

2023年度(令和5年)版


Ver. 2022-11-07a

Course number: CSC.T363



# コンピュータアーキテクチャ Computer Architecture

## 10. 仮想記憶 Virtual Memory



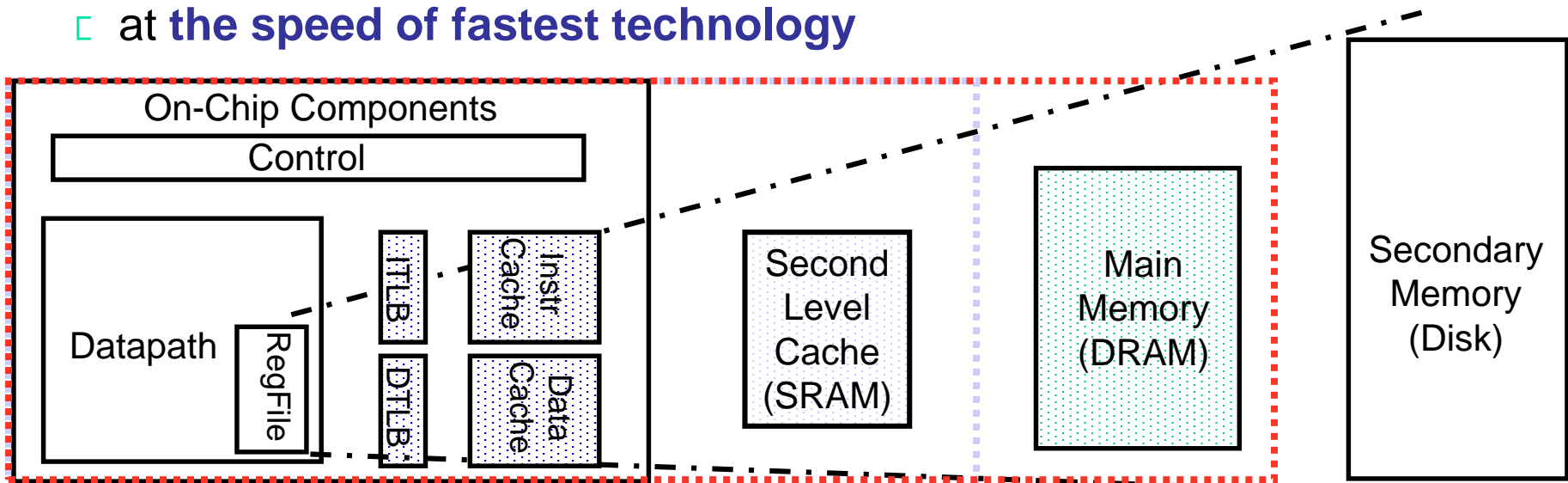
[www.arch.cs.titech.ac.jp/lecture/CA/](http://www.arch.cs.titech.ac.jp/lecture/CA/)  
Tue 13:30-15:10, 15:25-17:05  
Fri 13:30-15:10



吉瀬 謙二 情報工学系  
Kenji Kise, Department of Computer Science  
kise\_at\_c.titech.ac.jp

# A Typical Memory Hierarchy

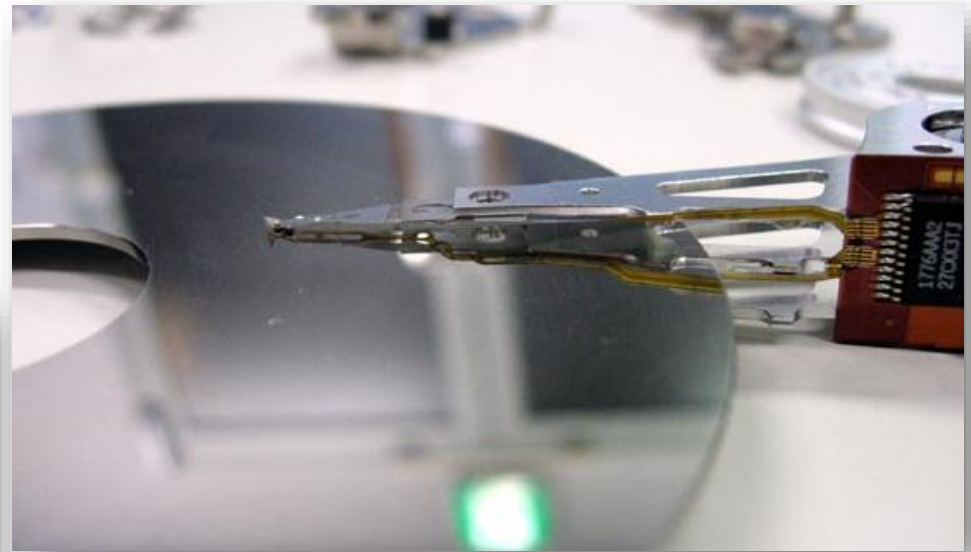
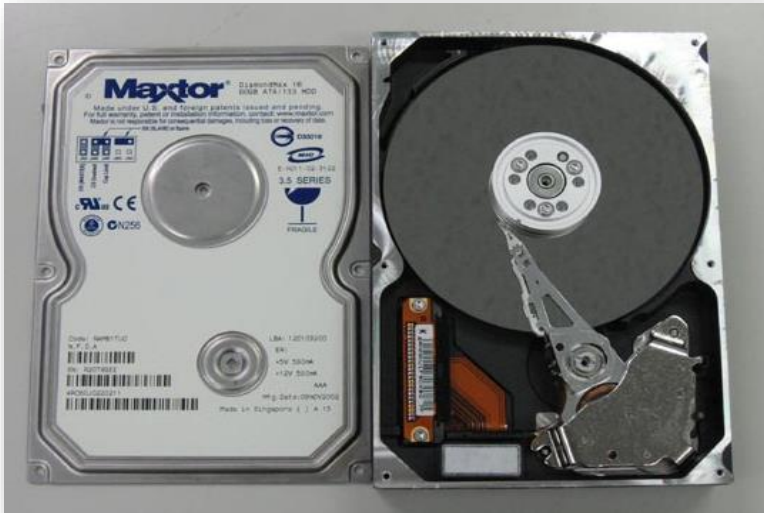
- By taking advantage of **the principle of locality**
  - Present **much memory** in the **cheapest technology**
  - at **the speed of fastest technology**



<b>Speed (%cycles):</b>	½'s	1's	10's	100's	1,000's
<b>Size (bytes):</b>	100's	K's	10K's	M's	G's to T's
<b>Cost:</b>	highest				lowest

TLB: Translation Lookaside Buffer

# Magnetic Disk (磁気ディスク)



<http://sougo057.aicomp.jp/0001.html>

# Magnetic Disk (磁気ディスク)

- Purpose

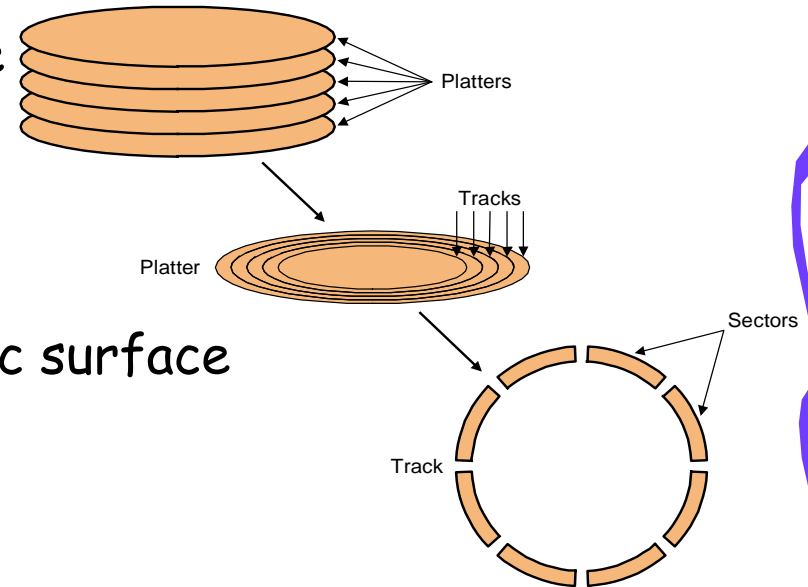
- Long term, **nonvolatile** (不揮発性) storage
- Lowest level in the memory hierarchy
  - slow, large, inexpensive

- General structure

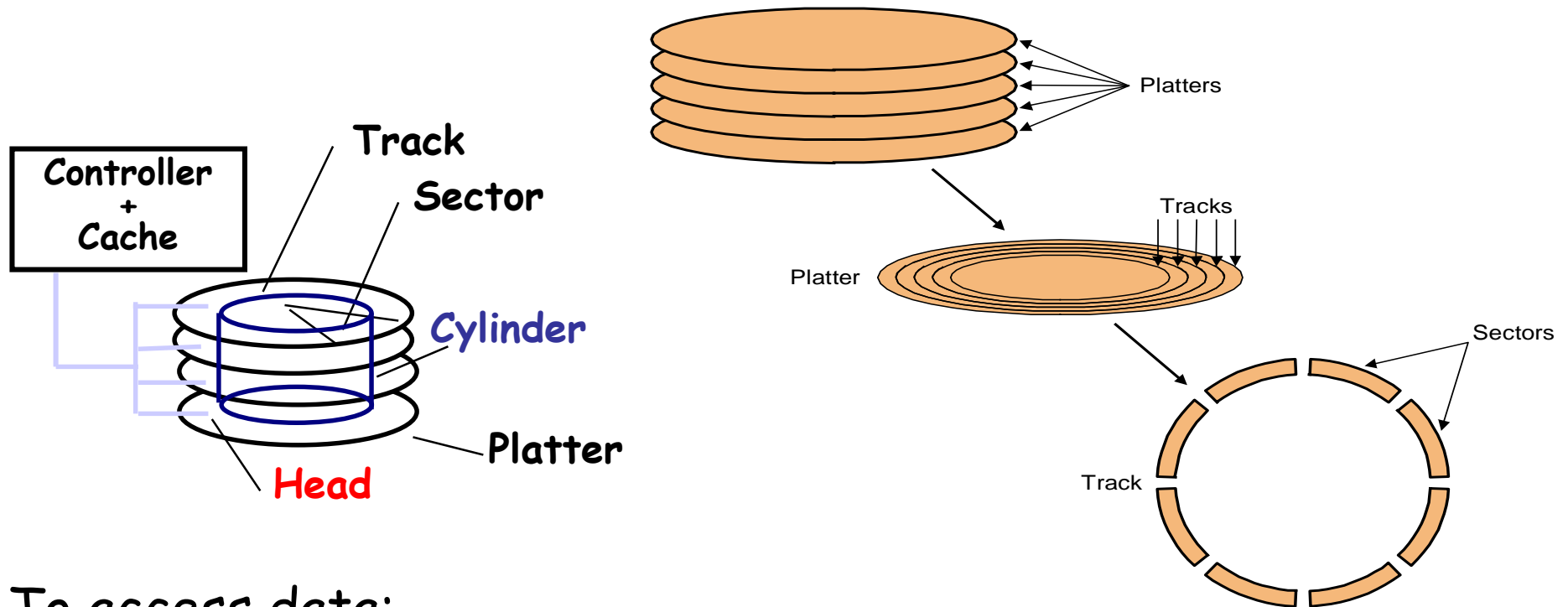
- A rotating **platter** coated with a magnetic surface
- A moveable read/write **head** to access the information on the disk

- Typical numbers

- 1 to 4 platters per disk of 1" to 5.25" in diameter (3.5" dominate in 2004)
- Rotational speeds of 5,400 to 15,000 RPM (rotation per minute)
- 10,000 to 50,000 **tracks** per surface
  - **cylinder** - all the tracks under the head at a given point on all surfaces
- 100 to 500 **sectors** per track
  - the smallest unit that can be read/written (typically **512B**)



# Disk Drives



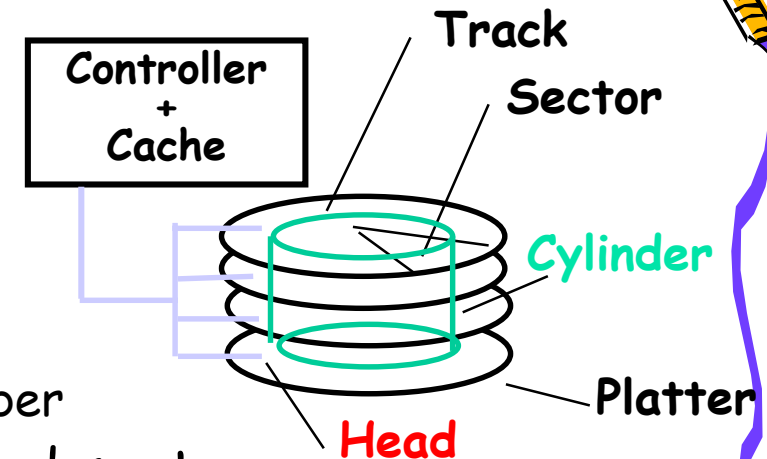
To access data:

- **seek time** (シーク時間): position the head over the proper track
- **rotational latency** (回転待ち時間): wait for desired sector
- **transfer time** (転送時間): grab the data (one or more sectors)
- **Controller time** (制御時間): the overhead the disk controller imposes in performing a disk I/O access

# Magnetic Disk Characteristic

## Disk read/write components

- Seek time:** position the head over the proper track (3 to 14 ms avg)
  - due to locality of disk references the actual average seek time may be only 25% to 33% of the advertised number
- Rotational latency:** wait for the desired sector to rotate under the head ( $\frac{1}{2}$  of  $1/\text{RPM}$  converted to ms)
  - $0.5/5400\text{RPM} = 0.5/90$  rotations per second = **5.6 ms**
  - $0.5/15000\text{RPM} = 0.5/250$  rotations per second = **2.0 ms**
- Transfer time:** transfer a block of bits (one or more sectors) under the head to the disk controller's cache (30 to 80 MB/s are typical disk transfer rates)
- Controller time:** the overhead the disk controller imposes in performing a disk I/O access (typically  $< .2$  ms)



# Disk Latency & Bandwidth Milestones

- Disk **latency** is one average seek time plus the rotational latency.
- Disk **bandwidth** is the peak transfer time of formatted data from the media (not from the cache).

	CDC Wren	SG ST41	SG ST15	SG ST39	SG ST37
Speed (RPM)	3600	5400	7200	10000	15000
<b>Year</b>	<b>1983</b>	<b>1990</b>	<b>1994</b>	<b>1998</b>	<b>2003</b>
Capacity (Gbytes)	0.03	1.4	4.3	9.1	73.4
Diameter (inches)	5.25	5.25	3.5	3.0	2.5
Interface	ST-412	SCSI	SCSI	SCSI	SCSI
<b>Bandwidth</b> (MB/s)	0.6	4	9	24	86
<b>Latency</b> (msec)	48.3	17.1	12.7	8.8	5.7

Patterson, CACM Vol 47, #10, 2004



# Example of 32-bit memory space (4GB)

0x00000000

00000000 00000000 00000000 00000000<sub>2</sub> = 0<sub>10</sub>



2GB Memory !

```
kterm
top - 11:35:26 up 10 days, 19:49, 2 users, load average: 0.01, 0.01, 0
Tasks: 164 total, 1 running, 163 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si,
Mem: 4002924k total, 3252404k used, 750520k free, 181808k buffers
Swap: 6062072k total, 0k used, 6062072k free, 2570804k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
    1 root        15   0 10348  896  584  S   0.0   0.0   0:01.00  init
    2 root         RT  -5     0     0     0  S   0.0   0.0   0:00.00  migration/0
    3 root         34  19     0     0     0  S   0.0   0.0   0:00.00  ksoftirqd/0
    4 root         RT  -5     0     0     0  S   0.0   0.0   0:00.00  watchdog/0
    5 root         RT  -5     0     0     0  S   0.0   0.0   0:00.00  migration/1
    6 root         34  19     0     0     0  S   0.0   0.0   0:00.00  ksoftirqd/1
    7 root         RT  -5     0     0     0  S   0.0   0.0   0:00.00  watchdog/1
    8 root         RT  -5     0     0     0  S   0.0   0.0   0:00.01  migration/2
    9 root         34  19     0     0     0  S   0.0   0.0   0:00.00  ksoftirqd/2
   10 root         RT  -5     0     0     0  S   0.0   0.0   0:00.00  watchdog/2
```

0xFFFFFFFF

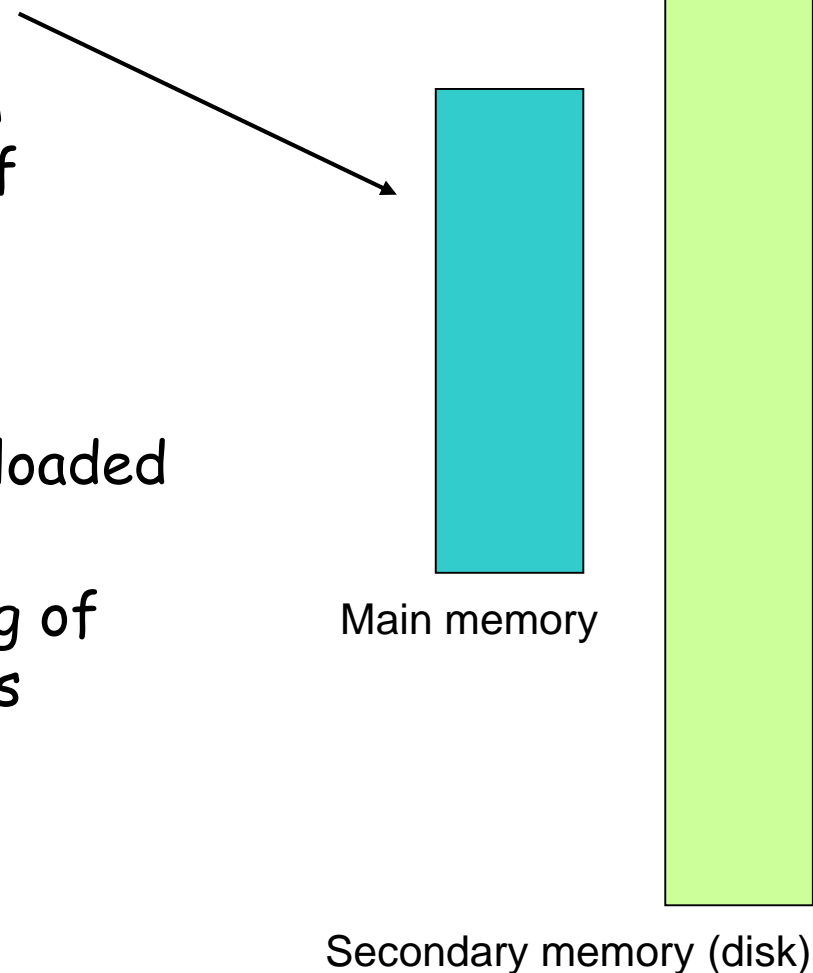
11111111 11111111 11111111 11111111<sub>2</sub> = 4,294,967,296 - 1<sub>10</sub>





# Virtual Memory (仮想記憶)

- Use main memory as a “cache” for secondary memory
  - Provides the ability to easily run programs larger than the size of physical memory
  - Simplifies loading a program for execution by providing for code relocation (i.e., the code can be loaded anywhere in main memory)
  - Allows efficient and safe sharing of memory among multiple programs
- **Security, memory protection**
  - control memory access rights



# Virtual Memory

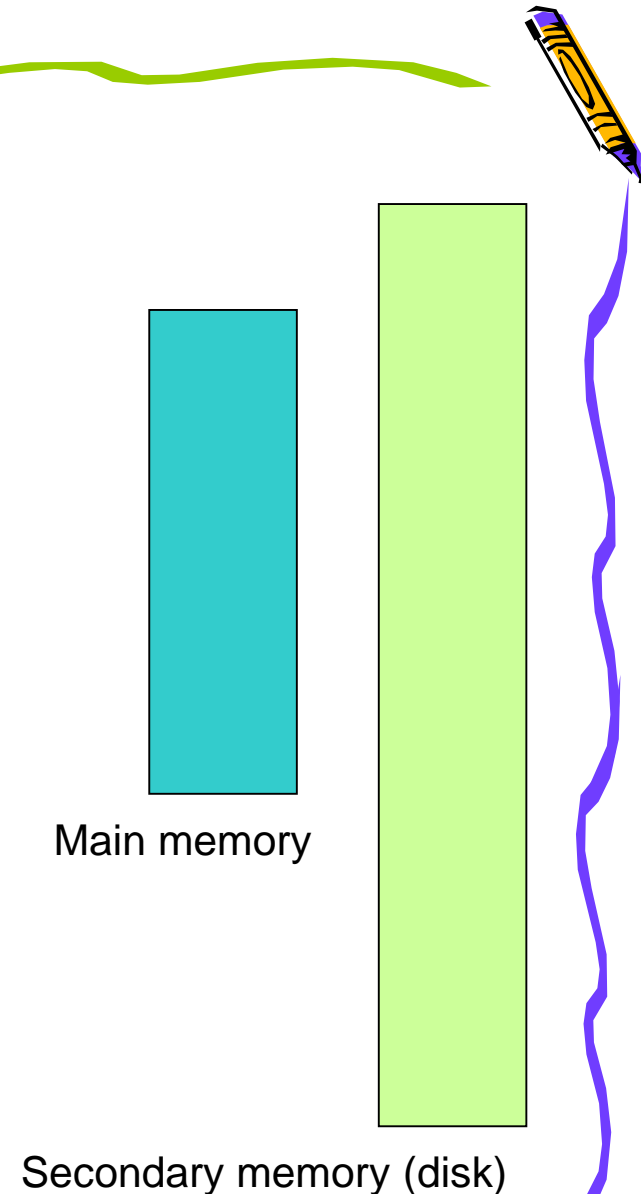


- What makes it work? – again the Principle of Locality
  - A program is likely to access a relatively small portion of its address space during any period of time



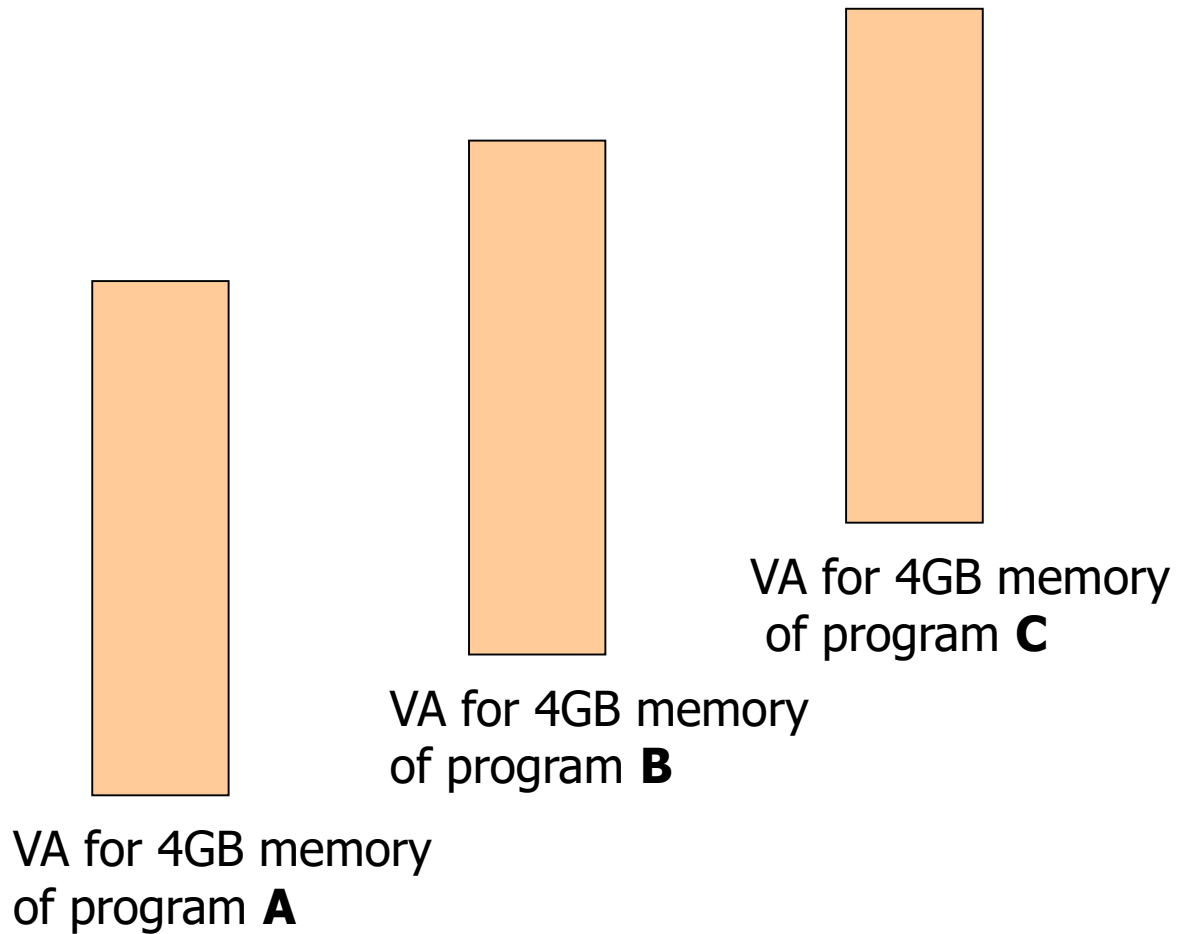
# Virtual Memory

- Each program is compiled into its own address space – a “**virtual address (VA)**” space
- **Physical address (PA)** for the access of physical devices
  - During run-time each **virtual address, VA** must be translated to a **physical address, PA**

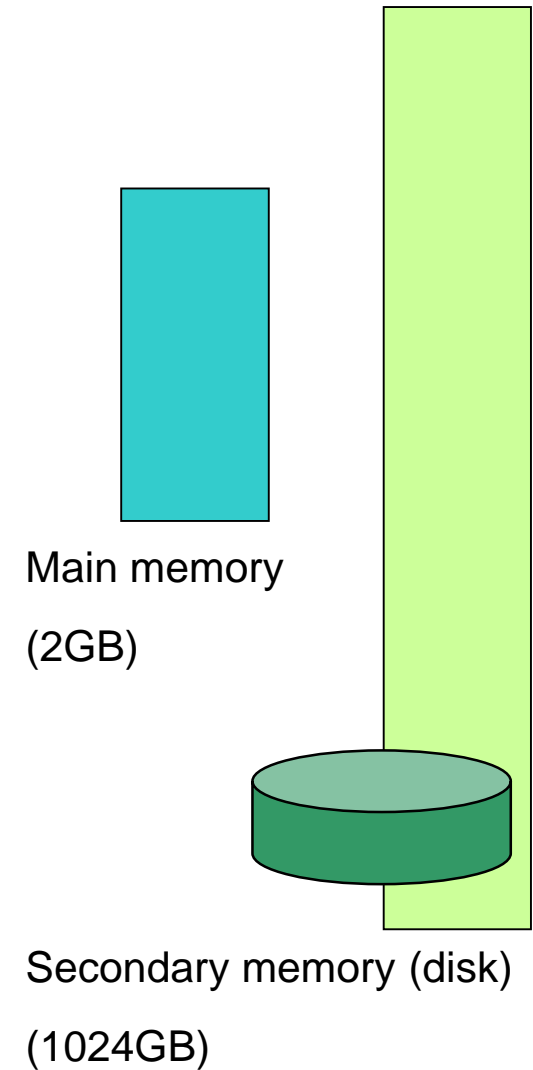


# Virtual Memory

## Virtual address world



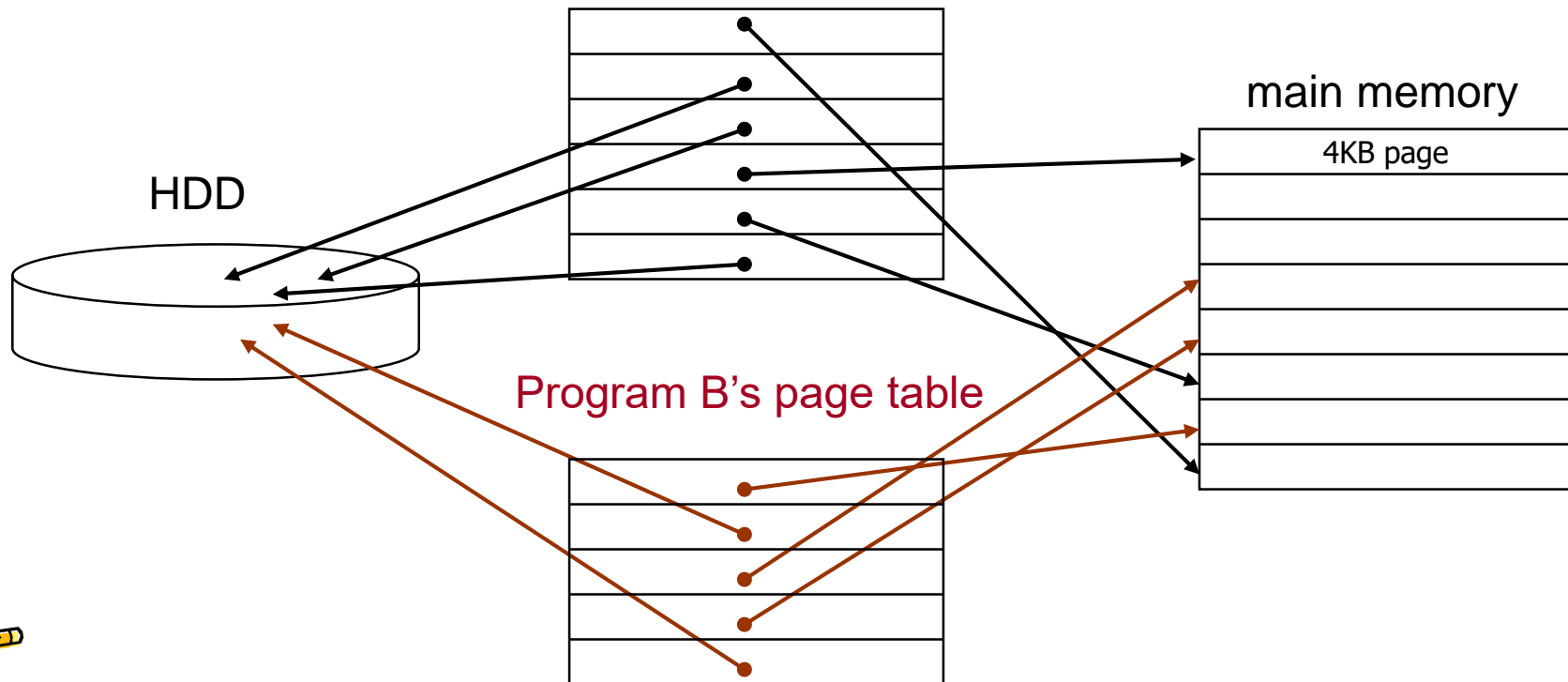
## Physical address world



# Two Programs Sharing Physical Memory

- A program's address space is divided into **pages** (all one fixed size, typical 4KB) or **segments** (variable sizes)
  - The starting location of each page (either in **main memory** or in **secondary memory**) is contained in the program's **page table**

Program A's **page table** (virtual address space)

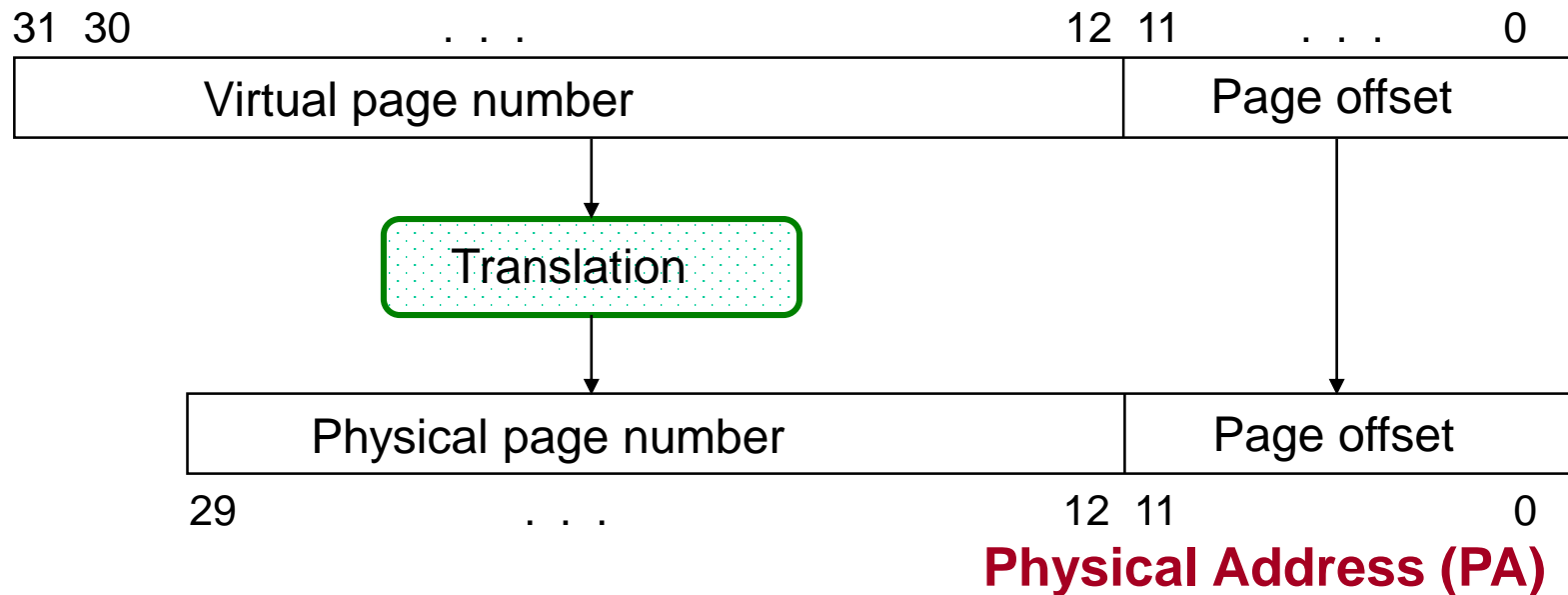


# Address Translation

- A virtual address is translated to a physical address by a combination of hardware and software

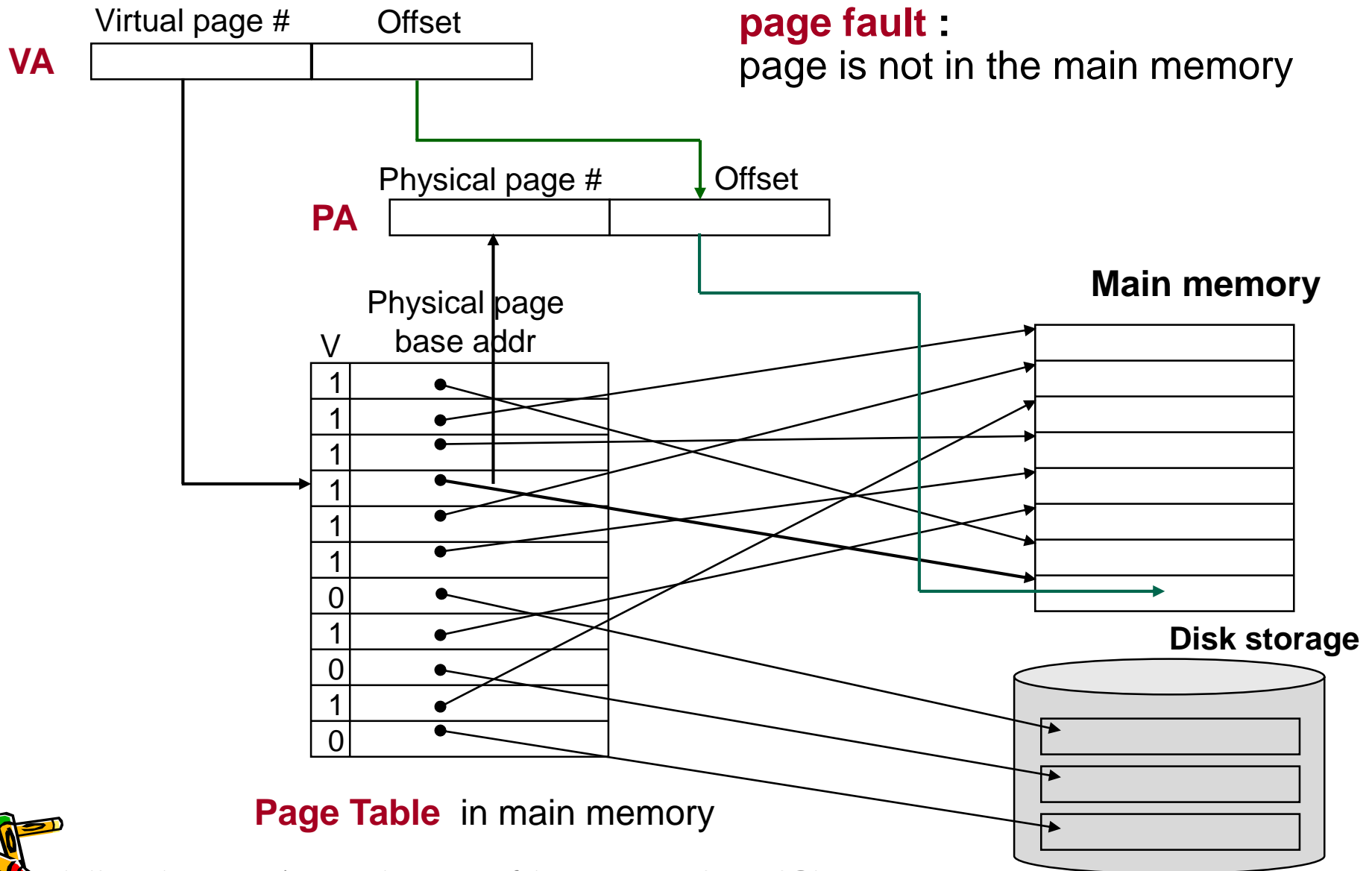
## Virtual Address (VA)

Assume 4KB page size



- So each memory request **first** requires an **address translation** from the virtual space to the physical space

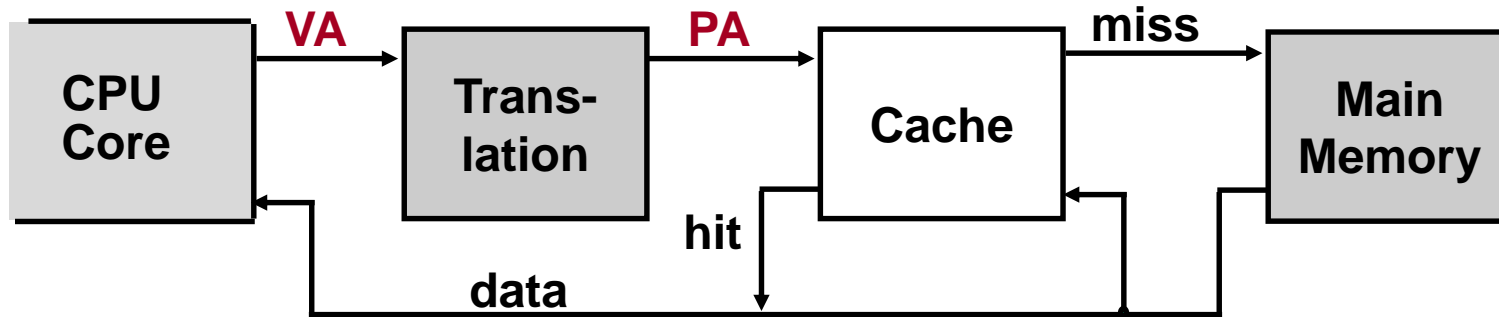
# Address Translation Mechanisms



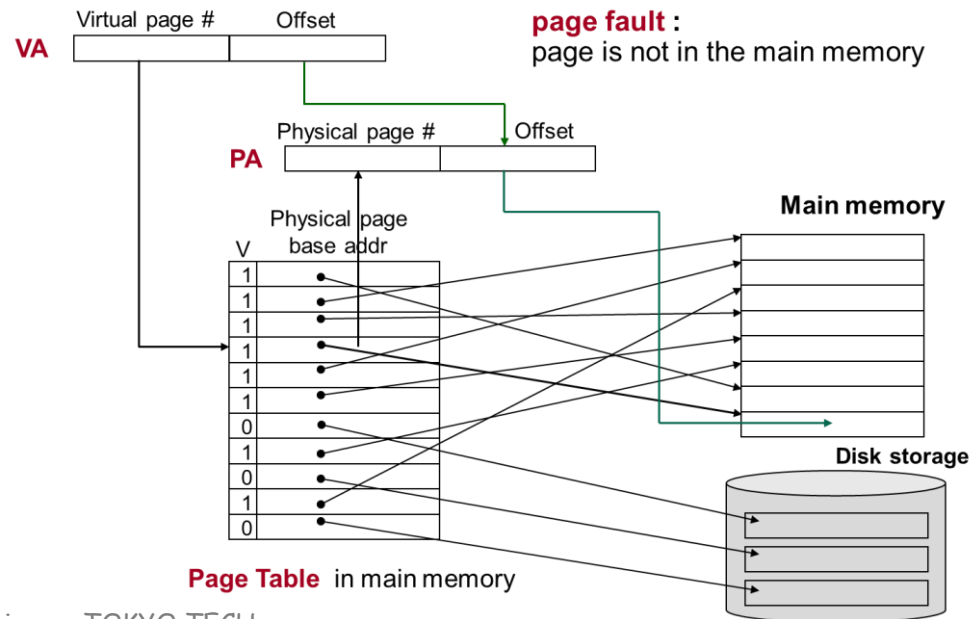


# Virtual Addressing, the hardware fix


- Thus it may take an **extra memory access** to translate a virtual address to a physical address



- This makes memory (cache) accesses **very expensive** (if every access was really **two** accesses)
- What's the solution ?**



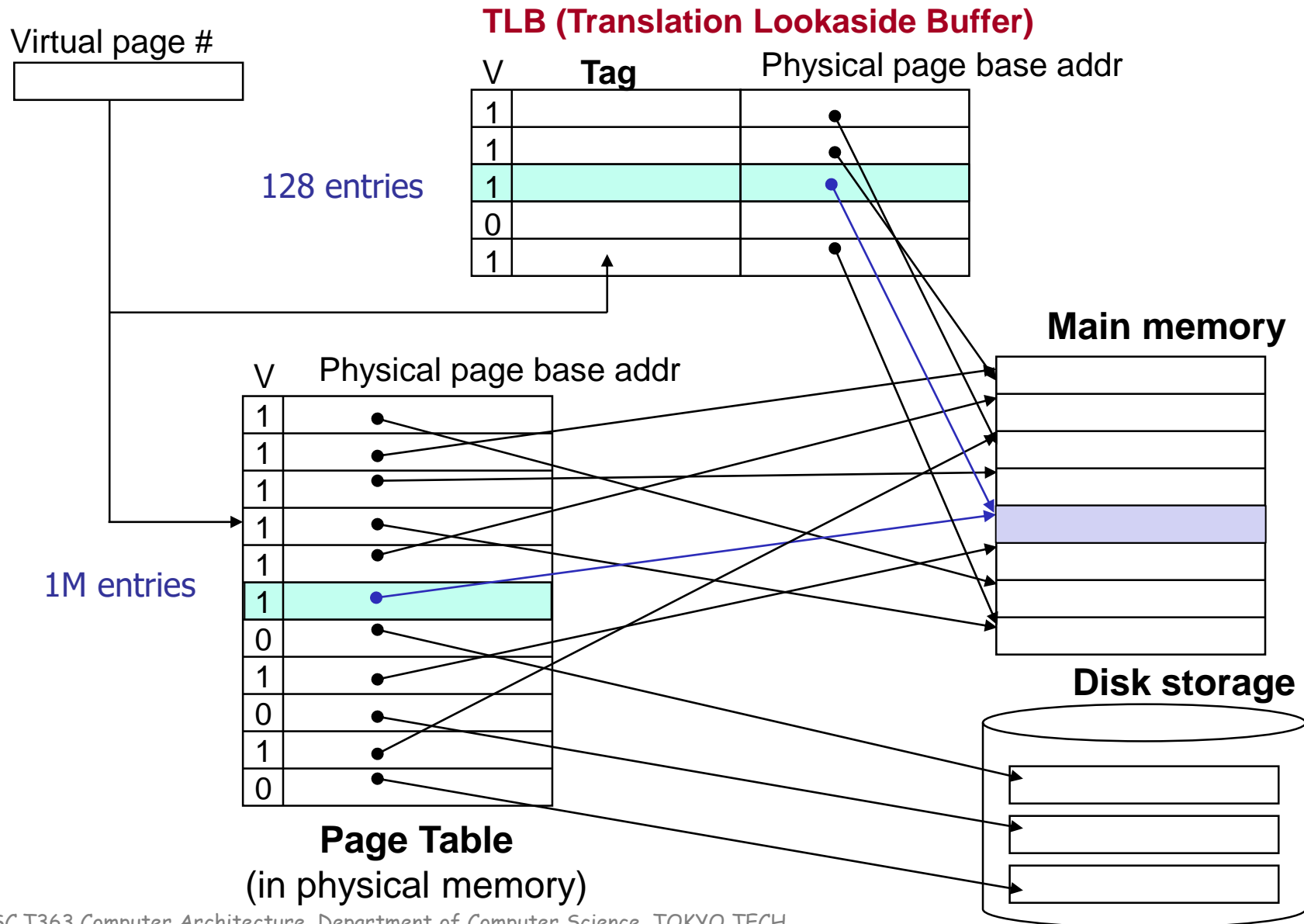
# Virtual Addressing, the hardware fix



- The hardware fix is to use a **Translation Lookaside Buffer (TLB)** (アドレス変換バッファ)
  - a small **cache** that keeps track of recently used address mappings to avoid having to do a **page table** lookup

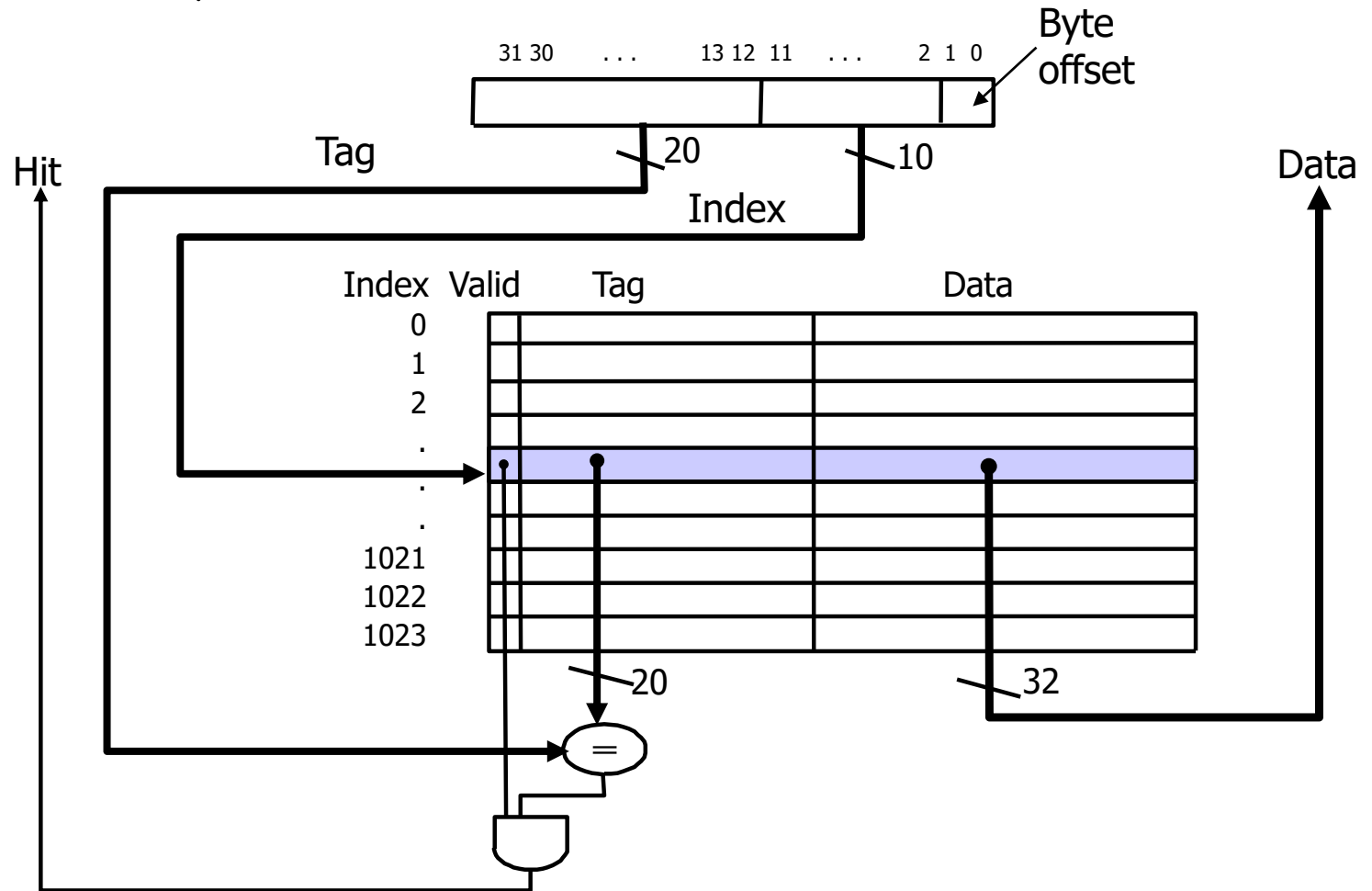


# Making Address Translation Fast



# MIPS Direct Mapped Cache Example

- One word/block, cache size = 1K words



*What kind of locality are we taking advantage of?*

# Translation Lookaside Buffers (TLBs)

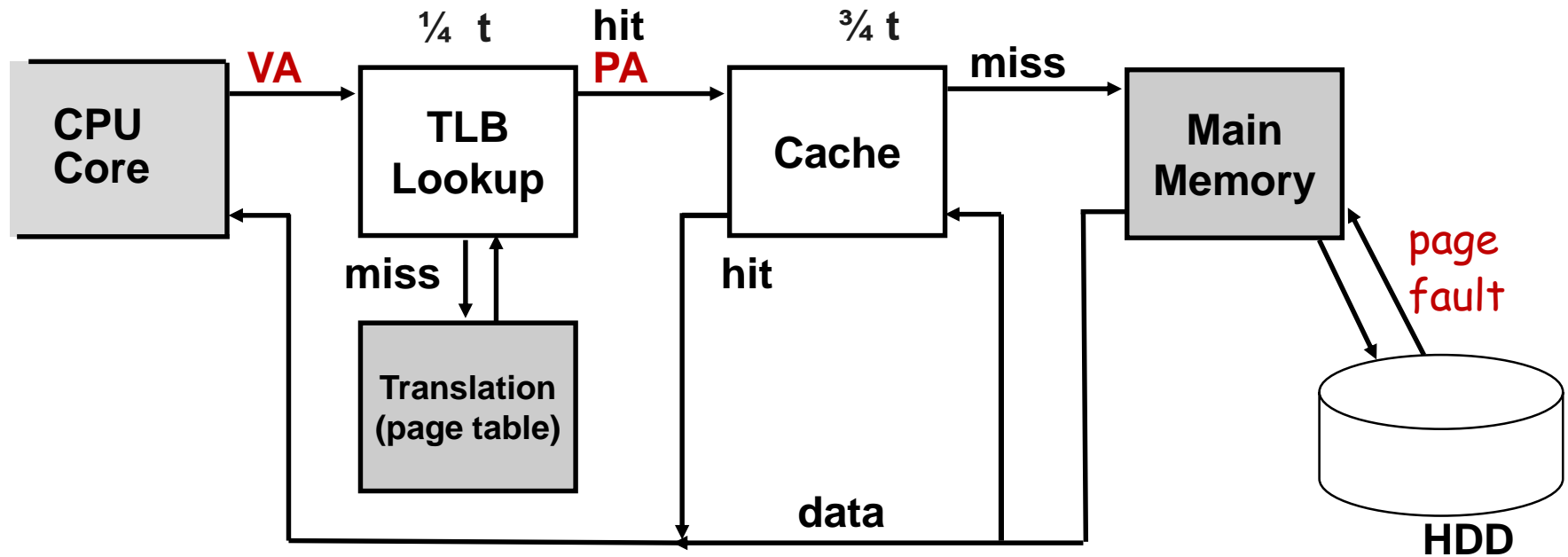
- Just like any other cache, the TLB can be organized as fully associative, set associative, or direct mapped

V	Virtual Page #	Physical Page #			

- TLB access time is typically smaller than cache access time (because TLBs are much smaller than caches)
  - TLBs are typically not more than 128 to 256 entries even on high end machines

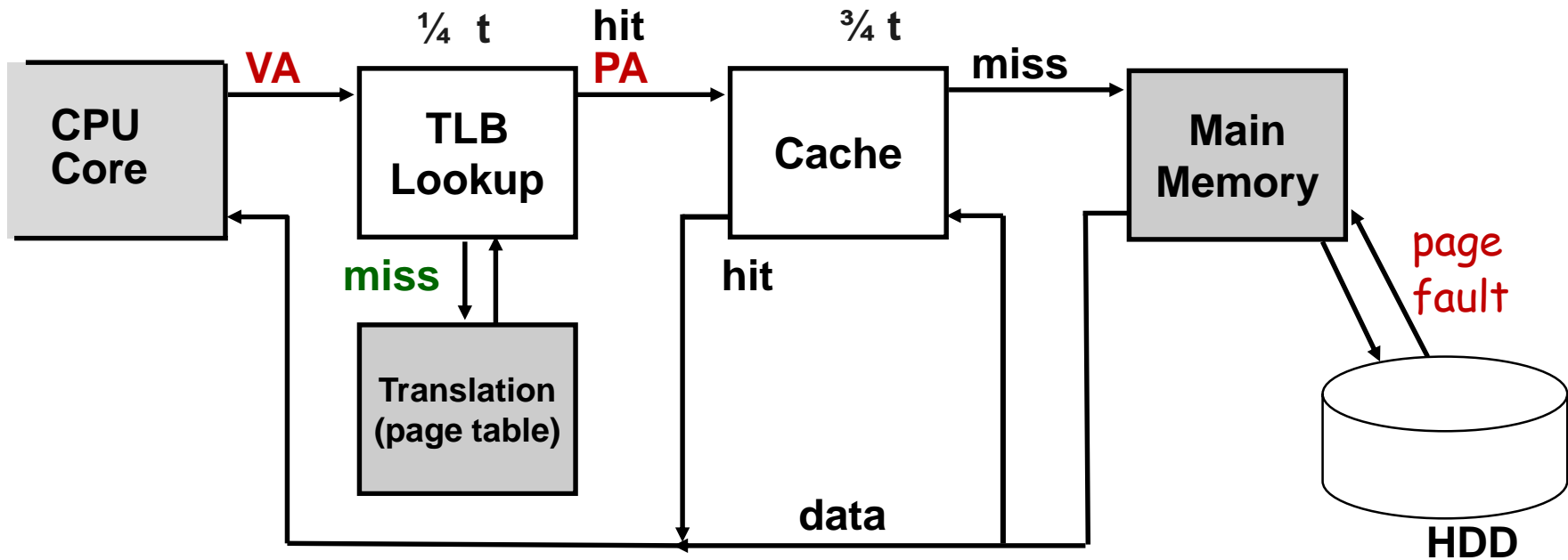


# A TLB in the Memory Hierarchy



- A **TLB miss** – is it a TLB miss or a page fault ?
  - If the page is in main memory, then the TLB miss can be handled (in hardware or software) by loading the translation information from the page table into the TLB
    - Takes 100's of cycles to find and load the translation info into the TLB
  - If the page is not in main memory, then it's a true **page fault**
    - Takes 1,000,000's of cycles to service a page fault

# A TLB in the Memory Hierarchy



- **page fault** : page is not in physical memory
- **TLB misses** are much more frequent than true page faults



# Two Machines' TLB Parameters

	<b>Intel P4</b>	<b>AMD Opteron</b>
TLB organization	<p>1 TLB for instructions and 1 TLB for data</p> <p>Both 4-way set associative</p> <p>Both use ~LRU replacement</p> <p>Both have 128 entries</p> <p>TLB misses handled in hardware</p>	<p>2 TLBs for instructions and 2 TLBs for data</p> <p>Both L1 TLBs fully associative with ~LRU replacement</p> <p>Both L2 TLBs are 4-way set associative with round-robin LRU</p> <p>Both L1 TLBs have 40 entries</p> <p>Both L2 TLBs have 512 entries</p> <p>TBL misses handled in hardware</p>



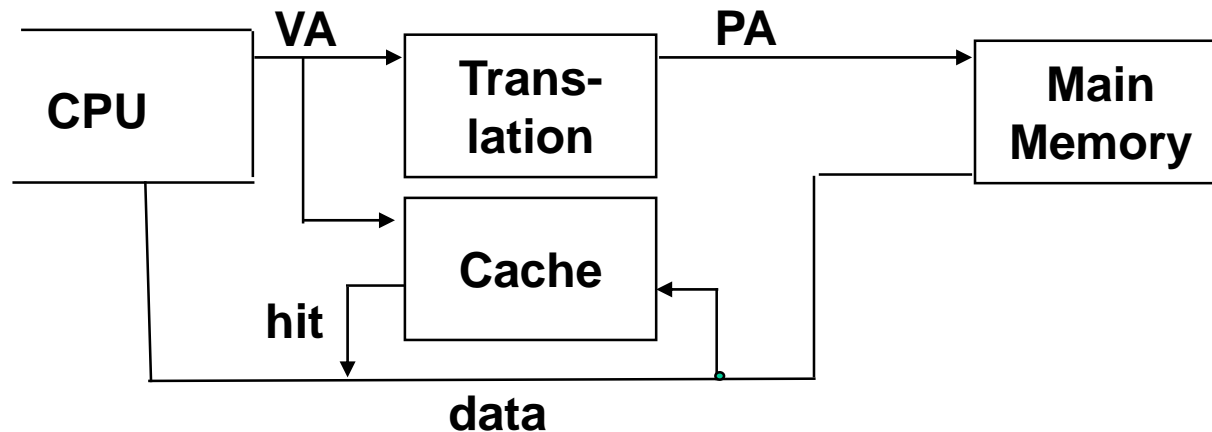
# TLB Event Combinations

TLB	Page Table	Cache	Possible? Under what circumstances?
Hit	Hit	Hit	Yes – what we want!
Hit	Hit	Miss	Yes – although the page table is not checked if the TLB hits
Miss	Hit	Hit	Yes – TLB miss, PA in page table
Miss	Hit	Miss	Yes – TLB miss, PA in page table, but data not in cache
Miss	Miss	Miss	Yes – page fault
Hit	Miss	Miss/ Hit	Impossible – TLB translation not possible if page is not present in memory
Miss	Miss	Hit	Impossible – data not allowed in cache if page is not in memory



# Why Not a Virtually Addressed Cache?

- A **virtually addressed cache** would only require address translation on cache misses



but

- Two different virtual addresses can map to the same physical address (when processes are sharing data),
- Two different cache entries hold data for the same physical address
  - **synonyms** (別名)
    - Must update all cache entries with the same physical address or the memory becomes inconsistent

# The Hardware/Software Boundary

- What parts of the virtual to physical address translation is done by or assisted by the hardware?
  - **Translation Lookaside Buffer (TLB)** that caches the recent translations
    - TLB access time is part of the cache hit time
    - May cause an extra stage in the pipeline for TLB access
  - Page table storage, fault detection and updating
    - Page faults result in interrupts (precise) that are then handled by the **OS**
    - Hardware must support (i.e., update appropriately) Dirty and Reference bits (e.g., ~LRU) in the Page Tables



# Q3 2022 Hard Drive Failure Rates

annualized failure rate (AFR)

## Backblaze SSD Quarterly Failure Rates for Q2 2022

Reporting period: 4/1/22 thru 6/30/22 for drive models active as of 6/30/22

MFG	Model	Size (GB)	Drive Count	Drive Days	Drive Failures	AFR
Crucial	CT250MX500SSD1	250	272	20,002	0	-
Dell	DELLBOSS VD	480	351	29,066	0	-
Micron	MTFDDAV240TCB	240	89	8,084	1	4.52%
Seagate	ZA25OCM10003	250	1,106	99,379	2	0.73%
Seagate	ZA500CM10003 (*)	500	3	42	0	-
Seagate	ZA2000CM10002	2000	3	271	0	-
Seagate	ZA25OCM10002	250	559	50,477	4	2.89%
Seagate	ZA500CM10002	500	18	1,625	0	-
Seagate	ZA25ONM1000 (*)	250	9	126	0	-
Seagate	SSD	300	106	9,541	0	-
WDC	WDS250G2BOA	250	42	3,781	0	-
			<b>2,558</b>	<b>222,394</b>	<b>7</b>	<b>1.15%</b>

(\*) - New drive model in Q2 2022



<https://www.backblaze.com/blog/ssd-drive-stats-mid-2022-review/>

<https://www.backblaze.com/blog/backblaze-drive-stats-for-q3-2022/>

## Backblaze Hard Drives Quarterly Failure Rates for Q3 2022

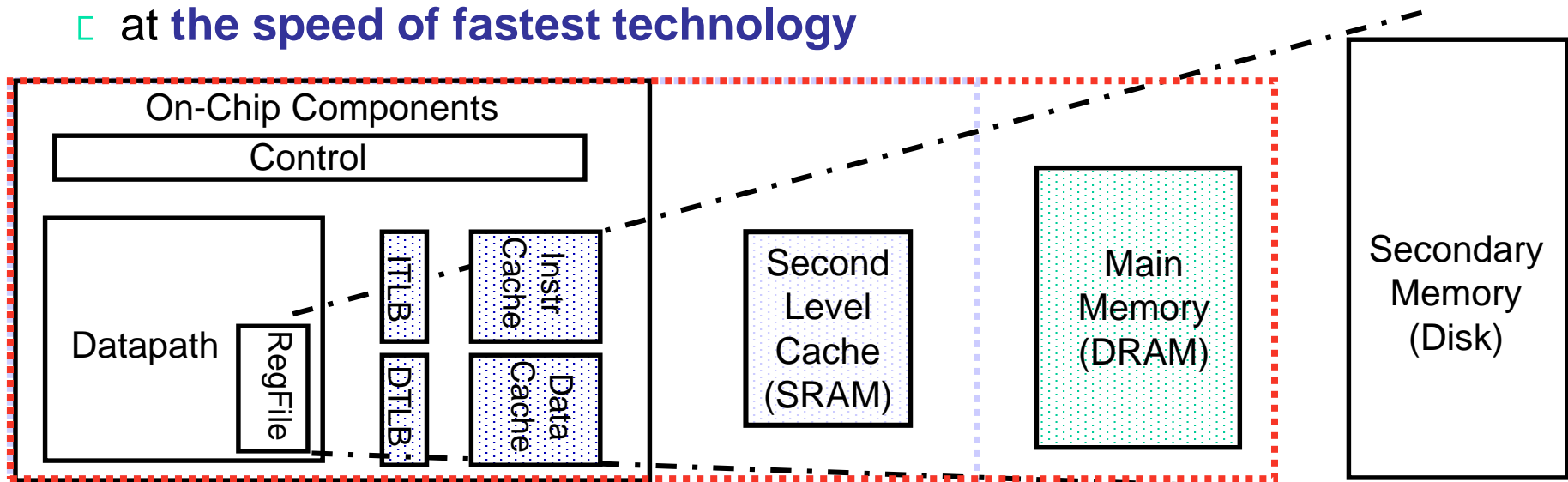
Reporting period: 7/1/2022 through 9/30/2022 for drive models active as of 9/30/2022

MFG	Model	Drive Size	Drive Count	Avg. Age (months)	Drive Days	Drive Failures	AFR
HGST	HMS5C4040ALE640	4TB	3,731	74.0	341,509	3	0.32%
HGST	HMS5C4040BLE640	4TB	12,730	71.1	1,170,925	14	0.44%
HGST	HUH728080ALE600	8TB	1,119	53.6	103,354	8	2.83%
HGST	HUH728080ALE604	8TB	95	62.6	7,637	-	0.00%
HGST	HUH721212ALE600	12TB	2,605	35.9	239,644	3	0.46%
HGST	HUH721212ALE604	12TB	13,157	18.3	1,209,798	19	0.57%
HGST	HUH721212ALN604	12TB	10,784	41.8	992,989	27	0.99%
Seagate	ST4000DM000	4TB	18,292	83.1	1,683,920	202	4.38%
Seagate	ST6000DX000	6TB	886	89.6	81,509	3	1.34%
Seagate	ST8000DM002	8TB	9,566	71.6	883,015	62	2.56%
Seagate	ST8000NM000A	8TB	79	11.2	26,974	-	0.00%
Seagate	ST8000NM0055	8TB	14,374	60.7	1,322,195	107	2.95%
Seagate	ST10000NM0086	10TB	1,174	58.6	108,372	9	3.03%
Seagate	ST12000NM0007	12TB	1,272	34.7	117,739	16	4.96%
Seagate	ST12000NM0008	12TB	19,910	30.1	1,837,021	124	2.46%
Seagate	ST12000NM001G	12TB	12,530	22.1	1,146,368	35	1.11%
Seagate	ST14000NM001G	14TB	10,737	19.9	987,184	40	1.48%
Seagate	ST14000NM0138	14TB	1,535	21.8	142,894	36	9.20%
Seagate	ST16000NM001G	16TB	20,402	10.7	1,696,759	29	0.62%
Seagate	ST16000NM002J	16TB	310	3.6	22,105	2	3.30%
Toshiba	MDO4ABA400V	4TB	95	88.3	8,849	2	8.25%
Toshiba	MG07ACA14TA	14TB	38,203	23.1	3,514,384	117	1.22%
Toshiba	MG07ACA14TEY	14TB	537	18.4	47,742	2	1.53%
Toshiba	MG08ACA16TA	16TB	3,751	3.9	243,198	5	0.75%
Toshiba	MG08ACA16TE	16TB	5,942	11.7	546,805	22	1.47%
Toshiba	MG08ACA16TEY	16TB	4,244	11.9	385,715	12	1.14%
WDC	WUH721414ALE6L4	14TB	8,409	21.8	773,557	5	0.24%
WDC	WUH721816ALE6LO	16TB	2,702	11.8	248,428	-	0.00%
WDC	WUH721816ALE6L4	16TB	7,138	2.8	310,502	6	0.71%
			<b>226,309</b>		<b>20,201,091</b>	<b>910</b>	<b>1.64%</b>



# A Typical Memory Hierarchy

- By taking advantage of **the principle of locality**
  - Present **much memory** in the **cheapest technology**
  - at **the speed of fastest technology**



<b>Speed (%cycles):</b>	½'s	1's	10's	100's	1,000's
<b>Size (bytes):</b>	100's	K's	10K's	M's	G's to T's
<b>Cost:</b>	highest				lowest

TLB: Translation Lookaside Buffer

2023年度(令和5年)版

Ver. 2023-11-08a

Course number: CSC.T363

# コンピュータアーキテクチャ Computer Architecture

スーパースカラ  
Superscalar

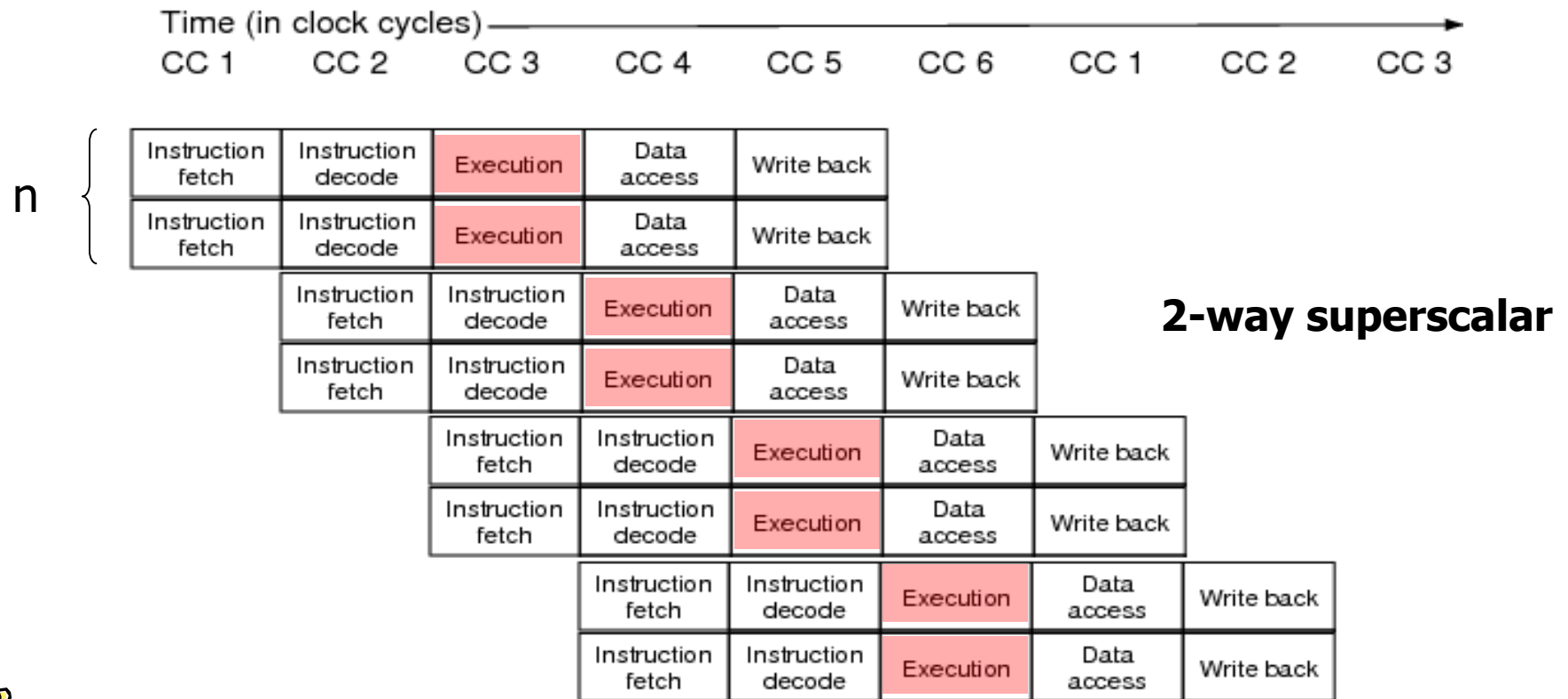
[www.arch.cs.titech.ac.jp/lecture/CA/](http://www.arch.cs.titech.ac.jp/lecture/CA/)  
Tue 13:30-15:10, 15:25-17:05  
Fri 13:30-15:10

吉瀬 謙二 情報工学系  
Kenji Kise, Department of Computer Science  
kise\_at\_c.titech.ac.jp

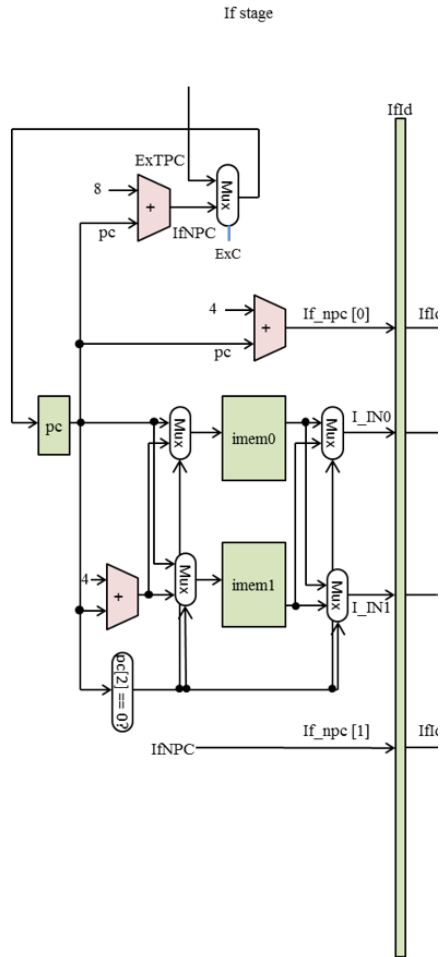


# Superscalar スーパースカラと命令レベル並列性

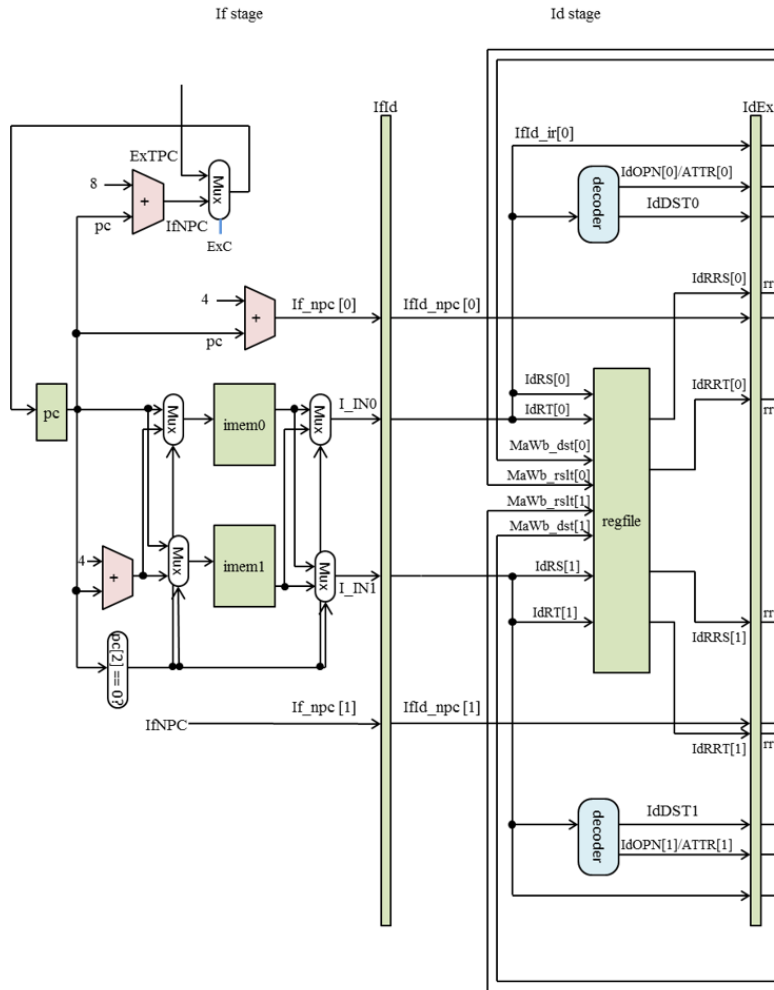
- 複数のパイプラインを利用して IPC (instructions per cycle) を 1以上に引き上げる, 複数の命令を並列に実行
  - n-way スーパースカラ



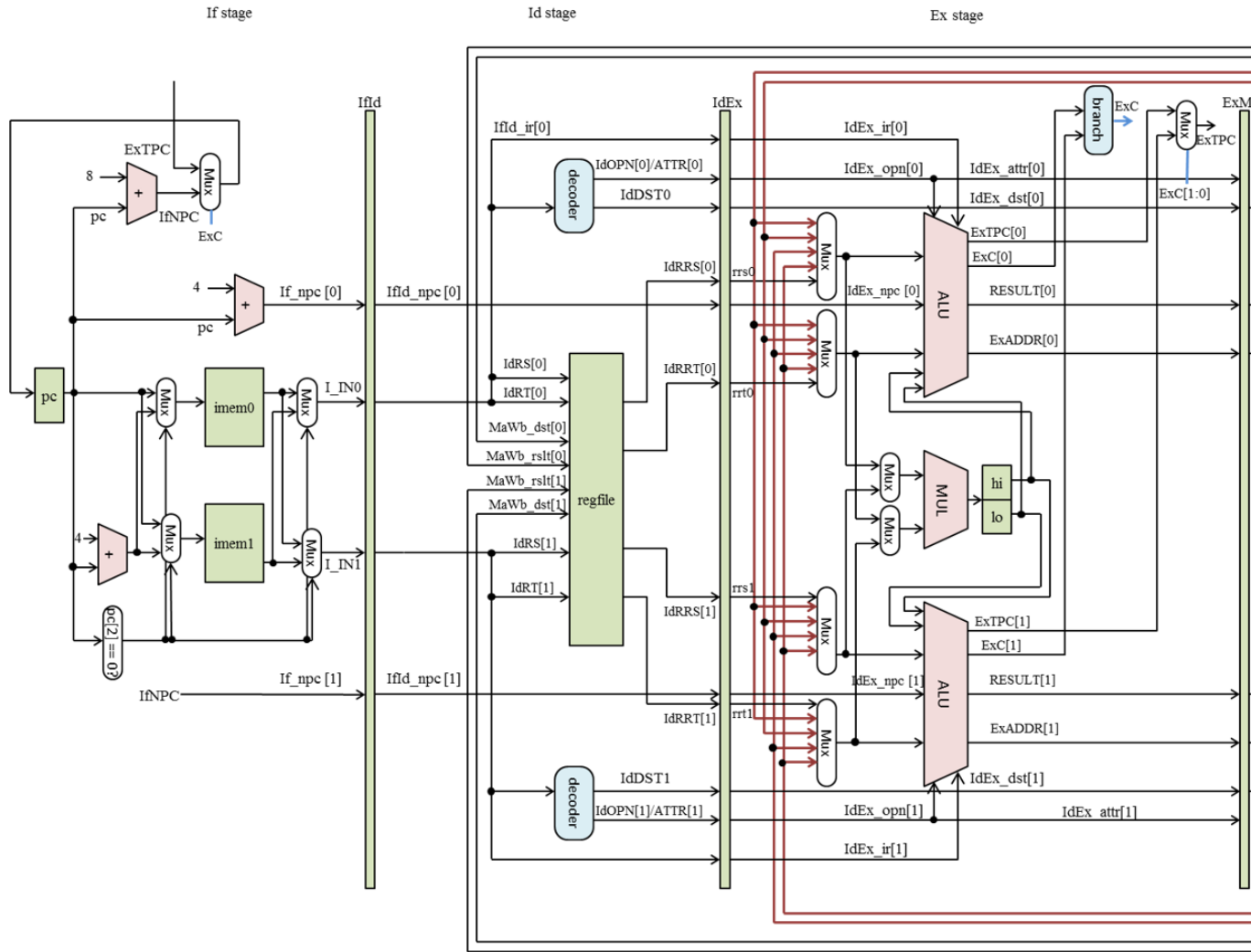
# スーパースカラプロセッサ (MIPSインタリーブ命令メモリ版)



# スーパースカラプロセッサ (MIPSインタリーブ命令メモリ版)



# スーパースカラプロセッサ (MIPSインタリーブ命令メモリ版)



# スーパースカラプロセッサ (MIPSインタリーブ命令メモリ版)

