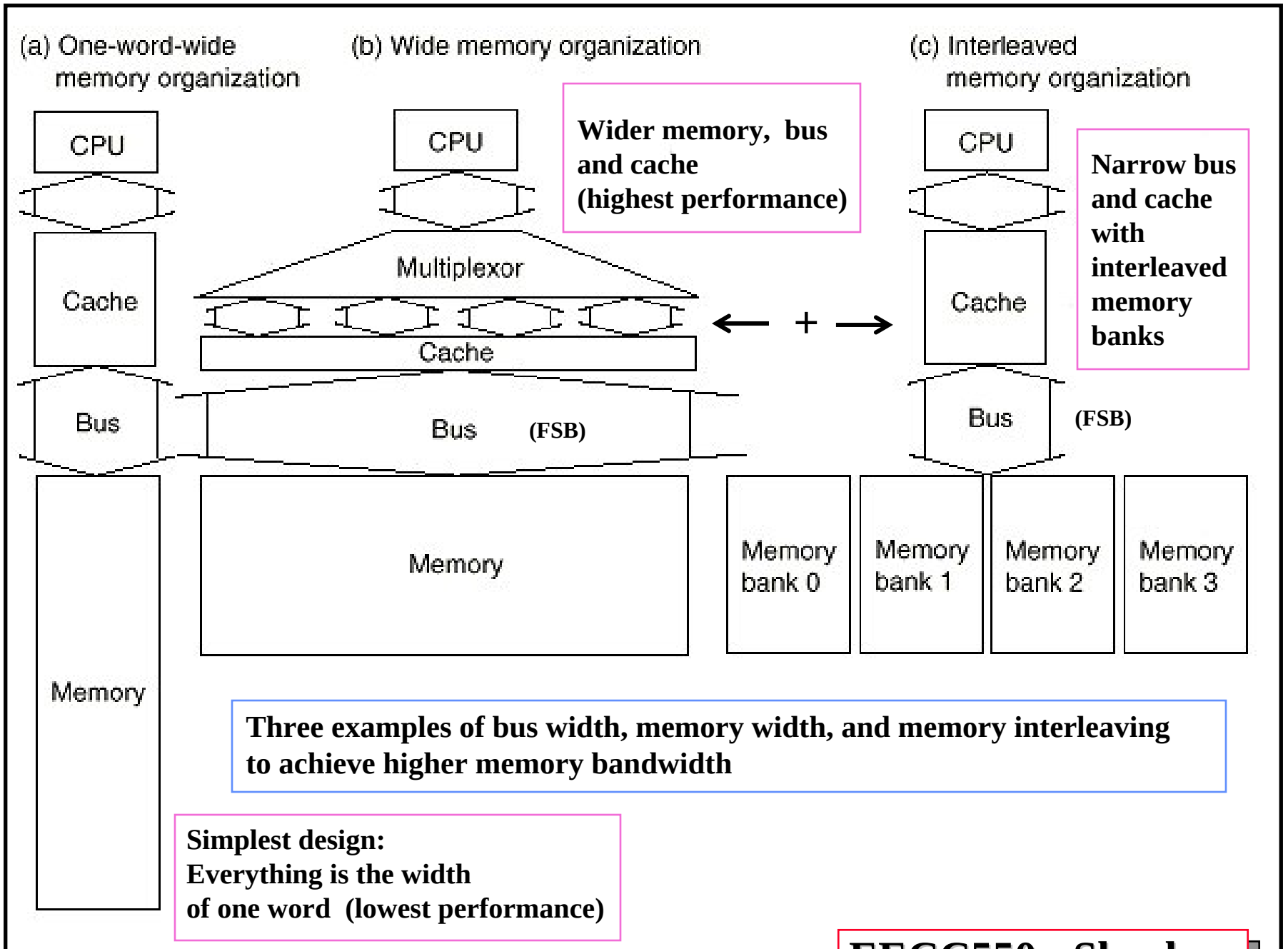
Memory is organized as a number of independent banks.

- Multiple interleaved memory reads or writes are accomplished by sending memory addresses to several memory banks at once or pipeline access to the banks.
- Interleaving factor: Refers to the mapping of memory addressees to memory banks. Goal reduce bank conflicts.

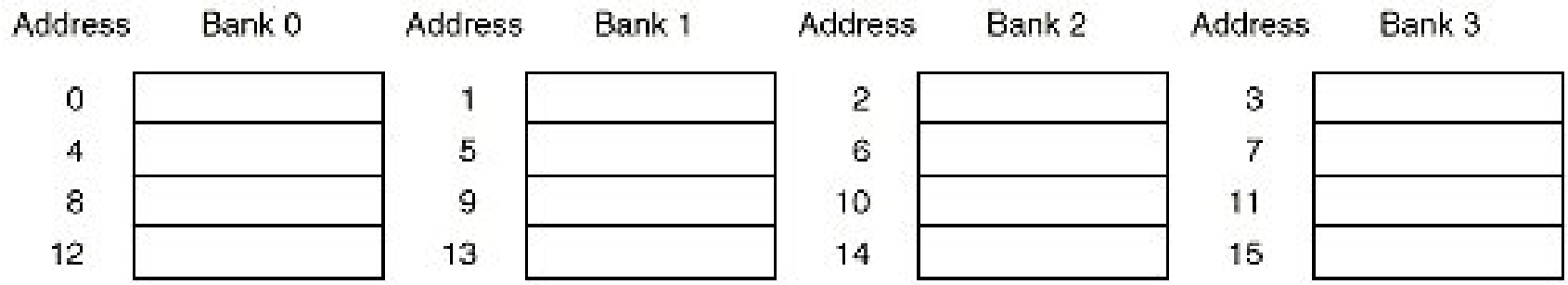
  e.g. using 4 banks (width one word), bank 0 has all words whose address is:

  (word address mod) 4 = 0

The above two techniques can also be applied to any cache level to reduce cache hit time and increase cache bandwidth.



Front Side Bus (FSB) = System Bus = CPU-memory Bus

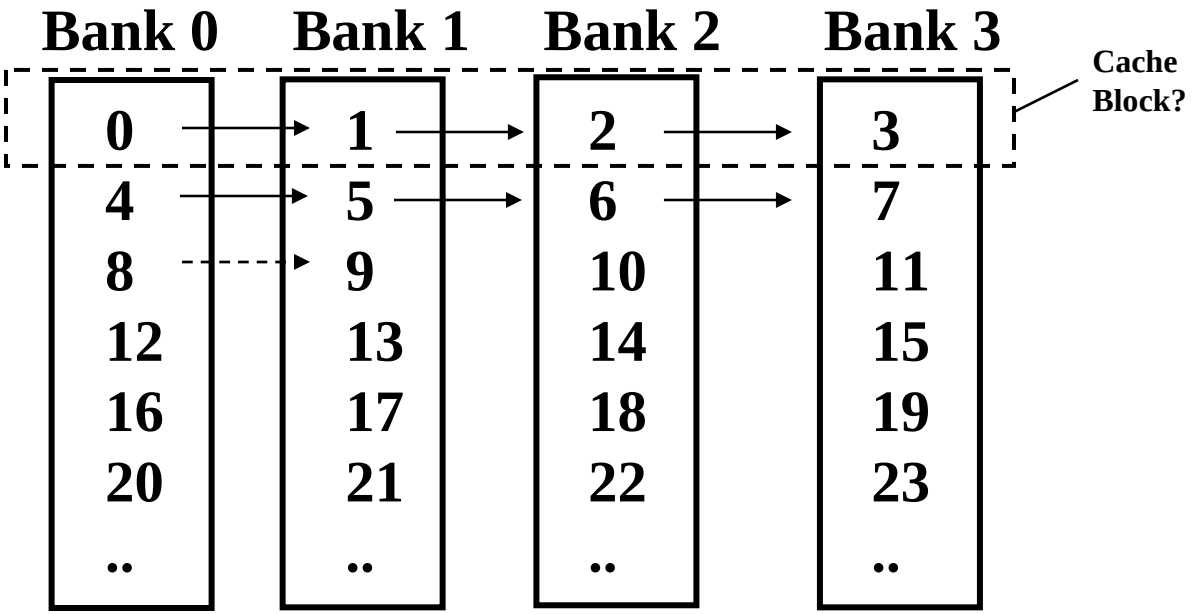


Four Way (Four Banks) Interleaved Memory

Memory Bank Number

Sequential Mapping of Memory Addresses To Memory Banks

Example



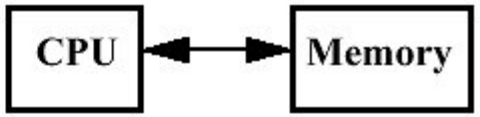
Address Within Bank

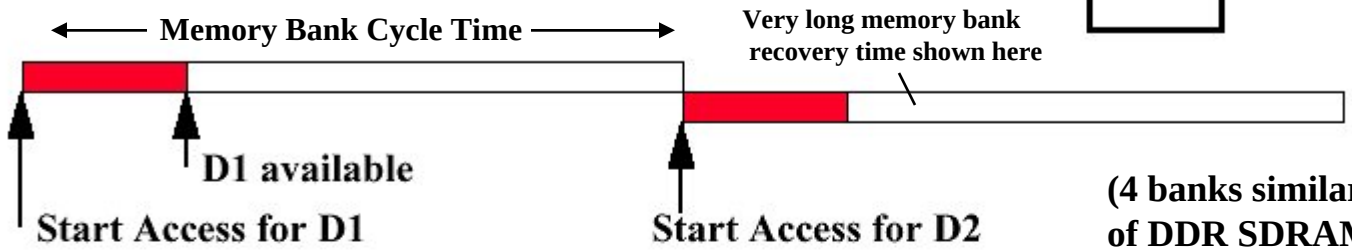
Bank Width = One Word
Bank Number = (Word Address) Mod (4)

Memory Bank Interleaving (Multi-Banked Memory)

***Can be applied at*:** 1- DRAM chip level (e.g SDRAM, DDR) 2- DRAM module level 3- DRAM channel level

Access Pattern without Interleaving: (One Memory Bank)



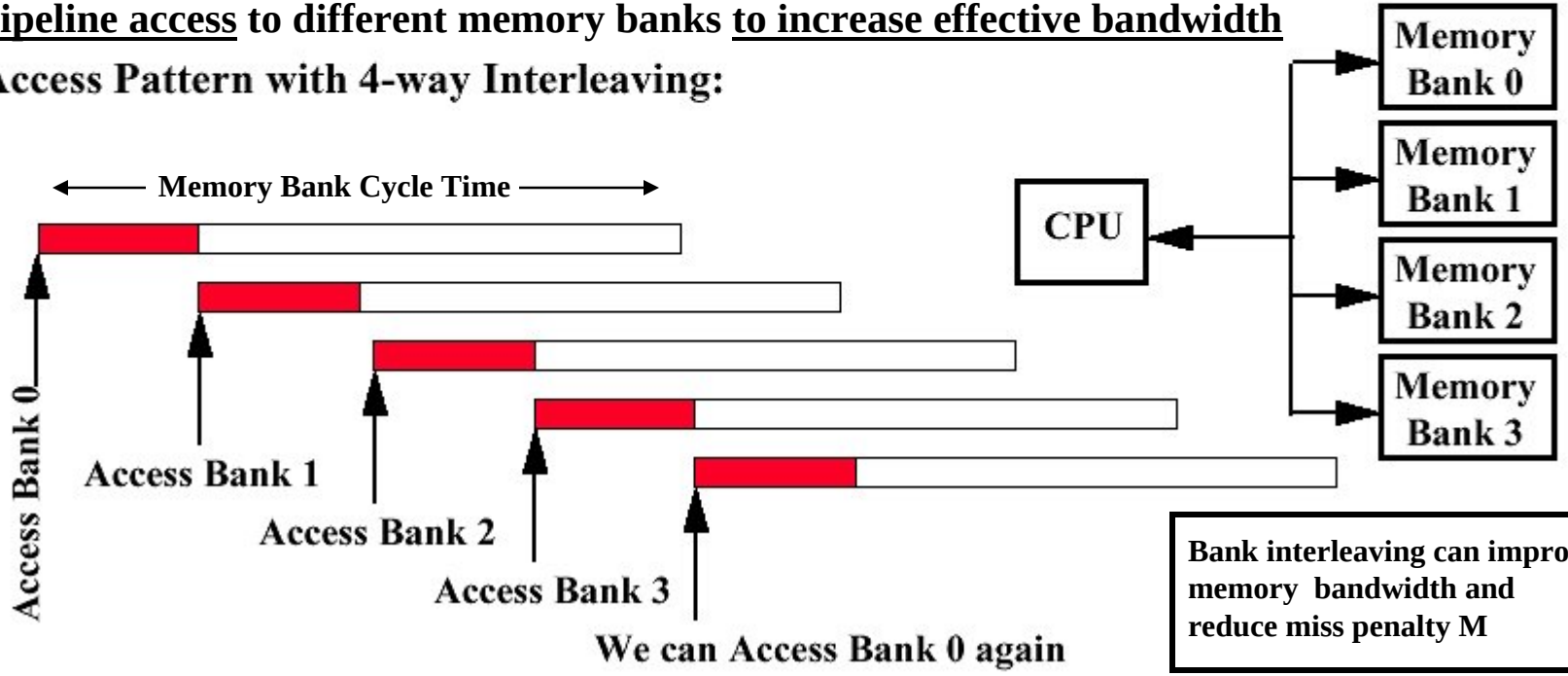


(4 banks similar to the organization of DDR SDRAM memory chips)

Also DDR2 (DDR3 increases the number to 8 banks)

Pipeline access to different memory banks to increase effective bandwidth

Access Pattern with 4-way Interleaving:



Bank interleaving can improve memory bandwidth and reduce miss penalty M

Number of banks ≥ Number of cycles to access word in a bank

Bank interleaving *does not reduce latency of accesses to the same bank*

Synchronous DRAM Generations Summary

All Use: 1- Fixed Clock Rate 2- Burst-Mode 3- Multiple Banks per DRAM chip

For Peak Bandwidth: Initial burst latency not taken into account

	SDR (Single Data Rate) SDRAM	DDR (Double Data Rate) SDRAM		
	SDR	DDR	DDR2	DDR3
Year of Introduction	Late 1990's	2002	2004	2007
# of Banks Per DRAM Chip	2	4	4	8
Example	PC100	DDR400 (PC-3200)	DDR2-800 (PC2-6400)	DDR3-1600 (PC3-12800)
Internal Base Frequency	100 MHz	200 MHz	200 MHz	200 MHz
External Interface Frequency	100 MHz	200 MHz	400 MHz	800 MHz
Peak Bandwidth (per 8 byte module)	0.8 GB/s (8 x 0.1)	3.2 GB/s (8 x 0.2 x 2)	6.4 GB/s (8 x 0.2 x 4)	12.8 GB/s (8 x 0.2 x 8)
Latency Range	60-90 ns	45-60 ns	35-50 ns	30-45 ns

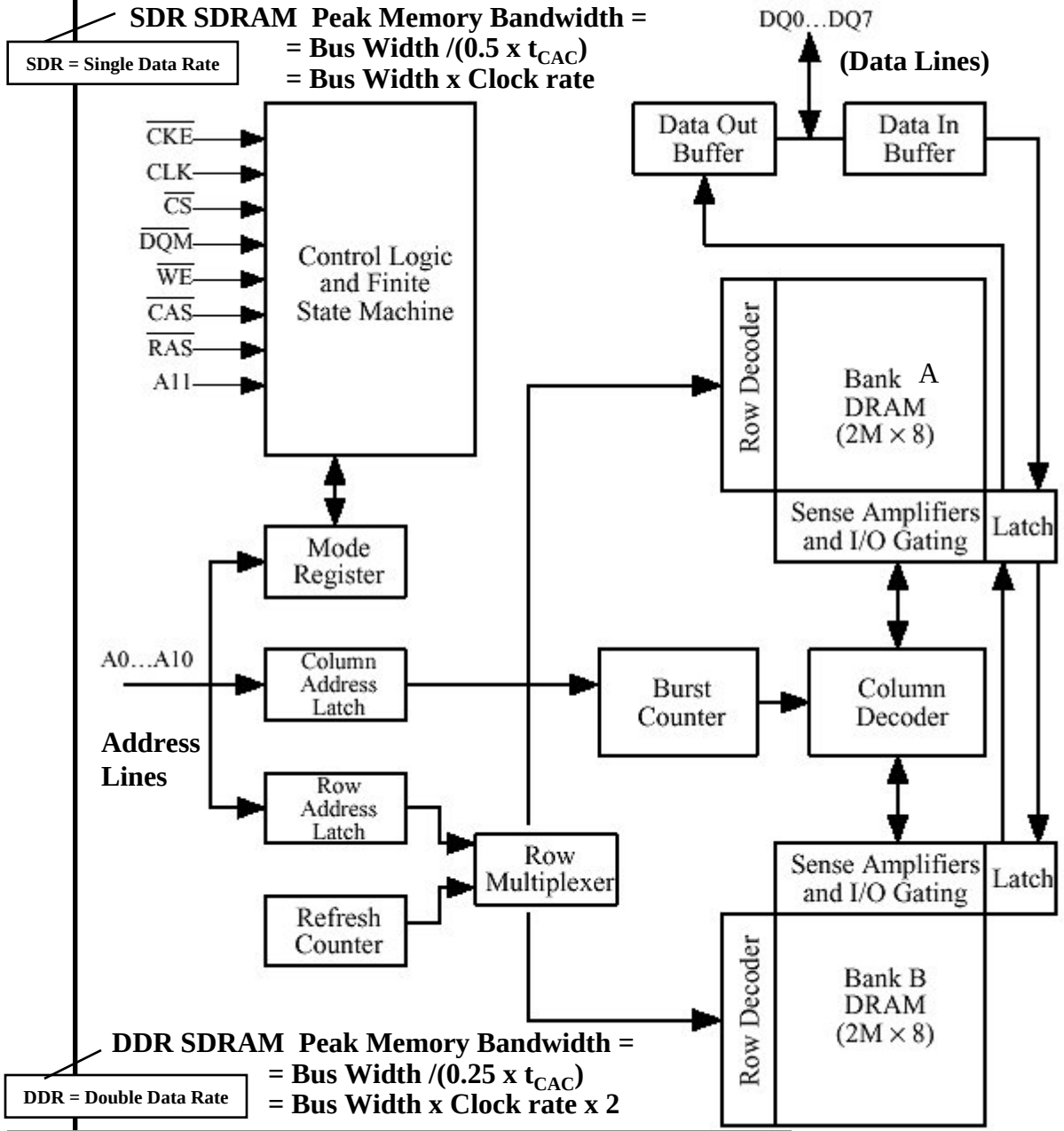
The latencies given only account for memory module latency and do not include memory controller latency or other address/data line delays. Thus realistic access latency is longer

All synchronous memory types above use burst-mode access with multiple memory banks per DRAM chip

Synchronous Dynamic RAM, (SDR SDRAM) Organization (mid 90s)

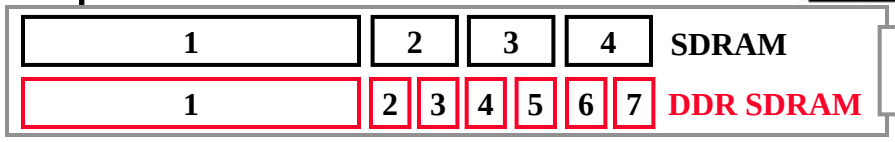
SDR = Single Data Rate

SDR SDRAM Peak Memory Bandwidth =
= Bus Width /(0.5 x t_{CAC})
= Bus Width x Clock rate



DDR SDRAM Peak Memory Bandwidth =
= Bus Width /(0.25 x t_{CAC})
= Bus Width x Clock rate x 2

DDR = Double Data Rate



Timing Comparison

SDRAM speed is rated at max. clock speed supported:
100MHZ = PC100
133MHZ = PC133

SDR = Single Data Rate

DDR SDRAM (late 90s - 2006)

DDR = Double Data Rate

organization is similar but **four banks** are used in each DDR SDRAM chip instead of two.

Also DDR2

(DDR3 increases the number of banks to 8 banks)

Data transfer on both **rising and falling edges of the clock**

DDR SDRAM rated by maximum or peak memory bandwidth

PC3200 = 8 bytes x 200 MHz x 2
= 3200 Mbytes/sec

Comparison of Synchronous Dynamic RAM SDRAM Generations:

DDR2 Vs. DDR and SDR SDRAM

For DDR3: The trend continues with another external frequency doubling

Single Data Rate (SDR) SDRAM transfers data on every rising edge of the clock.

Whereas both DDR and DDR2 are double pumped; they transfer data on the rising and falling edges of the clock.

DDR2 vs. DDR:

- **DDR2 doubles bus frequency for the same physical DRAM chip clock rate (as shown), thus doubling the effective data rate another time.**
- **Ability for much higher clock speeds than DDR, due to design improvements (still 4-banks per chip):**
  - **DDR2's bus frequency is boosted by electrical interface improvements, on-die termination, prefetch buffers and off-chip drivers.**
- **However, latency vs. DDR is greatly increased as a trade-off.**

Internal Base Frequency = 133 MHz

4258 MB/s = 8 x 133 x 4

DDR2 SDRAM

Shown: DDR2-533 (PC2-4200) ~ 4.2 GB/s peak bandwidth

2128 MB/s = 8 x 133 x 2

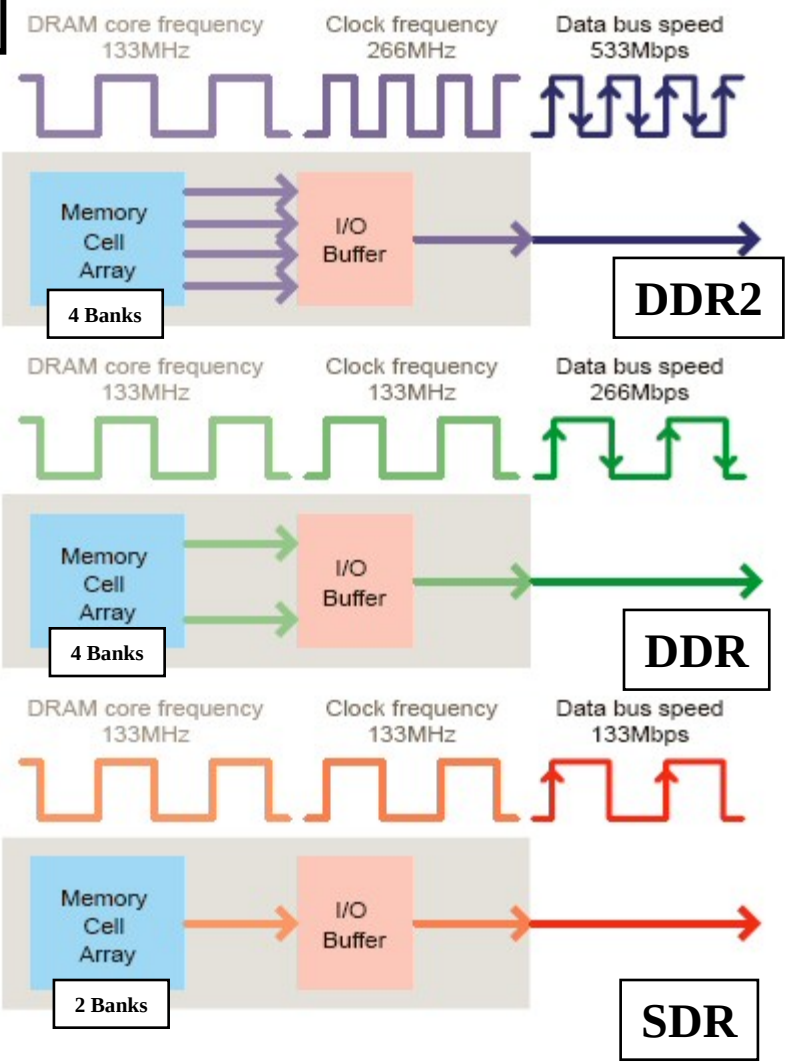
DDR SDRAM

Shown: DDR-266 (PC-2100) ~ 2.1 GB/s peak bandwidth

1064 MB/s = 8 x 133

SDR SDRAM

Shown: PC133 ~ 1.05 GB/s peak bandwidth



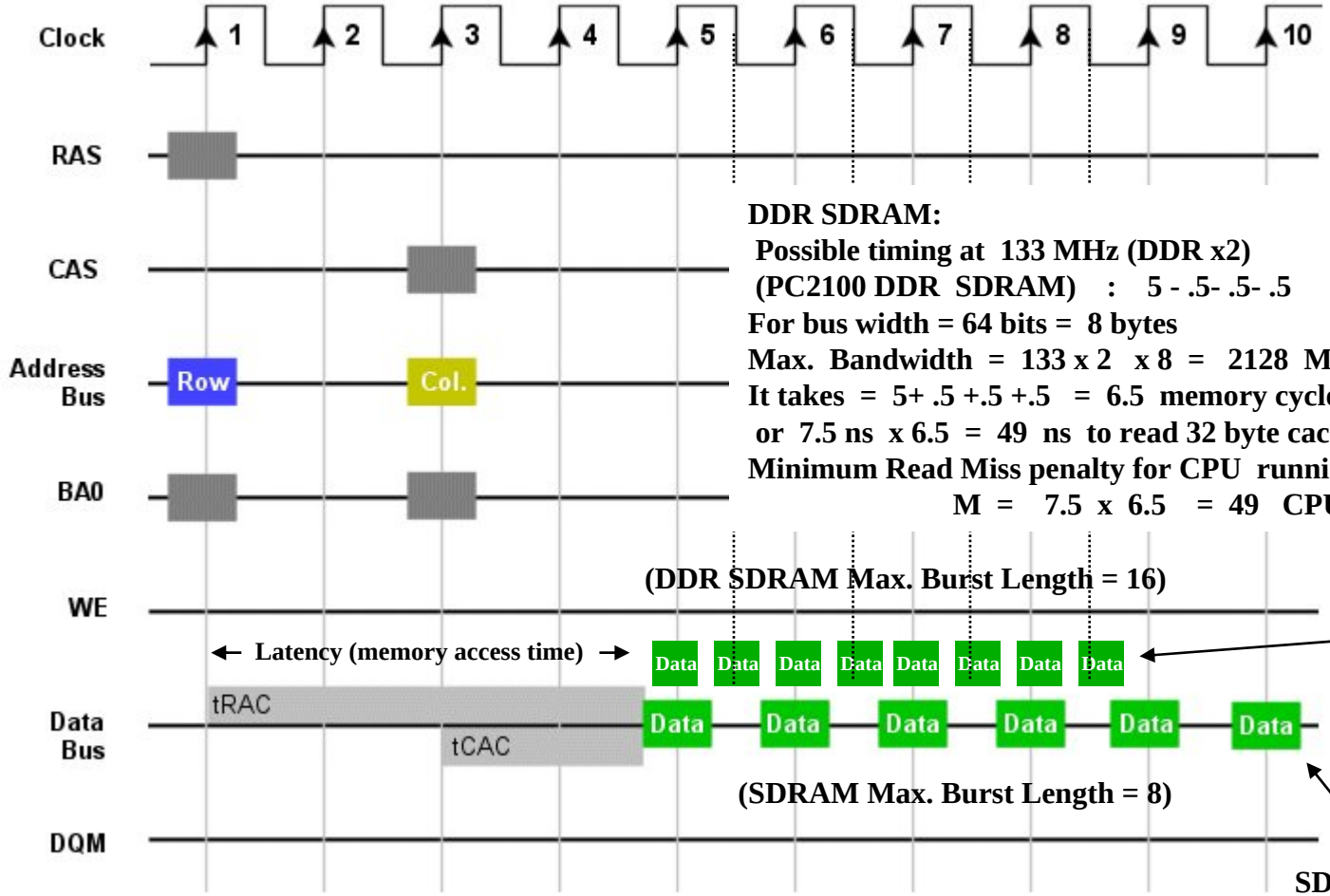
Peak bandwidth given for a single 64bit memory channel (i.e 8-byte memory bus width)

Figure Source: http://www.elpida.com/pdfs/E0678E10.pdf

Simplified SDR SDRAM/DDR SDRAM Read Timing

SDRAM Read

SDRAM clock cycle time ~ ½ tCAC



Twice as fast as SDR SDRAM?

DDR SDRAM:
Possible timing at 133 MHz (DDR x2)
(PC2100 DDR SDRAM) : 5 - .5- .5- .5
For bus width = 64 bits = 8 bytes
Max. Bandwidth = 133 x 2 x 8 = 2128 Mbytes/sec
It takes = 5+ .5 +.5 +.5 = 6.5 memory cycles
or 7.5 ns x 6.5 = 49 ns to read 32 byte cache block
Minimum Read Miss penalty for CPU running at 1 GHz =
M = 7.5 x 6.5 = 49 CPU cycles

(DDR SDRAM Max. Burst Length = 16)

DDR SDRAM (Late 90s-2006)

(SDRAM Max. Burst Length = 8)

SDR SDRAM (mid 90s)

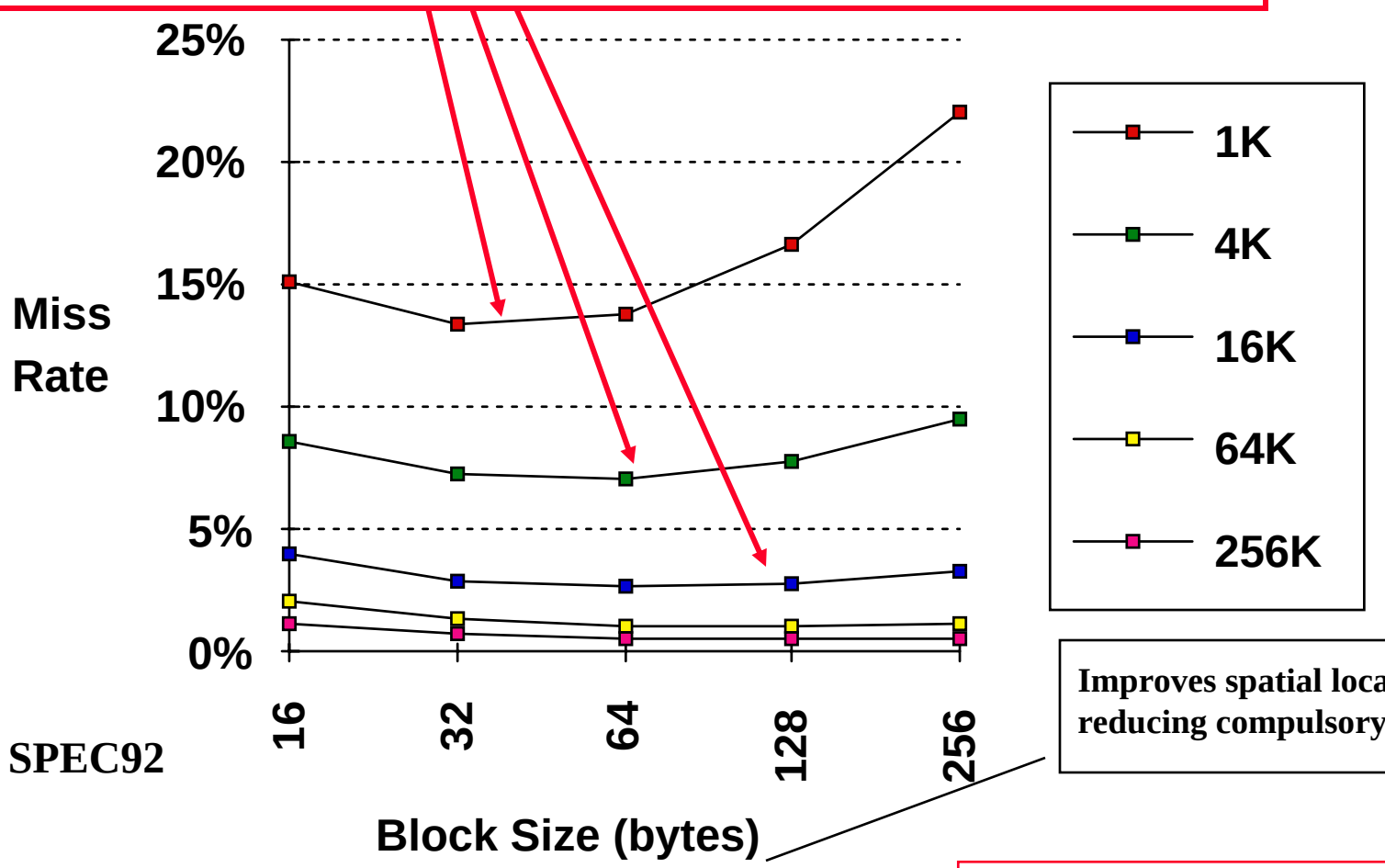
SDR SDRAM Typical timing at 133 MHz (PC133 SDRAM) : 5-1-1-1
For bus width = 64 bits = 8 bytes Max. Bandwidth = 133 x 8 = 1064 Mbytes/sec
It takes = 5+1+1+1 = 8 memory cycles or 7.5 ns x 8 = 60 ns to read 32 byte cache block
Minimum Read Miss penalty for CPU running at 1 GHz = M = 7.5 x 8 = 60 CPU cycles

In this example for SRD SDRAM: M = 60 cycles for DDR SDRAM: M = 49 cycles
Thus accounting for access latency DDR is 60/49 = 1.22 times faster
Not twice as fast (2128/1064 = 2) as indicated by peak bandwidth!

The Impact of Larger Cache Block Size on Miss Rate

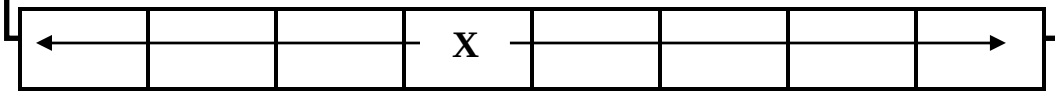
- A larger cache block size improves cache performance by taking better advantage of spatial locality However, for a fixed cache size, larger block sizes mean fewer cache block frames

- Performance keeps improving to a limit when the fewer number of cache block frames increases conflicts and thus overall cache miss rate



For SPEC92

Improves spatial locality reducing compulsory misses



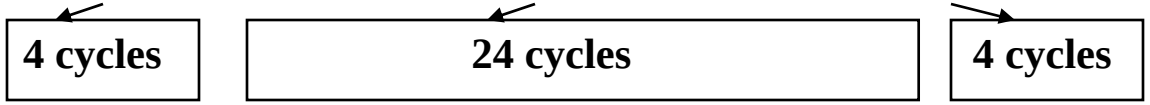
Memory Width, Interleaving: Performance Example

Given the following system parameters with single unified cache level L_1 (ignoring write policy):

Block size= 1 word Memory bus width= 1 word Miss rate =3% M = Miss penalty = 32 cycles

(4 cycles to send address 24 cycles access time, 4 cycles to send a word to CPU)

(Base system)



Miss Penalty = M= 4 + 24 + 4 = 32 cycles

(Base system)

Memory access/instruction = 1.2 $CPI_{execution}$ (ignoring cache misses) = 2

Miss rate (block size = 2 word = 8 bytes) = 2% Miss rate (block size = 4 words = 16 bytes) = 1%

- **The CPI of the base machine with 1-word blocks = 2 + (1.2 x 0.03 x 32) = 3.15**

(For Base system)

Increasing the block size to two words (64 bits) gives the following CPI: (miss rate = 2%)

- **32-bit bus and memory, no interleaving, M = 2 x 32 = 64 cycles CPI = 2 + (1.2 x .02 x 64) = 3.54**
- **32-bit bus and memory, interleaved, M = 4 + 24 + 8 = 36 cycles CPI = 2 + (1.2 x .02 x 36) = 2.86**
- **64-bit bus and memory, no interleaving, M = 32 cycles CPI = 2 + (1.2 x 0.02 x 32) = 2.77**

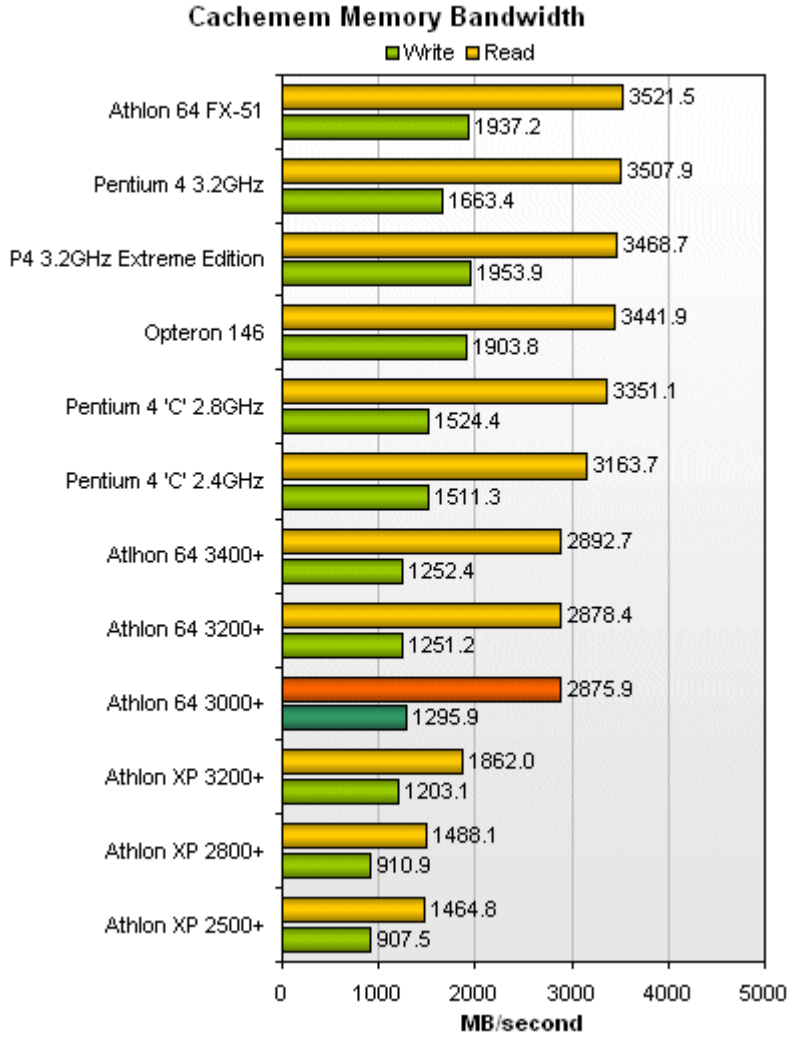
Increasing the block size to four words (128 bits); resulting CPI: (miss rate = 1%)

- **32-bit bus and memory, no interleaving , M = 4 x 32 = 128 cycles CPI = 2 + (1.2 x 0.01 x 128) = 3.54**
- **32-bit bus and memory, interleaved , M = 4 + 24 + 16 = 44 cycles CPI = 2 + (1.2 x 0.01 x 44) = 2.53**
- **64-bit bus and memory, no interleaving, M = 2 x 32 = 64 cycles CPI = 2 + (1.2 x 0.01 x 64) = 2.77**
- **64-bit bus and memory, interleaved, M = 4 + 24 + 8 = 36 cycles CPI = 2 + (1.2 x 0.01 x 36) = 2.43**
- **128-bit bus and memory, no interleaving, M = 32 cycles CPI = 2 + (1.2 x 0.01 x 32) = 2.38**

Miss Penalty = M = Number of CPU stall cycles for an access missed in cache and satisfied by main memory

X86 CPU Dual Channel PC3200 DDR SDRAM

Sample (Realistic?) Bandwidth Data



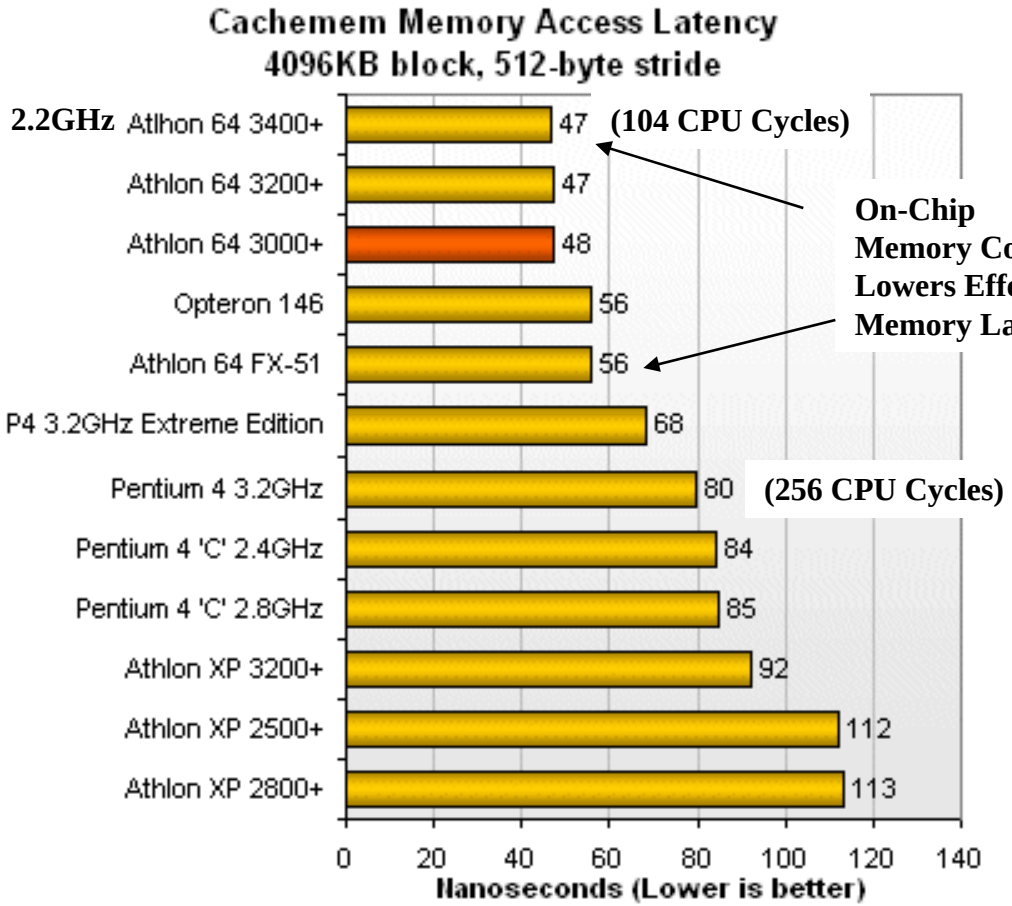
Dual (64-bit) Channel PC3200 DDR SDRAM has a theoretical peak bandwidth of

400 MHz x 8 bytes x 2 = 6400 MB/s

Is memory bandwidth still an issue?

Source: The Tech Report 1-21-2004
http://www.tech-report.com/reviews/2004q1/athlon64-3000/index.x?pg=3

X86 CPU Dual Channel PC3200 DDR SDRAM Sample (Realistic?) Latency Data



(104 CPU Cycles)

On-Chip Memory Controller Lowers Effective Memory Latency

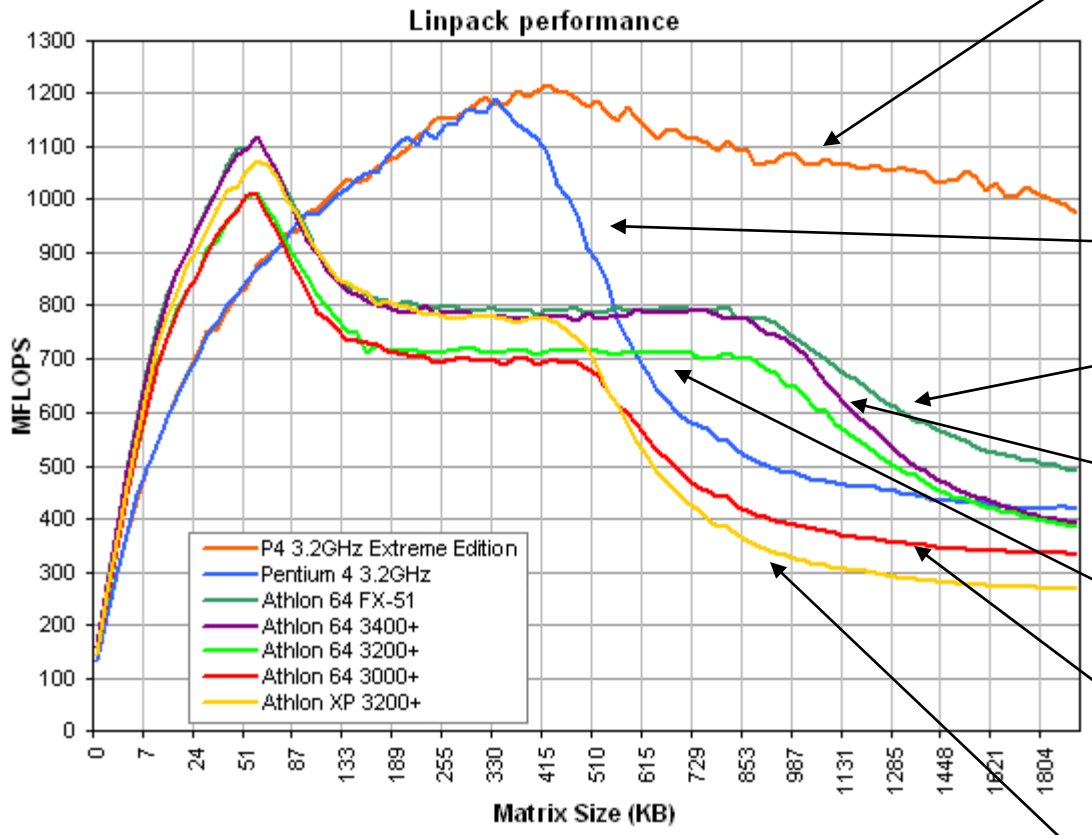
(256 CPU Cycles)

PC3200 DDR SDRAM has a theoretical latency range of 18-40 ns (not accounting for memory controller latency or other address/data line delays).

Is memory latency still an issue?

Source: The Tech Report (1-21-2004)
http://www.tech-report.com/reviews/2004q1/athlon64-3000/index.x?pg=3

X86 CPU Cache/Memory Performance Example:
AMD Athlon XP/64/FX Vs. Intel P4/Extreme Edition



Intel P4 3.2 GHz Extreme Edition
Data L1: 8KB
Data L2: 512 KB
Data L3: 2048 KB

Intel P4 3.2 GHz
Data L1: 8KB
Data L2: 512 KB

AMD Athon 64 FX51 2.2 GHz
Data L1: 64KB
Data L2: 1024 KB (exclusive)

AMD Athon 64 3400+ 2.2 GHz
Data L1: 64KB
Data L2: 1024 KB (exclusive)

AMD Athon 64 3200+ 2.0 GHz
Data L1: 64KB
Data L2: 1024 KB (exclusive)

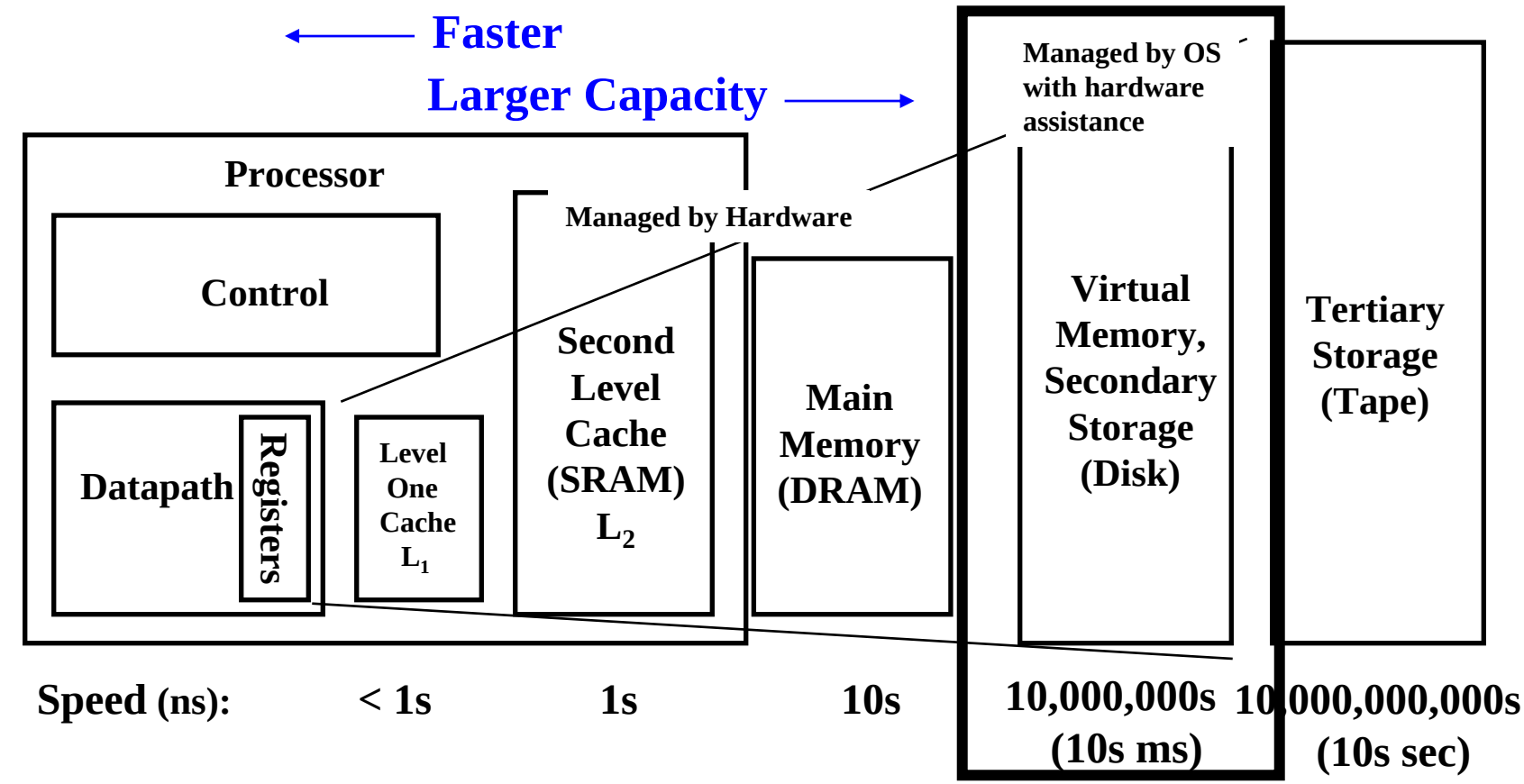
AMD Athon 64 3000+ 2.0 GHz
Data L1: 64KB
Data L2: 512 KB (exclusive)

AMD Athon XP 2.2 GHz
Data L1: 64KB
Data L2: 512 KB (exclusive)

Main Memory: Dual (64-bit) Channel PC3200 DDR SDRAM peak bandwidth of 6400 MB/s

Source: The Tech Report 1-21-2004
http://www.tech-report.com/reviews/2004q1/athlon64-3000/index.x?pg=3

A Typical Memory Hierarchy



Virtual Memory: 4th Edition in 5.4 (3rd Edition in 7.4)

Virtual Memory: Overview

4th Edition in 5.4
(3rd Edition in 7.4)

- Virtual memory controls two levels of the memory hierarchy:
  - Main memory (DRAM).
  - Mass storage (usually magnetic disks or SSDs).
- Main memory is divided into blocks allocated to different running processes in the system by the OS:
  - Fixed size blocks: Pages (size 4k to 64k bytes). (Most common)
  - Variable size blocks: Segments (largest size 2^{16} up to 2^{32}).
  - Paged segmentation: Large variable/fixed size segments divided into a number of fixed size pages (X86, PowerPC).

Superpages can be much larger

- At any given time, for any running process, a portion of its data/code is loaded (allocated) in main memory while the rest is available only in mass storage.
- A program code/data block needed for process execution and not present in main memory result in a page fault (address fault) and the page has to be loaded into main memory by the OS from disk (demand paging).
- A program can be run in any location in main memory or disk by using a relocation/mapping mechanism controlled by the operating system which maps (translates) the address from virtual address space (logical program address) to physical address space (main memory, disk).

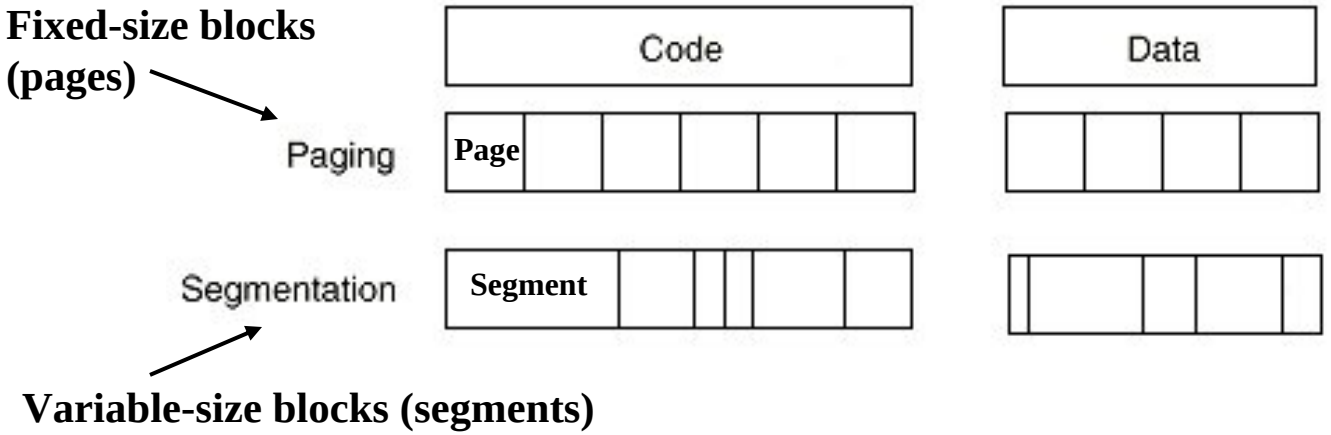
Using page tables

Virtual Memory: Motivation

- **Original Motivation:**
  - **Illusion of having more physical main memory (using demand paging)** e.g Full address space for each running process
  - **Allows program and data address relocation by automating the process of code and data movement between main memory and secondary storage.** Demand paging
- **Additional Current Motivation:**
  - **Fast process start-up.**
  - **Protection from illegal memory access.**
    - ***Needed for multi-tasking operating systems.***
  - **Controlled code and data sharing among processes.**
    - ***Needed for multi-threaded programs.***
  - **Uniform data access**
    - **Memory-mapped files**
    - **Memory-mapped network communication**

e.g local vs. remote memory access

Paging Versus Segmentation



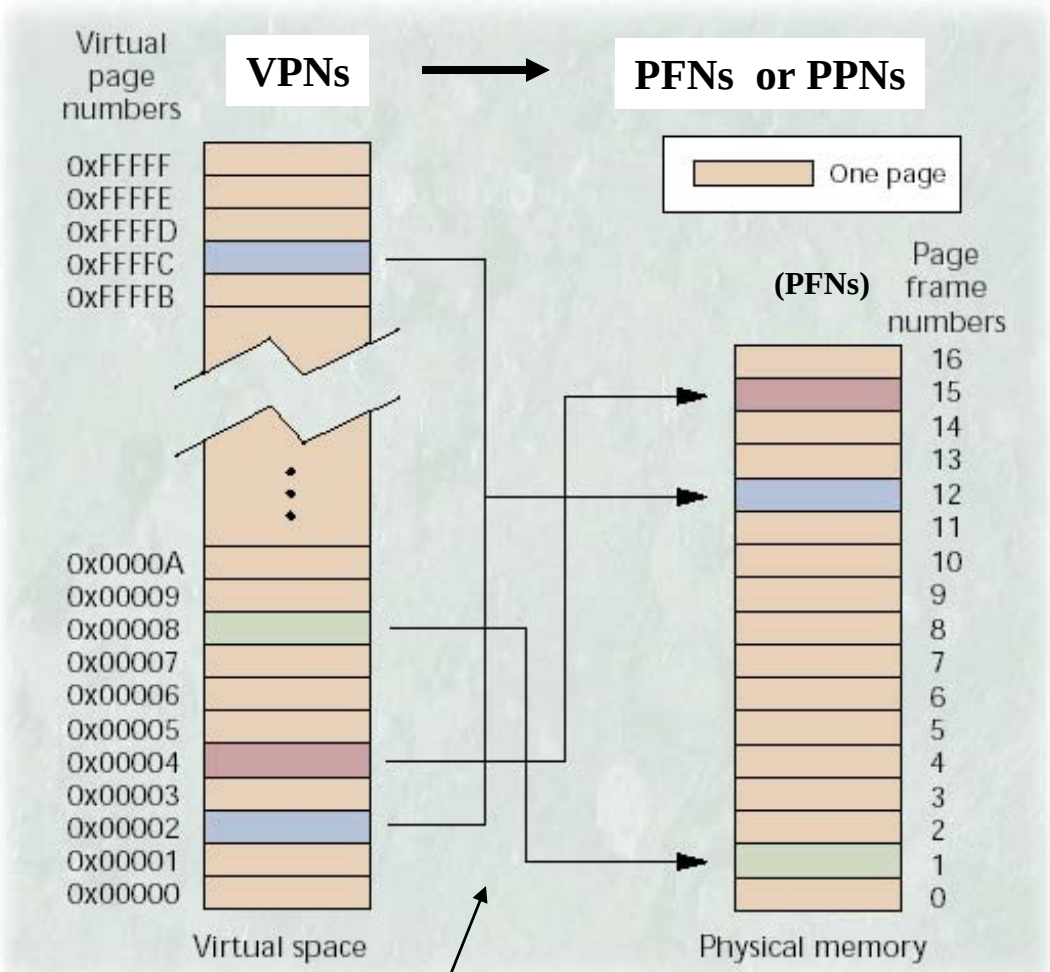
	Page	Segment
Words per address	One	Two (segment and offset)
Programmer visible?	Invisible to application programmer	May be visible to application programmer
Replacing a block	Trivial (all blocks are the same size)	Hard (must find contiguous, variable-size, unused portion of main memory)
Memory use inefficiency	Internal fragmentation (unused portion of page)	External fragmentation (unused pieces of main memory)
Efficient disk traffic	Yes (adjust page size to balance access time and transfer time)	Not always (small segments may transfer just a few bytes)

Virtual Address Space Vs. Physical Address Space

(logical)

Virtual memory stores only the most often used portions of a process address space in main memory and retrieves other portions from a disk as needed (demand paging).

The virtual-memory space is divided into pages identified by virtual page numbers (VPNs), shown on the far left, which are mapped to page frames or physical page numbers (PPNs) or page frame numbers (PFNs), in physical memory as shown on the right.



(or process logical address space)

Virtual address to physical address mapping or translation

Using a page table

Paging is assumed here

Virtual Address Space = Process Logical Address Space

Basic Virtual Memory Management

- Operating system makes decisions regarding which virtual (logical) pages of a process should be allocated in real physical memory and where (demand paging) assisted with hardware Memory Management Unit (MMU)
- On memory access -- If no valid virtual page to physical page translation (i.e page not allocated in main memory)
  - Page fault to operating system (e.g system call to handle page fault))

1 – Operating system requests page from disk

2 – Operating system chooses page for replacement
  - writes back to disk if modified

3 – Operating system allocates a page in physical memory and updates page table w/ new page table entry (PTE).

Then restart faulting process

Paging is assumed

Typical Parameter Range For Cache & Virtual Memory

Parameter	First-level cache	Virtual memory
Block (page) size	16–128 bytes	4096–65,536 bytes
Hit time	1–2 clock cycles	40–100 clock cycles
Miss penalty M	8–100 clock cycles	700,000–6,000,000 clock cycles
(Access time)	(6–60 clock cycles)	(500,000–4,000,000 clock cycles)
(Transfer time)	(2–40 clock cycles)	(200,000–2,000,000 clock cycles)
Miss rate	0.5–10%	0.00001– 0.001%
Data memory size	0.016–1MB	16–8192 MB

i.e page fault

Program assumed in steady state

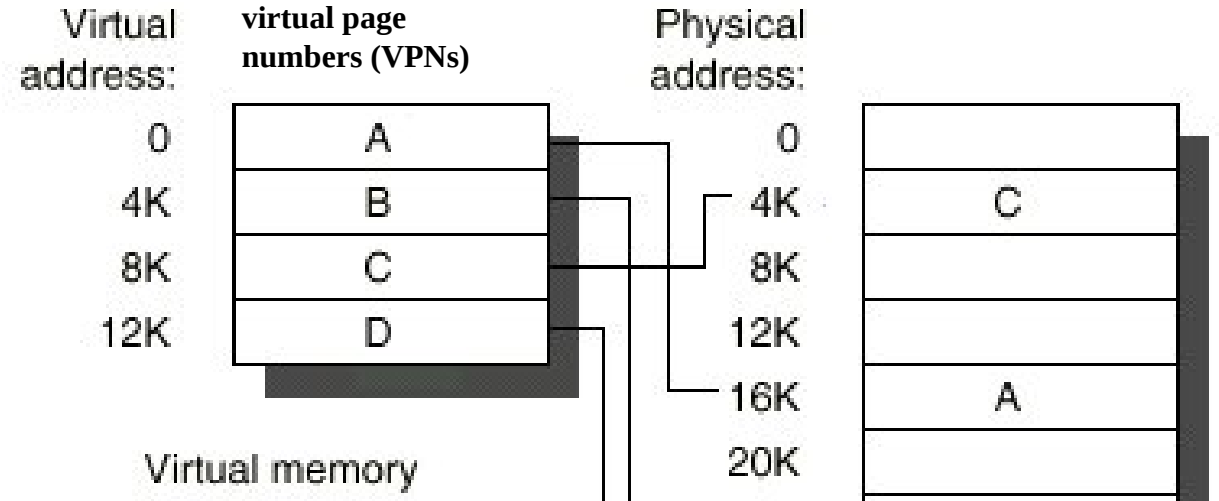
Paging is assumed here

Virtual Memory Basic Strategies

- **Main memory page placement(allocation): Fully associative placement or allocation (by OS) is used to lower the miss rate.**
- **Page replacement: The least recently used (LRU) page is replaced when a new page is brought into main memory from disk.**
- **Write strategy: Write back is used and only those pages changed in main memory are written to disk (dirty bit scheme is used).**
- **Page Identification and address translation: To locate pages in main memory a page table is utilized to translate from virtual page numbers (VPNs) to physical page numbers (PPNs) . The page table is indexed by the virtual page number and contains the physical address of the page.**
  - **In paging: Offset is concatenated to this physical page address.**
  - **In segmentation: Offset is added to the physical segment address.**
- **Utilizing address translation locality, a translation look-aside buffer (TLB) is usually used to cache recent address translations (PTEs) and prevent a second memory access to read the page table.**

PTE = Page Table Entry

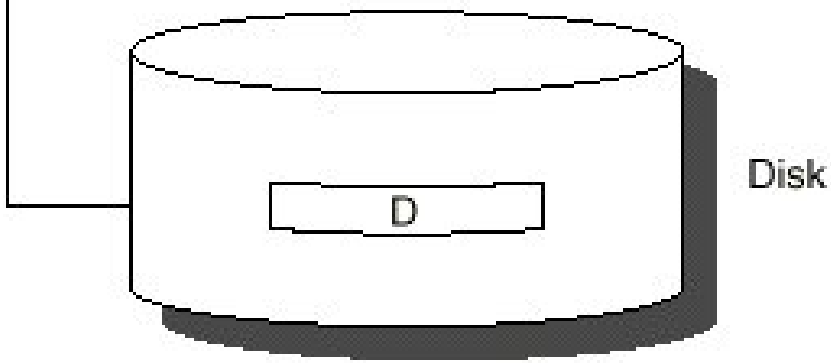
Virtual → Physical Address Translation



Contiguous virtual address (or logical) space of a program

Physical location of blocks A, B, C

Virtual address to physical address translation using page table



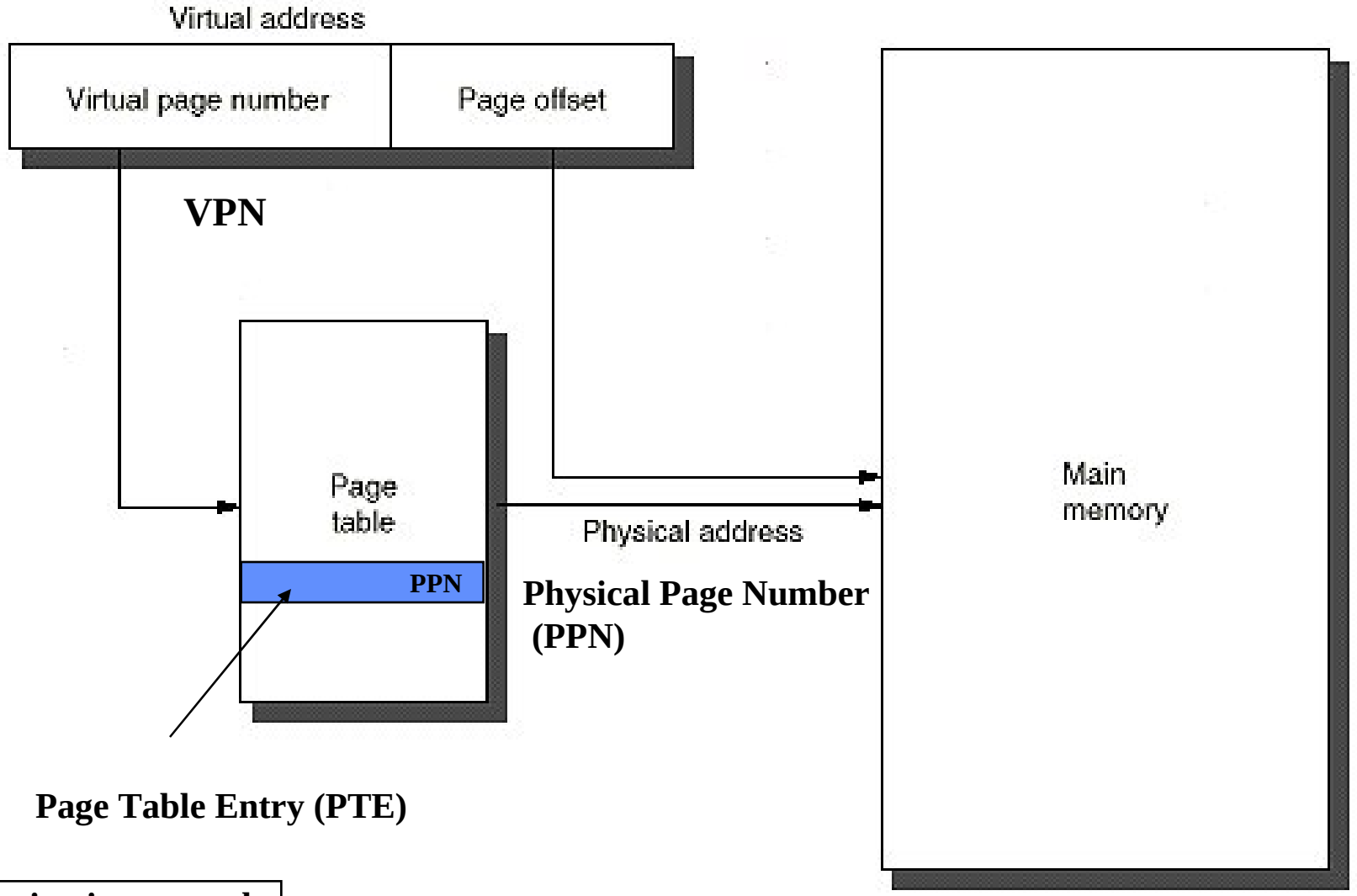
<u>Page Fault:</u> D in Disk (<u>not allocated in main memory</u>) OS allocates a page in physical main memory

Paging is assumed

Virtual to Physical Address Translation: Page Tables

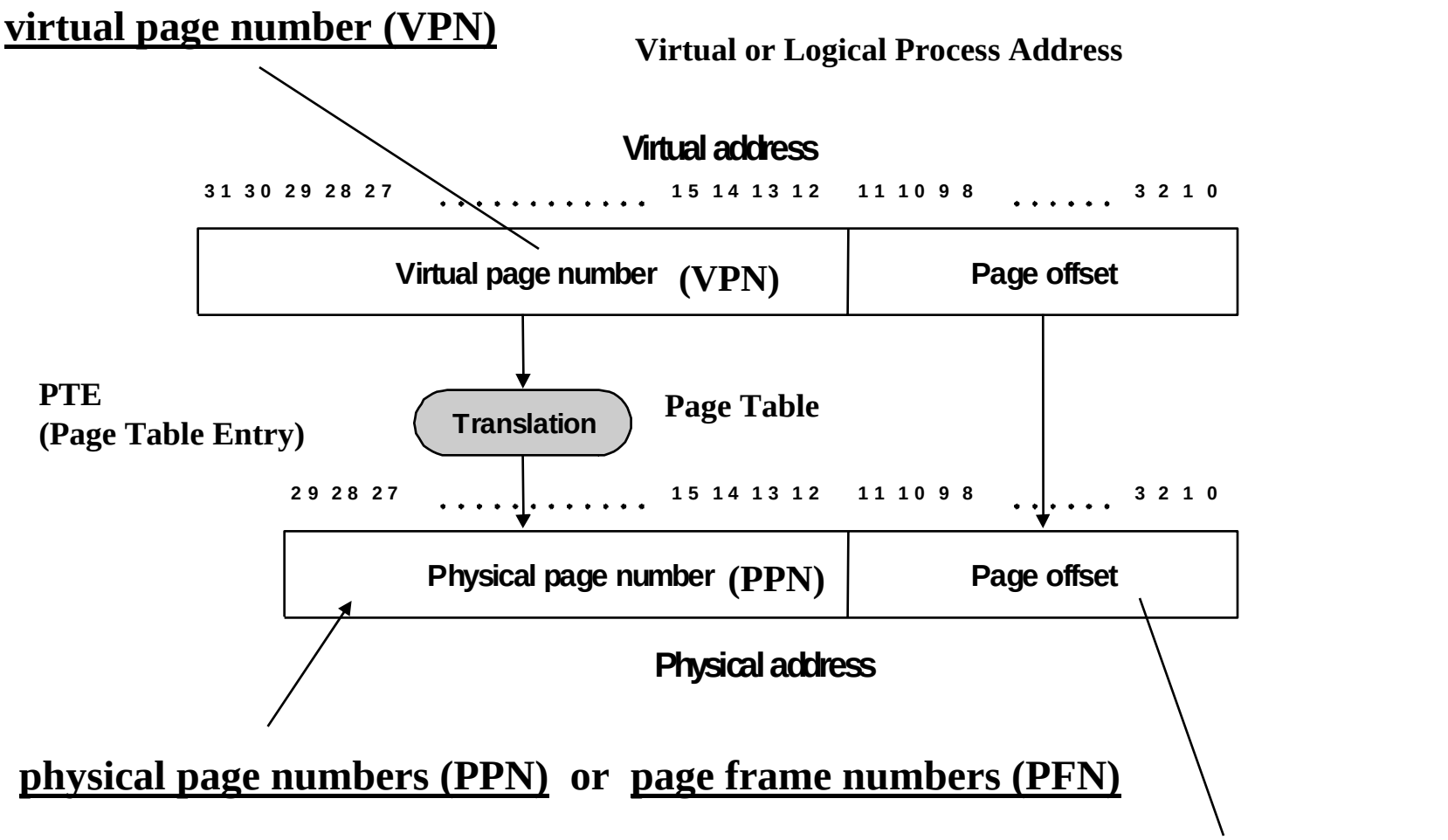
- **Mapping information from virtual page numbers (VPNs) to physical page numbers is organized into a page table which is a collection of page table entries (PTEs).**
- **At the minimum, a PTE indicates whether its virtual page is in memory, on disk, or unallocated and the PPN (or PFN) if the page is allocated.**
- **Over time, virtual memory evolved to handle additional functions including data sharing, address-space protection and page level protection, so a typical PTE now contains additional information including:**
  - **A *valid* bit, which indicates whether the PTE contains a valid translation;**
  - **The page's location in memory (page frame number, PFN) or location on disk (for example, an offset into a swap file);**
  - **The ID of the page's owner (the *address-space identifier (ASID)*, sometimes called Address Space Number (ASN) or *access key*;**
  - **The virtual page number (VPN);**
  - **A *reference* bit, which indicates whether the page was recently accessed;**
  - **A *modify* bit, which indicates whether the page was recently written; and**
  - ***Page-protection* bits, such as read-write, read only, kernel vs. user, and so on.**

Basic Mapping Virtual Addresses to Physical Addresses Using A Direct Page Table



Paging is assumed

Virtual to Physical Address Translation

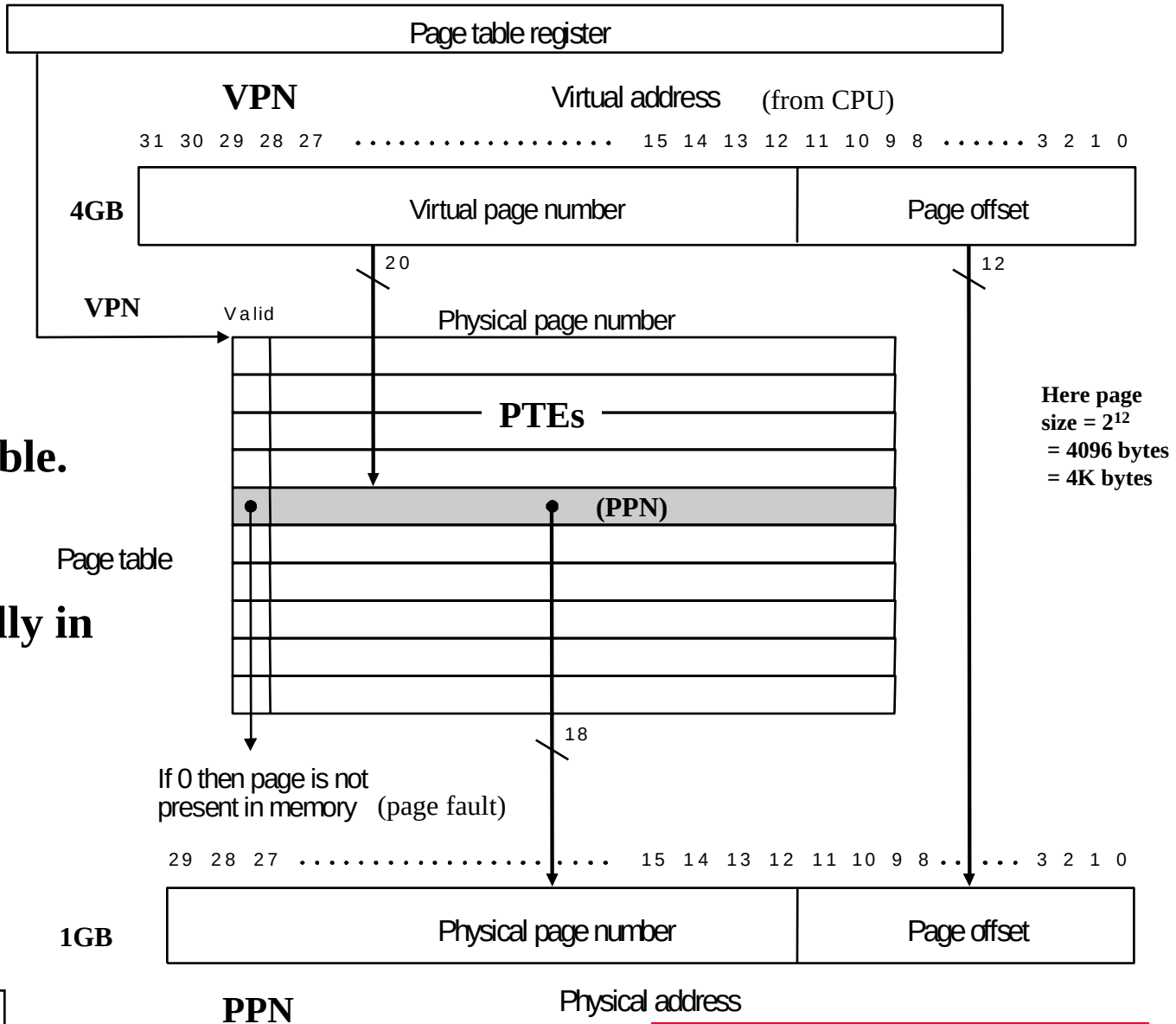


Here page size = 2^{12} = 4096 bytes = 4K bytes

Paging is assumed

Cache is normally designed to be physically addressed

Direct Page Table Organization



Two memory accesses needed:

- **First to page table.**
- **Second to item.**

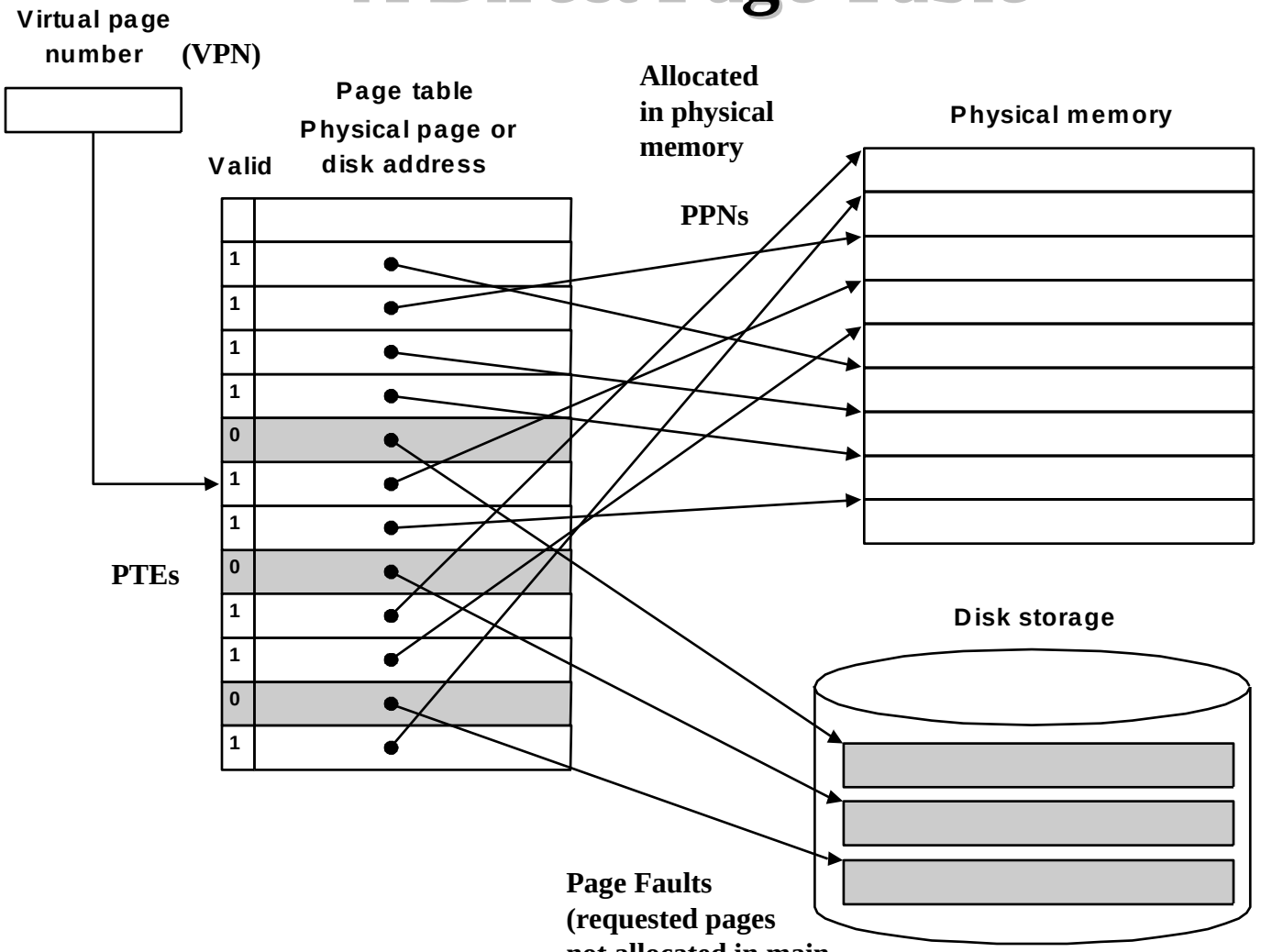
- **Page table usually in main memory.**

How to speedup virtual to physical address translation?

Paging is assumed

Cache is normally designed to be physically addressed

Virtual Address Translation Using A Direct Page Table



Paging is assumed

Speeding Up Address Translation:
Translation Lookaside Buffer (TLB)

- **Translation Lookaside Buffer (TLB) : Utilizing address reference locality, a small on-chip cache that contains recent address translations (PTEs).** i.e. recently used PTEs
  - TLB entries usually 32-128
  - High degree of associativity usually used
  - Separate instruction TLB (I-TLB) and data TLB (D-TLB) are usually used.
  - A unified larger second level TLB is often used to improve TLB performance and reduce the associativity of level 1 TLBs.

- **If a virtual address is found in TLB (a TLB hit), the page table in main memory is not accessed.**
- **TLB-Refill: If a virtual address is not found in TLB, a TLB miss (TLB fault) occurs and the system must search (walk) the page table for the appropriate entry and place it into the TLB this is accomplished by the TLB-refill mechanism .**

- **Types of TLB-refill mechanisms:**

Fast but not flexible

– **Hardware-managed TLB:** A **hardware finite state machine** is used to refill the TLB on a TLB miss by walking the page table. (PowerPC, IA-32)

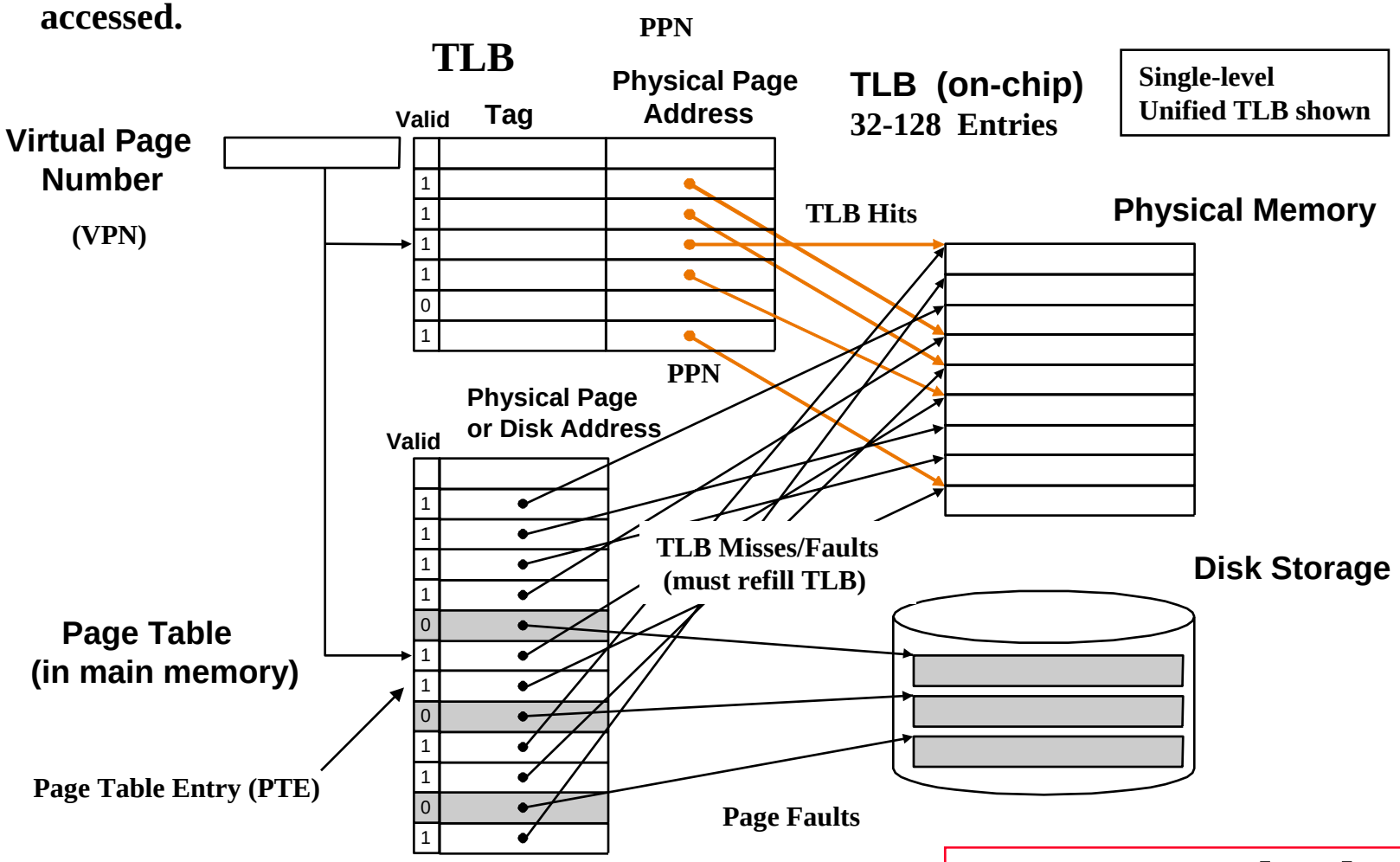
Flexible but slower

– **Software-managed TLB:** TLB refill handled by the **operating system**. (MIPS, Alpha, UltraSPARC, HP PA-RISC, …)

Speeding Up Address Translation:

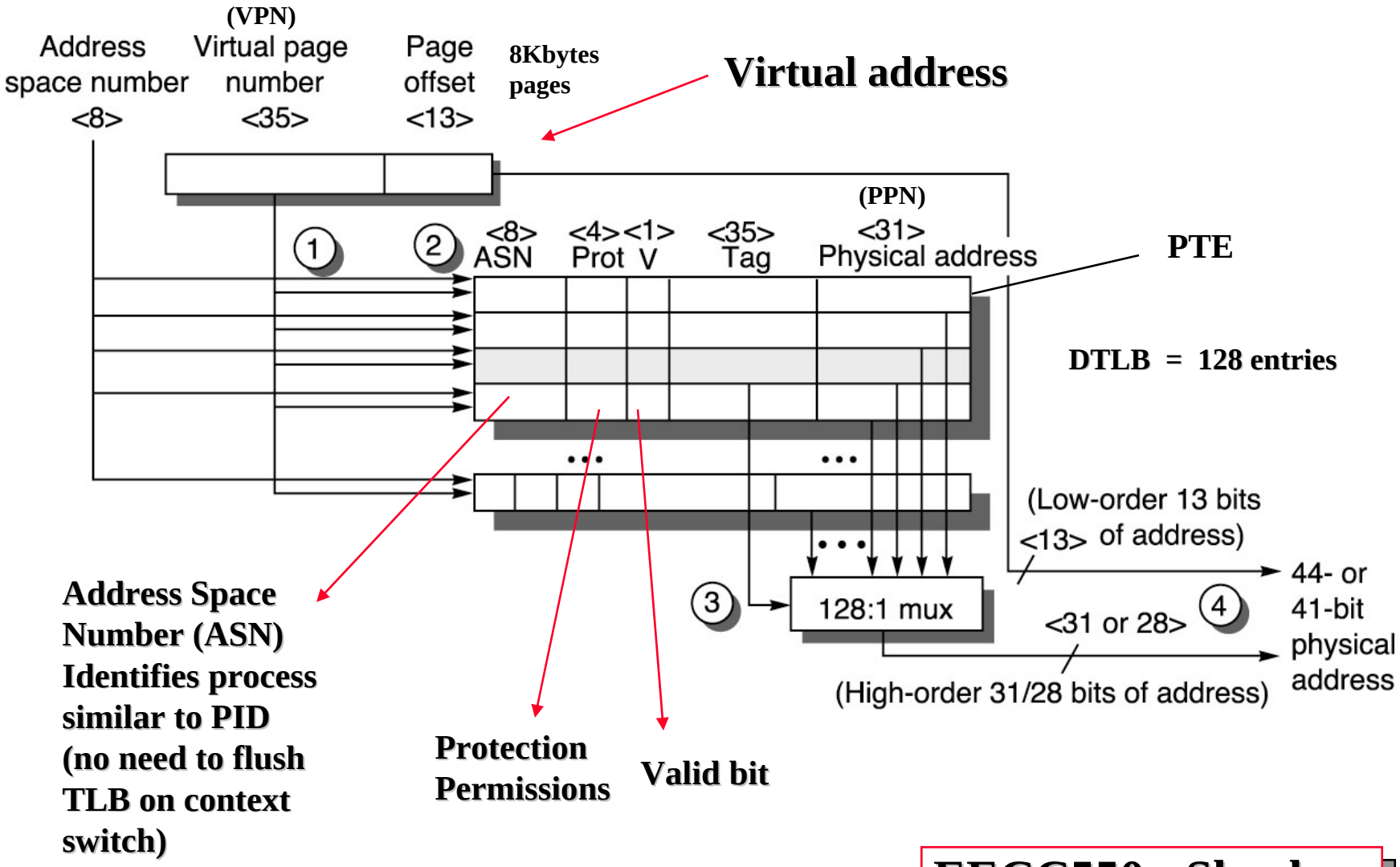
Translation Lookaside Buffer (TLB)

- **TLB: A small on-chip cache that contains recent address translations (PTEs).**
- **If a virtual address is found in TLB (a TLB hit), the page table in main memory is not accessed.**



Paging is assumed

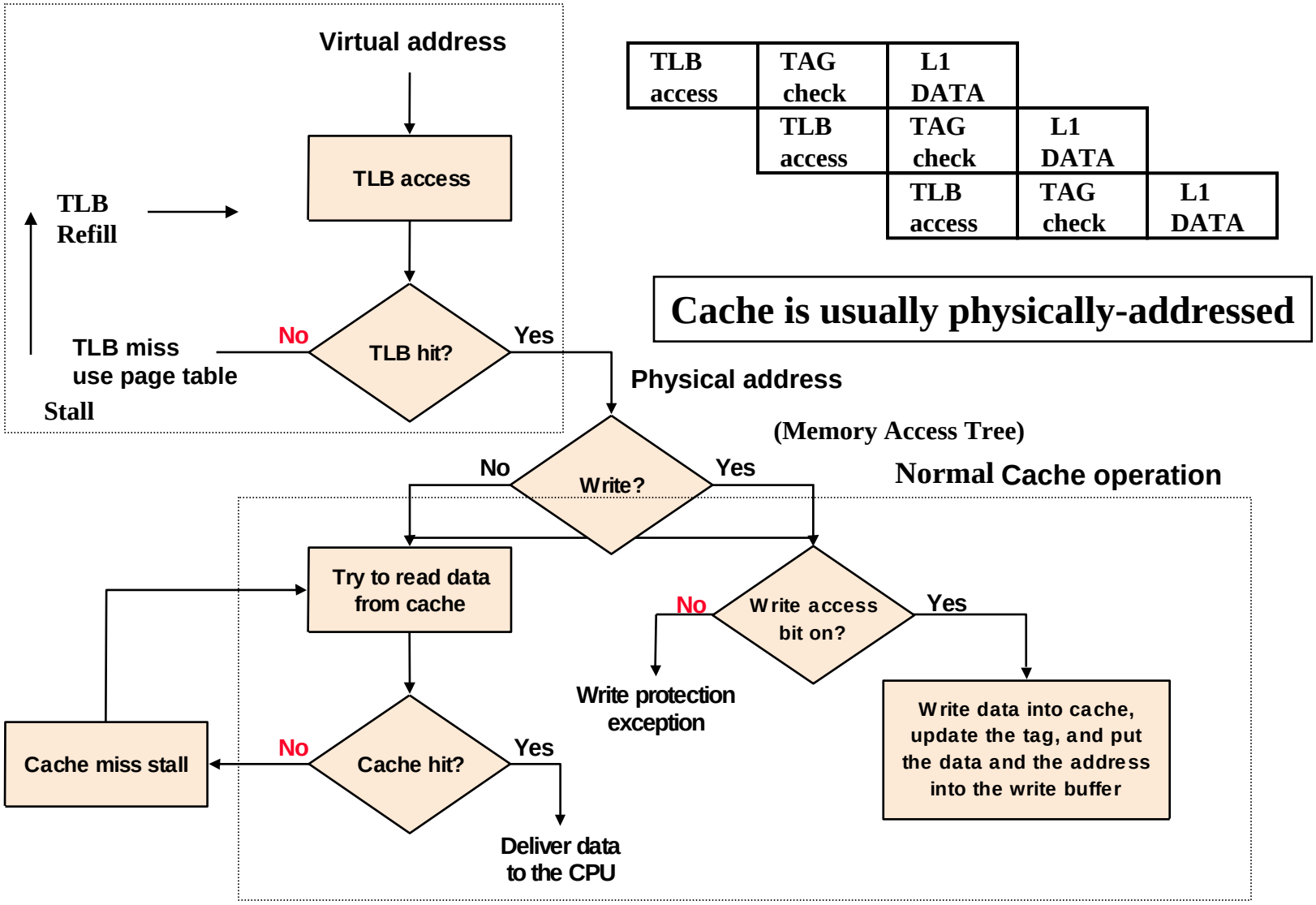
Operation of The Alpha 21264 Data TLB (DTLB) During Address Translation



PID = Process ID PTE = Page Table Entry

Basic TLB & Cache Operation

TLB Operation



TLB access	TAG check	L1 DATA		
	TLB access	TAG check	L1 DATA	
		TLB access	TAG check	L1 DATA

Cache is usually physically-addressed

CPU Performance with Real TLBs

When a real TLB is used with a TLB miss rate and a TLB miss penalty (time needed to refill the TLB) is used:

$$\text{CPI} = \text{CPI}_{\text{execution}} + \text{mem stalls per instruction} + \text{TLB stalls per instruction}$$

Where:

Mem Stalls per instruction = Mem accesses per instruction x mem stalls per access

Similarly:

1 + fraction of loads and stores

TLB Stalls per instruction = Mem accesses per instruction x TLB stalls per access

TLB stalls per access = TLB miss rate x TLB miss penalty

Example: **(For unified single-level TLB)**

Given: $\text{CPI}_{\text{execution}}$ = 1.3 Mem accesses per instruction = 1.4

Mem stalls per access = .5 TLB miss rate = .3% TLB miss penalty = 30 cycles

What is the resulting CPU CPI?

Mem Stalls per instruction = 1.4 x .5 = .7 cycles/instruction

TLB stalls per instruction = 1.4 x (TLB miss rate x TLB miss penalty)

= 1.4 x .003 x 30 = .126 cycles/instruction

CPI = 1. 3 + .7 + .126 = 2.126

EECC550 - Shaaban

$\text{CPI}_{\text{execution}}$ = Base CPI with ideal memory

Event Combinations of Cache, TLB, Virtual Memory

Cache	TLB	Virtual Memory	Possible? When?
Hit	Hit	Hit	TLB/Cache Hit
Miss	Hit	Hit	Possible, no need to check page table
Hit	Miss	Hit	TLB miss, found in page table
Miss	Miss	Hit	TLB miss, cache miss
Miss	Miss	Miss	Page fault
Miss	Hit	Miss	Impossible, cannot be in TLB if not in main memory
Hit	Hit	Miss	Impossible, cannot be in TLB or cache if not in main memory
Hit	Miss	Miss	Impossible, cannot be in cache if not in memory