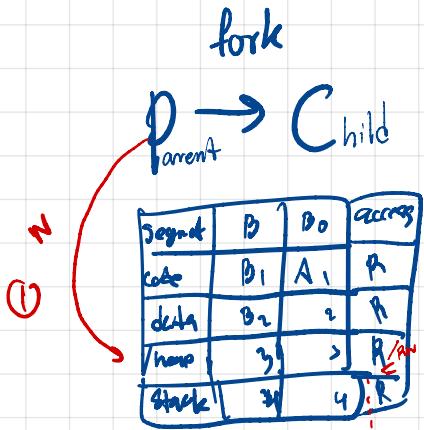


# Virtual Memory



- និងការរាយការណ៍
- ១ ចូលរួម RAM
  - ២ Child ការពារទិន្នន័យ
  - ៣ ដឹងទែរការ: Ready និងទិន្នន័យ

exception ?

③

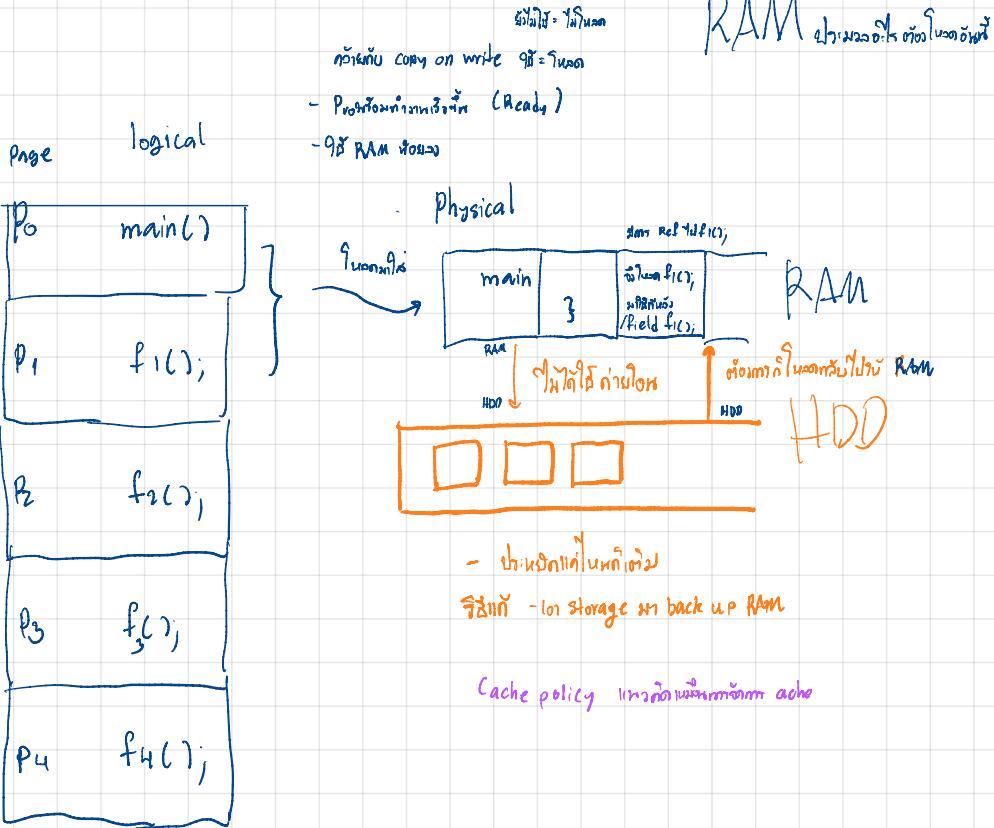
B<sub>3</sub> copy



ទិន្នន័យ

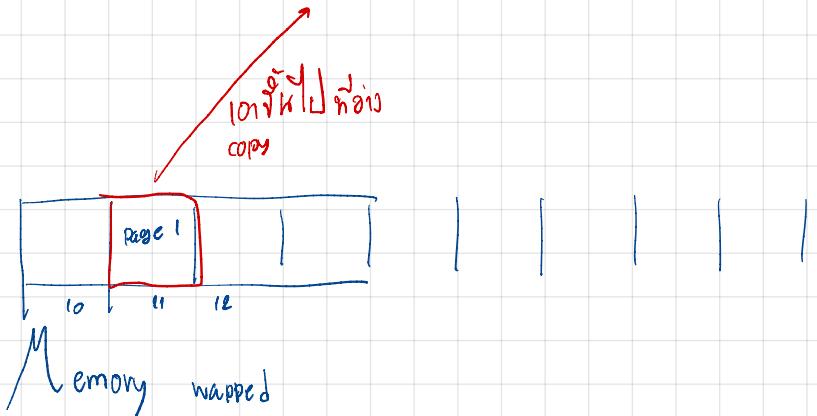
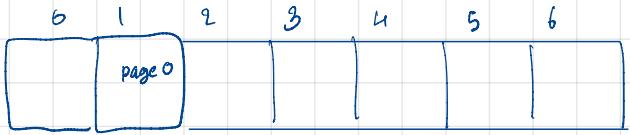
COPY ON WRITE

នៅ Copy តិចដែលការពារនូវ W



**f<sub>open</sub>**

RAM



frame	access	
	page	action
0	1	R
1	11	invalid
2		found valid RAM address → exception
3		invalid → exception

Annotations:

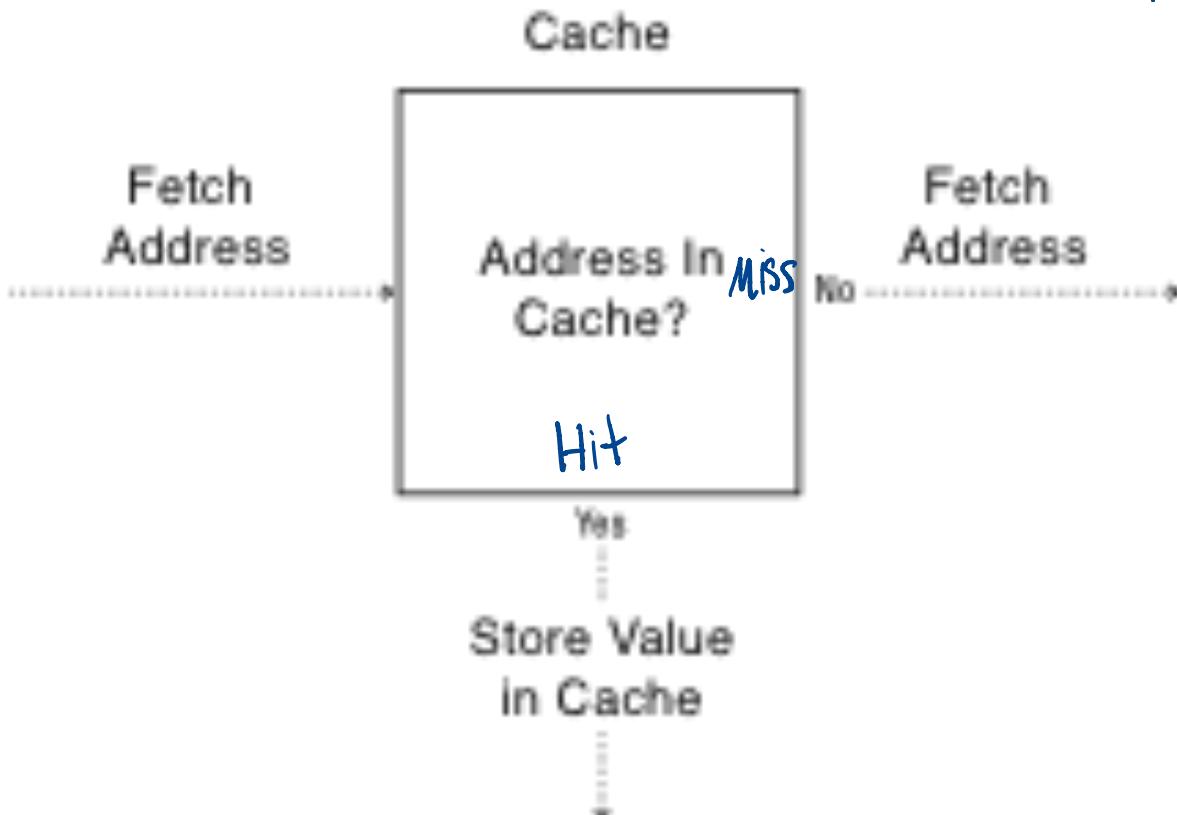
- copy 111111111111
- in frame 111111111111 copy to HDD
- then swap
- ① 111111111111
- ② 111111111111
- with 111111111111

# Definitions

- Cache
  - Copy of data that is faster to access than the original
  - Hit: if cache has copy
  - Miss: if cache does not have copy
- Cache block ប្រអប់ដីនា
  - Unit of cache storage (multiple memory locations)
- Temporal locality ការវិភ័យចុងឆ្នាំ ឱកាសមូលដ្ឋាន: loop
  - Programs tend to reference the same memory locations multiple times
  - Example: instructions in a loop
- Spatial locality បិទការ ទំនាក់ទំនងការ
  - Programs tend to reference nearby locations
  - Example: data in a loop

# Cache Concept (Read)

*processor request in cache*



# Cache Concept (Write)

write through

write through

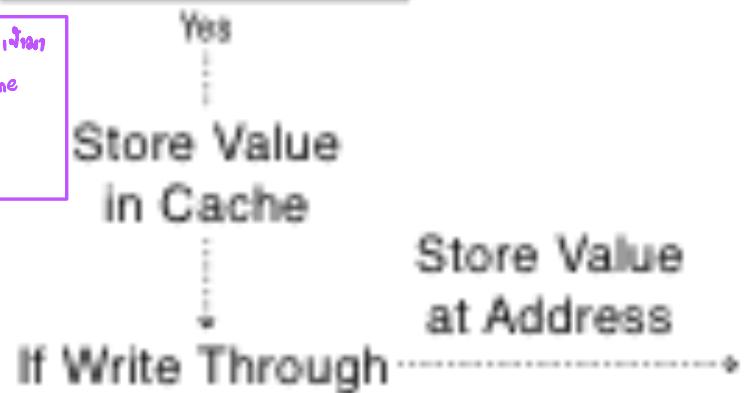


Write through: changes sent immediately to next level of storage

ក្រុមហ៊ែវនៃការផ្តល់

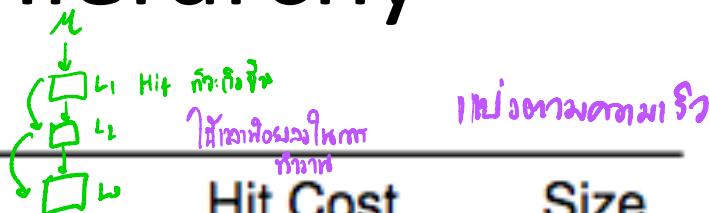
ស្ថិតិយោគ នូវបញ្ជី  
ក្នុងការផ្តល់ទៅការណ៍

Write back: changes stored in cache until cache block is replaced  
ក្នុងការផ្តល់ទៅការណ៍ — ដែលត្រូវបានដោះស្រាយឡើង



ដែលត្រូវបានដោះស្រាយឡើង

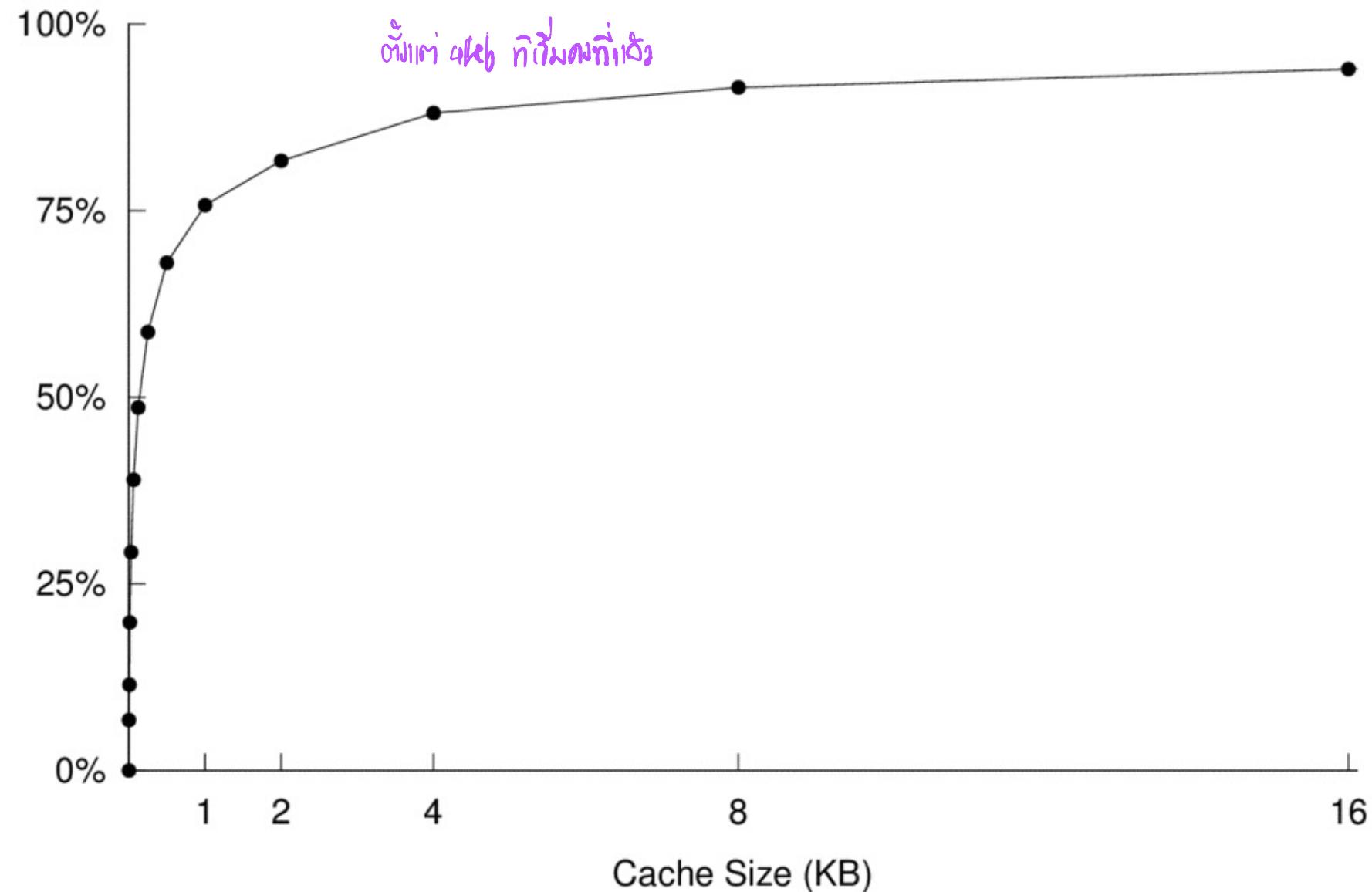
# Memory Hierarchy



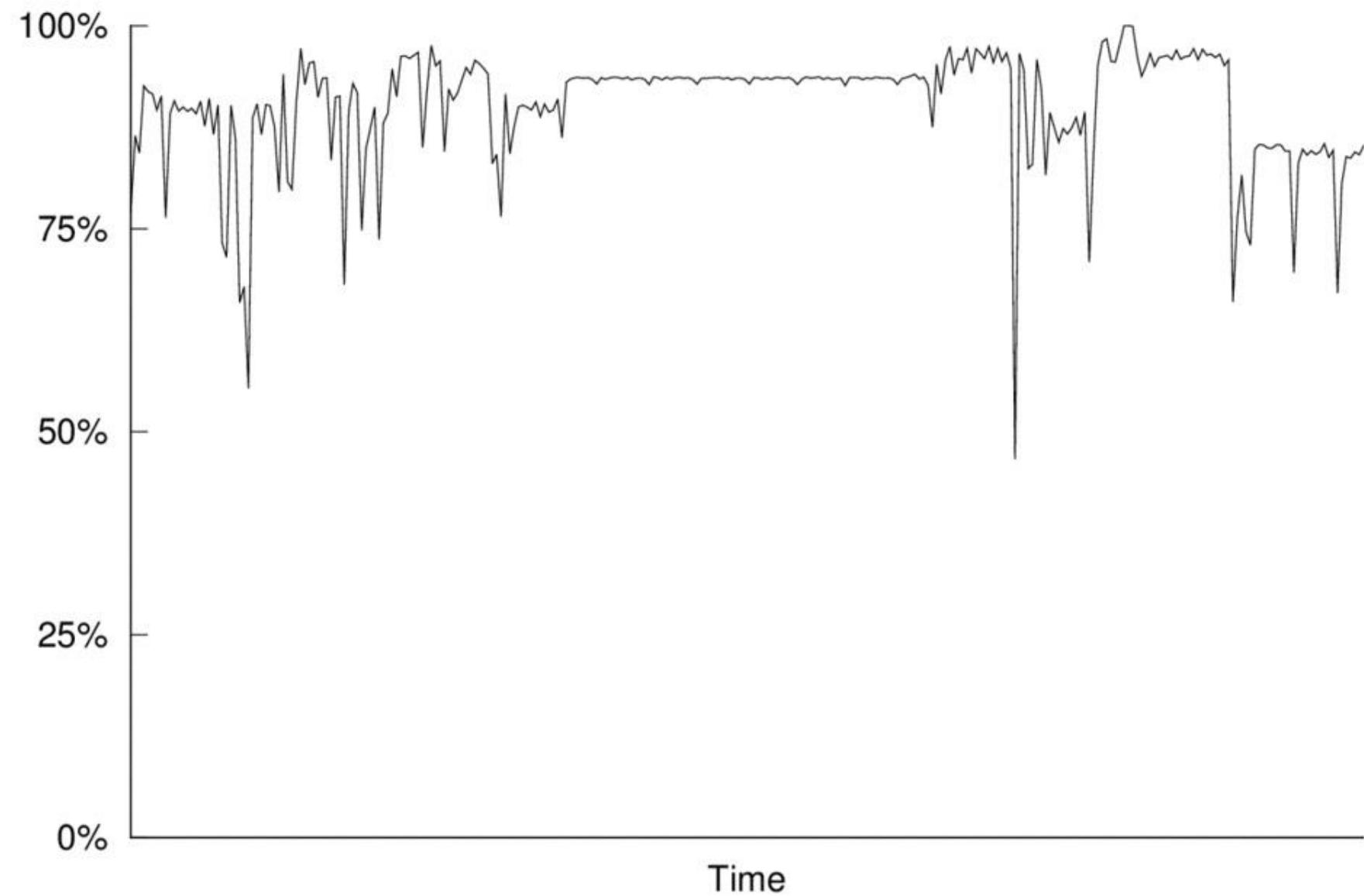
Cache	Hit Cost	Size
1st level cache/first level TLB	1 ns	64 KB
2nd level cache/second level TLB	4 ns	256 KB
3rd level cache	12 ns	2 MB
Memory (DRAM) ข้อมูล เก็บอยู่	100 ns	10 GB
Data center memory (DRAM) ข้อมูล สำหรับ + ห้องแม่ข่าย	100 $\mu$ s	100 TB
Local non-volatile memory	100 $\mu$ s	100 GB
Local disk	10 ms	1 TB
Data center disk	10 ms	100 PB
Remote data center disk	200 ms	1 XB

i7 has 8MB as shared 3<sup>rd</sup> level cache; 2<sup>nd</sup> level cache is per-core

# Cache hit rate



# Example of cache hit rate over time



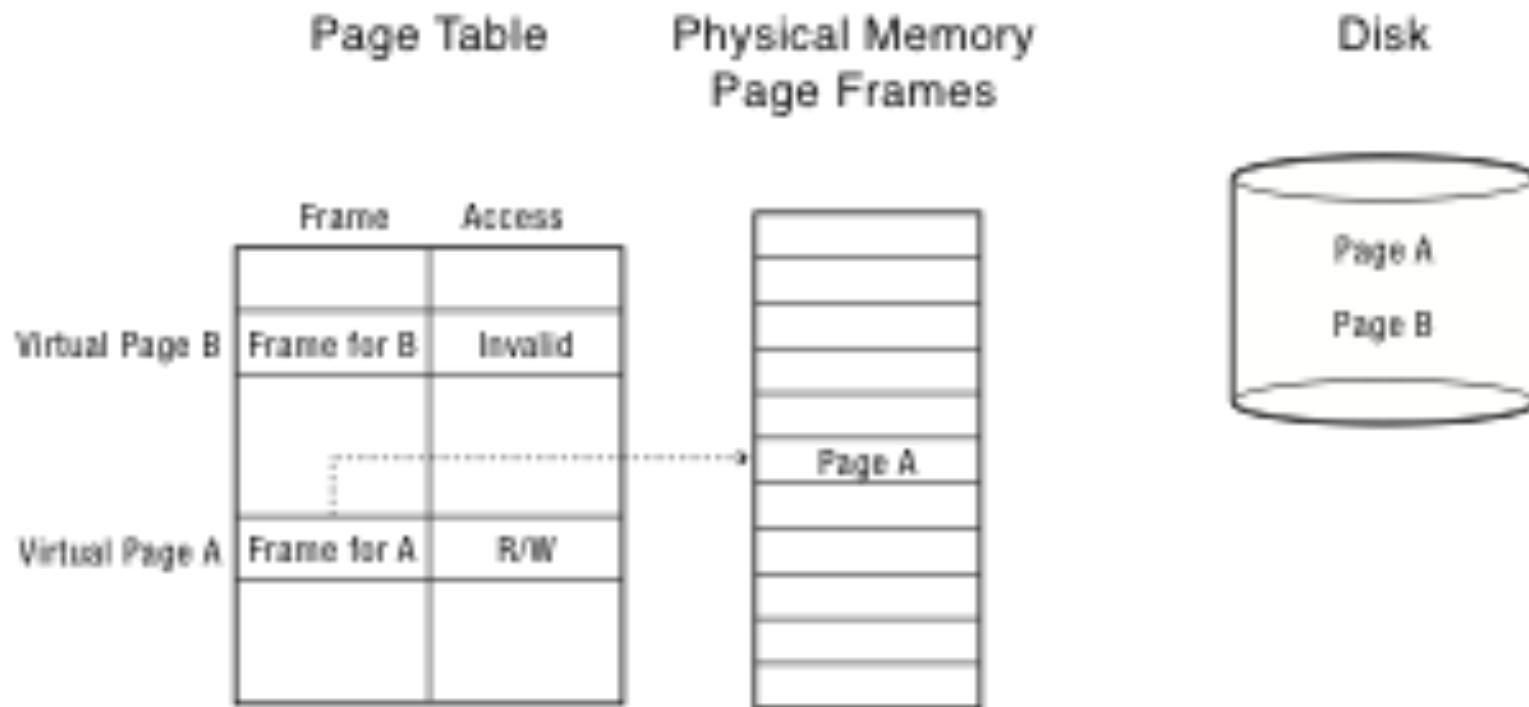
# Main Points

- Can we provide the illusion of near infinite memory in limited physical memory?
  - Demand-paged virtual memory
  - Memory-mapped files
- How do we choose which page to replace?
  - FIFO, MIN, LRU, LFU, Clock *... چົກນາຍັງ*
- What types of workloads does caching work for, and how well?
  - Spatial/temporal locality vs. Zipf workloads  
*ດ້ວຍການປ່ຽນແປງດີ*

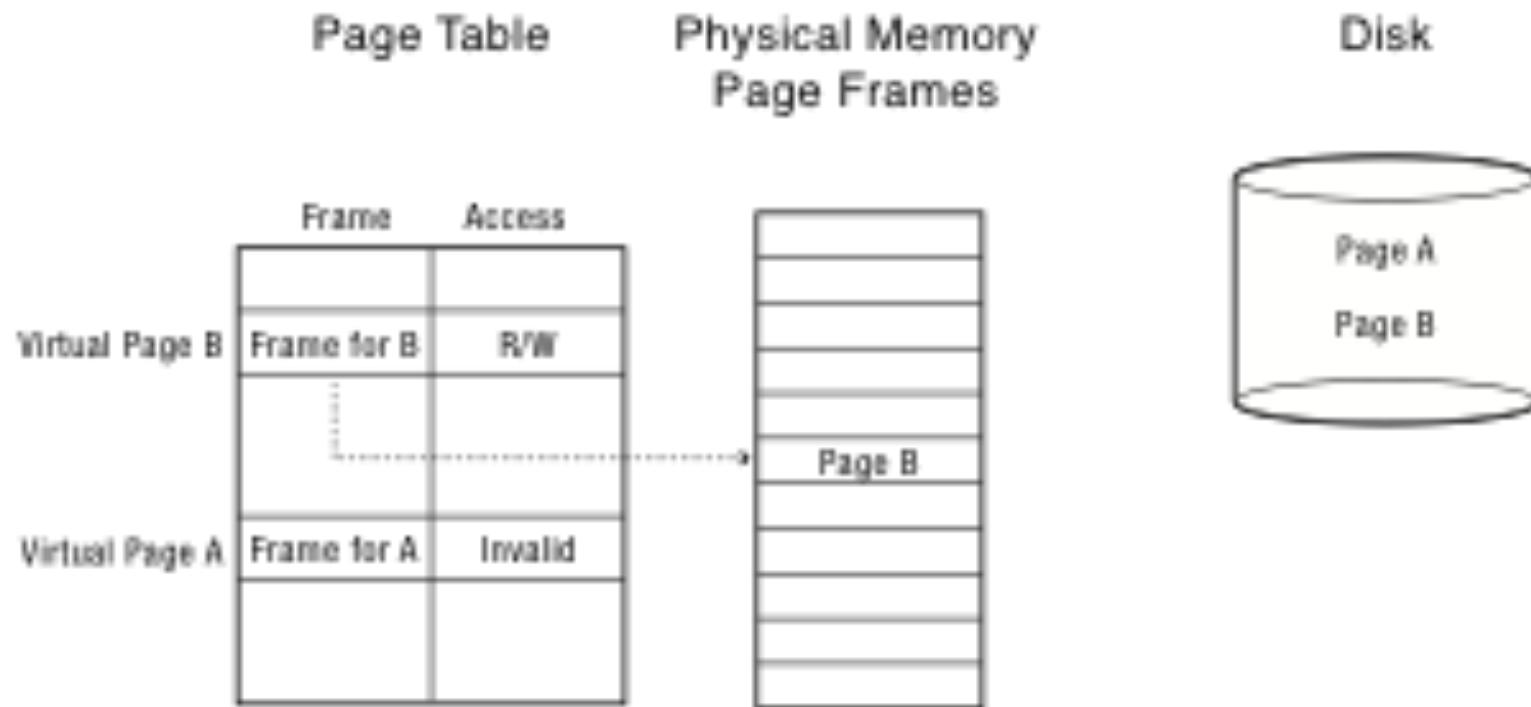
# Hardware address translation is a power tool

- Kernel trap on read/write to selected addresses
  - Copy on write
  - Fill on reference
  - Demand paged virtual memory
  - Memory mapped files
  - Modified bit emulation
  - Use bit emulation

# Demand Paging (Before)



# Demand Paging (After)



# Demand Paging

1. TLB miss
  2. Page table walk
  3. Page fault (page invalid in page table)
  4. Trap to kernel
  5. Convert virtual address to file + offset
  6. Allocate page frame
    - Evict page if needed
  7. Initiate disk block read into page frame
  8. Disk interrupt when DMA complete
  9. Mark page as valid
  10. Resume process at faulting instruction
  11. TLB miss
  12. Page table walk to fetch translation
  13. Execute instruction
- (Handwritten notes in purple ink):*
- Step 3: "Page fault (page invalid in page table)"
  - Step 7: "Initiate disk block read into page frame"
    - "Disk access from Invalid page"
    - "return"
    - "block on copy"
    - "from frame"

# Allocating a Page Frame

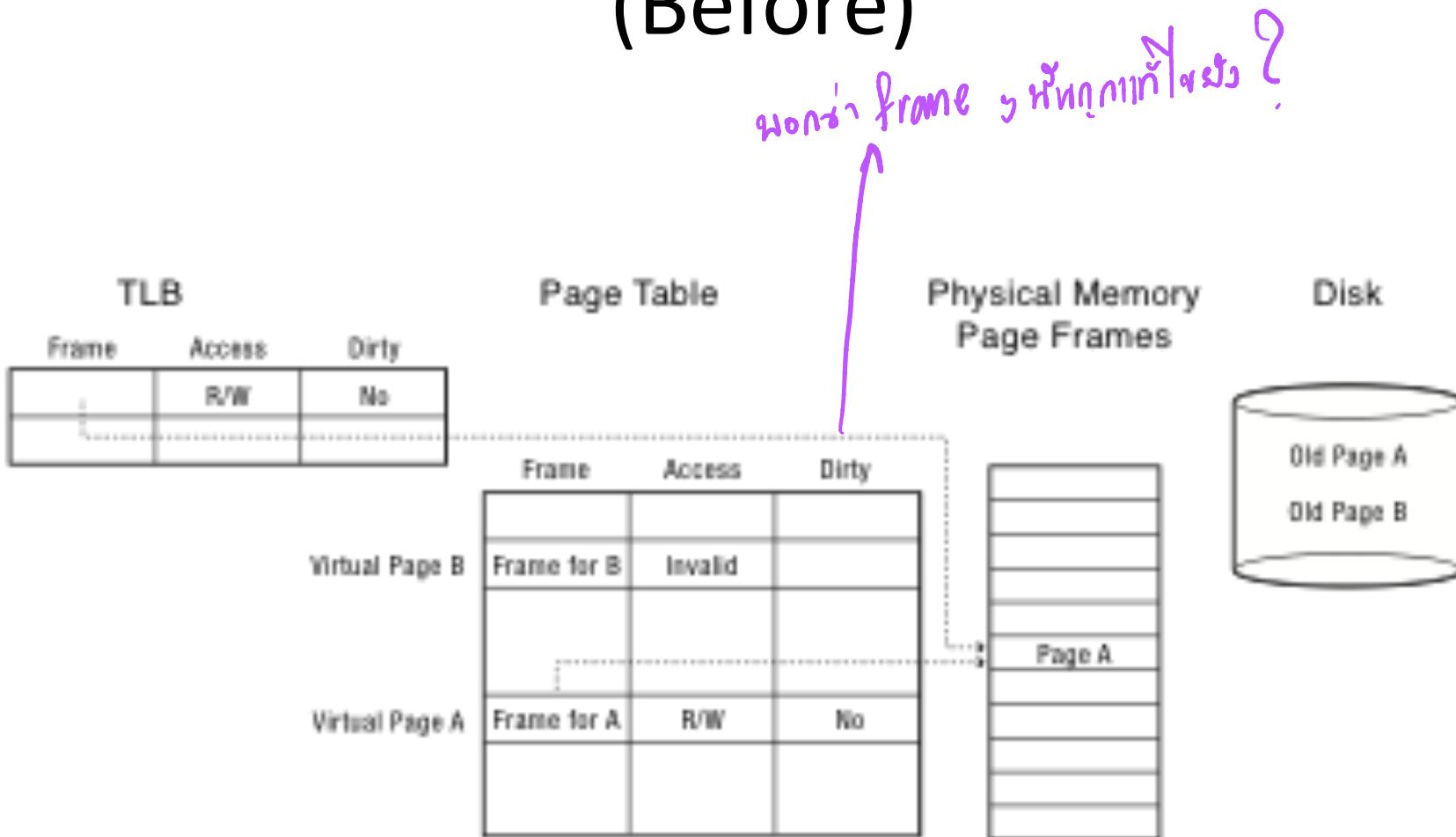
- Select old page to evict
- Find all page table entries that refer to old page
  - If page frame is shared
- Set each page table entry to invalid
- Remove any TLB entries
  - Copies of now invalid page table entry
- Write changes on page back to disk, if necessary

# How do we know if page has been modified?

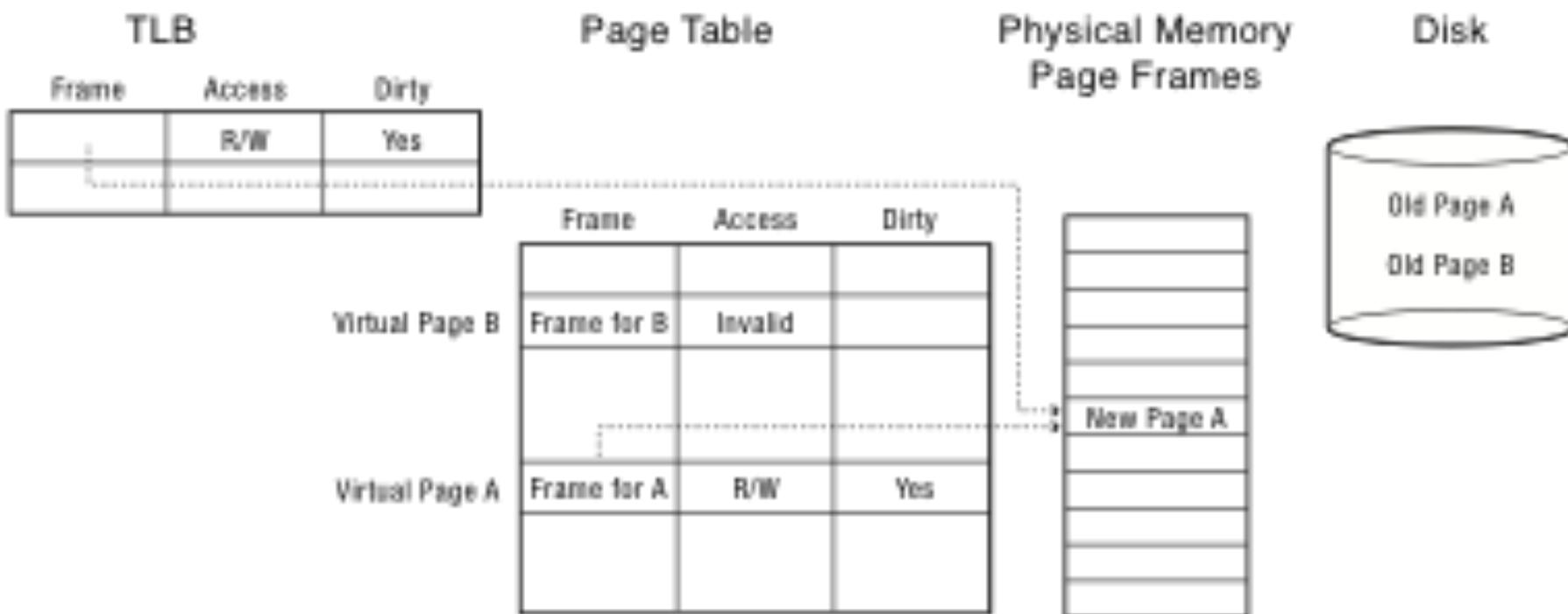
-frame 93%  
Set Array 98% bit map allocation

- Every page table entry has some bookkeeping
  - Has page been modified?
    - Set by hardware on store instruction
    - In both TLB and page table entry
  - Has page been recently used?
    - Set by hardware on in page table entry on every TLB miss
- Bookkeeping bits can be reset by the OS kernel
  - When changes to page are flushed to disk
  - To track whether page is recently used

# Keeping Track of Page Modifications (Before)



# Keeping Track of Page Modifications (After)



# Virtual or Physical Dirty/Use Bits

booleeping bit

- Most machines keep dirty/use bits in the page table entry
- Physical page is
  - Modified if *any* page table entry that points to it is modified
  - Recently used if *any* page table entry that points to it is recently used  
q&A
- On MIPS, simpler to keep dirty/use bits in the core map
  - Core map: map of physical page frames

core map

# Models for Application File I/O

- Explicit read/write system calls
  - Data copied to user process using system call
  - Application operates on data
  - Data copied back to kernel using system call
- Memory-mapped files
  - Open file as a memory segment
  - Program uses load/store instructions on segment memory, implicitly operating on the file
  - Page fault if portion of file is not yet in memory
  - Kernel brings missing blocks into memory, restarts process

# Advantages to Memory-mapped Files

- Programming simplicity, esp for large files
    - Operate directly on file, instead of copy in/copy out
  - Zero-copy I/O
    - Data brought from disk directly into page frame
  - Pipelining
    - Process can start working before all the pages are populated
  - Interprocess communication
    - Shared memory segment vs. temporary file
- |||: ฝึกงาน กับหัวเรื่อง เมมโมรี่  
2 process share page , เศรษฐกัน

# Cache Replacement Policy

- On a cache miss, how do we choose which entry to replace?
  - Assuming the new entry is more likely to be used in the near future
  - In direct mapped caches, not an issue!

សំណង់

កំណត់ប្រើ និងលក្ខណៈ

- Policy goal: reduce cache misses
  - Improve expected case performance
  - Also: reduce likelihood of very poor performance

# A Simple Policy

- Random?
  - Replace a random entry
- FIFO?
  - Replace the entry that has been in the cache the longest time
  - What could go wrong?

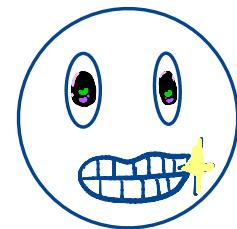
# FIFO in Action



...

Reference	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
1	A				E				D				C		
2		B				A				E			D		
3			C				B				A				E
4				D				C				B			

Worst case for FIFO is if program strides through memory that is larger than the cache



# MIN, LRU, LFU

- MIN
  - Replace the cache entry that will not be used for the longest time into the future
  - Optimality proof based on exchange: if evict an entry used sooner, that will trigger an earlier cache miss
- Least Recently Used (LRU)
  - Replace the cache entry that has not been used for the longest time in the past
  - Approximation of MIN
- Least Frequently Used (LFU)
  - Replace the cache entry used the least often (in the recent past)

MIN

ต้องการเข้ามาที่ A ก็ต้องเข้ามาที่ A แล้ว

m: 111111

① หามใน cache

↓

② หามใน RAM

? ไม่คิดจะมา

จึงหามใน RAM

เมื่อมาแล้ว หา sequence ต่อไป ที่จะมีค่าใน RAM เท่านั้น

- กรณีไม่พบ ให้ย้อนกลับ

	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
0	A <sub>1</sub> hit	B <sub>2</sub> hit	C <sub>3</sub> hit	D <sub>4</sub> hit	E <sub>5</sub> hit										
1						A <sub>1</sub> hit	B <sub>2</sub> hit	C <sub>3</sub> hit	D <sub>4</sub> hit	E <sub>5</sub> hit					
2															
3															

การแทนที่

มองดู

	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
0	A <sub>4</sub> hit		E <sub>1</sub> hit			D		C							
1	B <sub>3</sub> hit			A <sub>2</sub> hit			E		D						
2	C <sub>2</sub> hit			B <sub>1</sub> hit			A			E					
3	D <sub>1</sub> hit			C <sub>0</sub> hit			B								



MJN

	A	B	A	C	B	D	A	D	E	D	A	X	E	B	A	C
0	A		+hit			+				+		+		+		
1		B			+											
2			C				E			+						
3				D		+		+	+							

hit 9

LRU ≈ MJN  
לפי הילך

LRU

	A	B	A	C	B	D	A	D	E	D	A	X	E	B	A	C
0	A		+hit			+				+		+		+		
1		B			+					+		+				
2			C				E			+						
3				D		+		+					C			

hit 9

FIFO

לפי הילך FIFO

	A	B	A	C	B	D	A	D	E	D	A	X	E	B	A	C
0	A		+hit			+				E		+		+		
1		B			+						A		+		+	
2			C				E				B					
3				D		+		+			C					

hit 7

# LRU/MIN for Sequential Scan

LRU															
Reference	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
1	A				E				D				C		
2		B				A				E			D		
3			C				B				A			E	
4				D				C				B			

MIN														
1	A					+				+			+	
2		B					+				+	C		
3			C					+	D				+	
4				D	E				+				+	

**LRU**

Reference	A	B	A	C	B	D	A	D	E	D	A	E	B	A	C
1	A			+				+			+			+	
2		B			+								+		
3				C					E			+			
4					D			+		+					C

**FIFO**

1	A		+			+		E			+				
2		B			+					A			+		
3				C								B			
4					D			+		+					C

**MIN**

1	A		+			+				+			+		
2		B			+								+		C
3				C					E			+			
4					D			+		+					

ඩිමුන්ස්

පෙරාache ගුණු Hit මූද්‍යව

# Belady's Anomaly

ache 3 ඇත්තේ hit 3 නො

FIFO (3 slots)													
Reference	A	B	C	D	A	B	E	A	B	C	D	E	
1	A			D			E						+
2		B			A			+		C			
3			C			B			+		D		

FIFO (4 slots)													
Reference	A	B	C	D	E	F	G	H	I	J	K	L	M
1	A				+			E			D		
2		B				+			A			E	
3			C						B				
4				D						C			

# Recap

ជូនការបង្កើត  
Least Recently Used

- MIN is optimal
  - replace the page or cache entry that will be used farthest into the future
- LRU is an approximation of MIN
  - For programs that exhibit spatial and temporal locality

# Openbook

- Multiple

- 10 ข้อ

อย่างที่ - เก็ง

ฟอร์มแบบง่ายๆ ตัวเลข 2 ตัวบน / เลือกมากกว่า ตัวบน - ตามสั่นต่อ

1 2 3 4 5 6 7 8 9 10

เขียน 9 ตัวบน ก็เขียนตัวเลขให้สี่ตัวบน น่าจะเขียนให้สี่ตัวบน