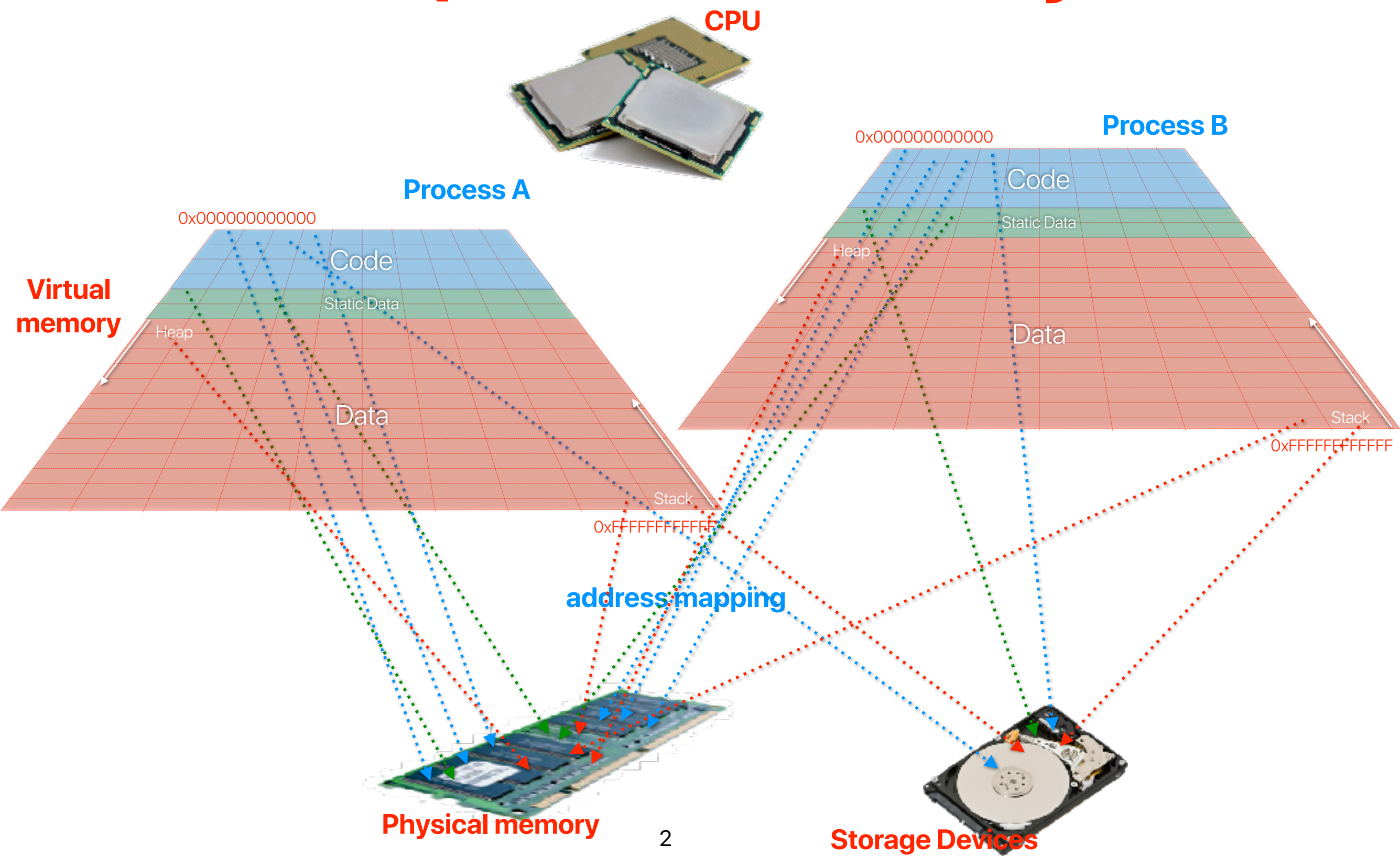


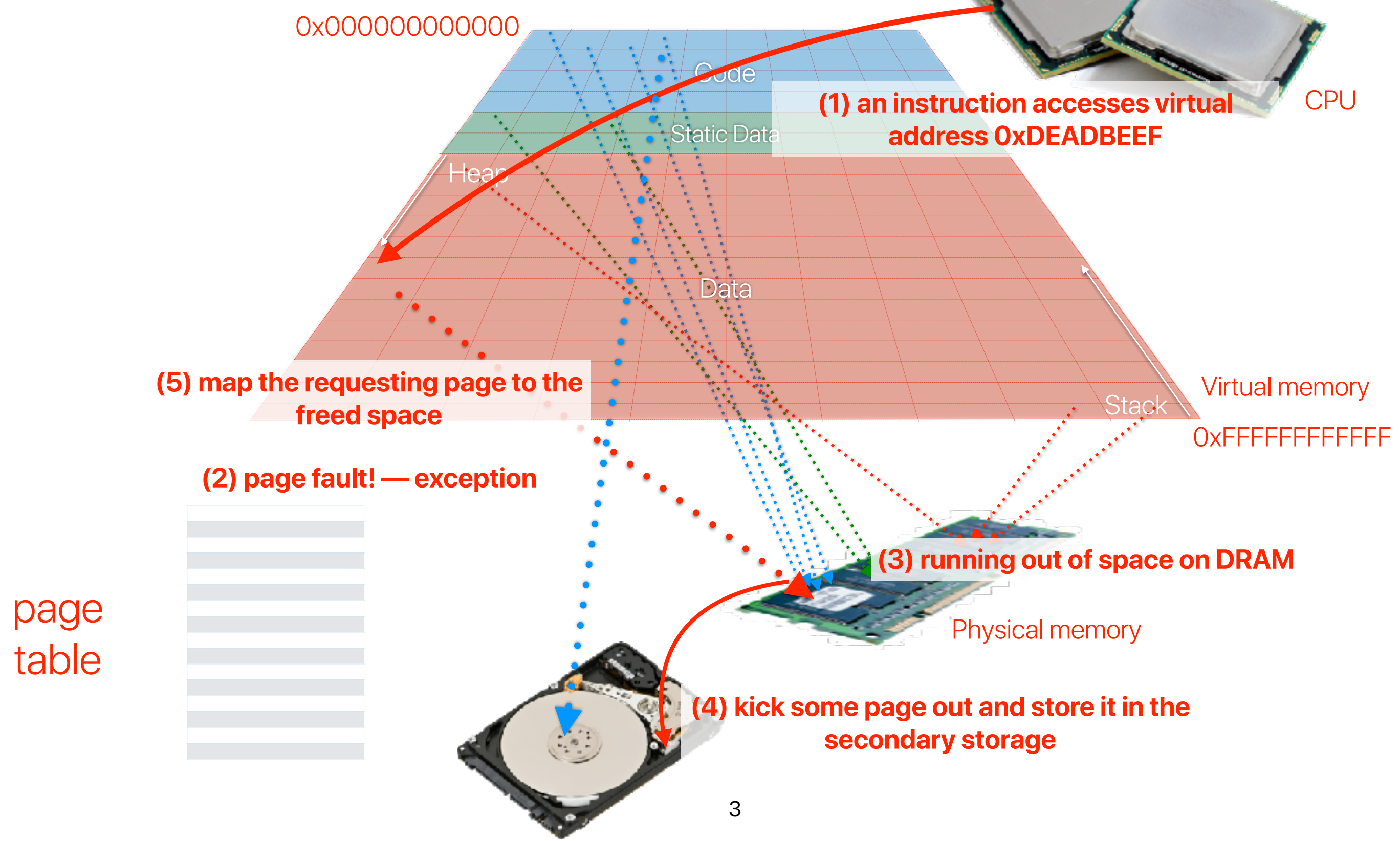
Virtual Memory (IV) — Policies

Hung-Wei Tseng

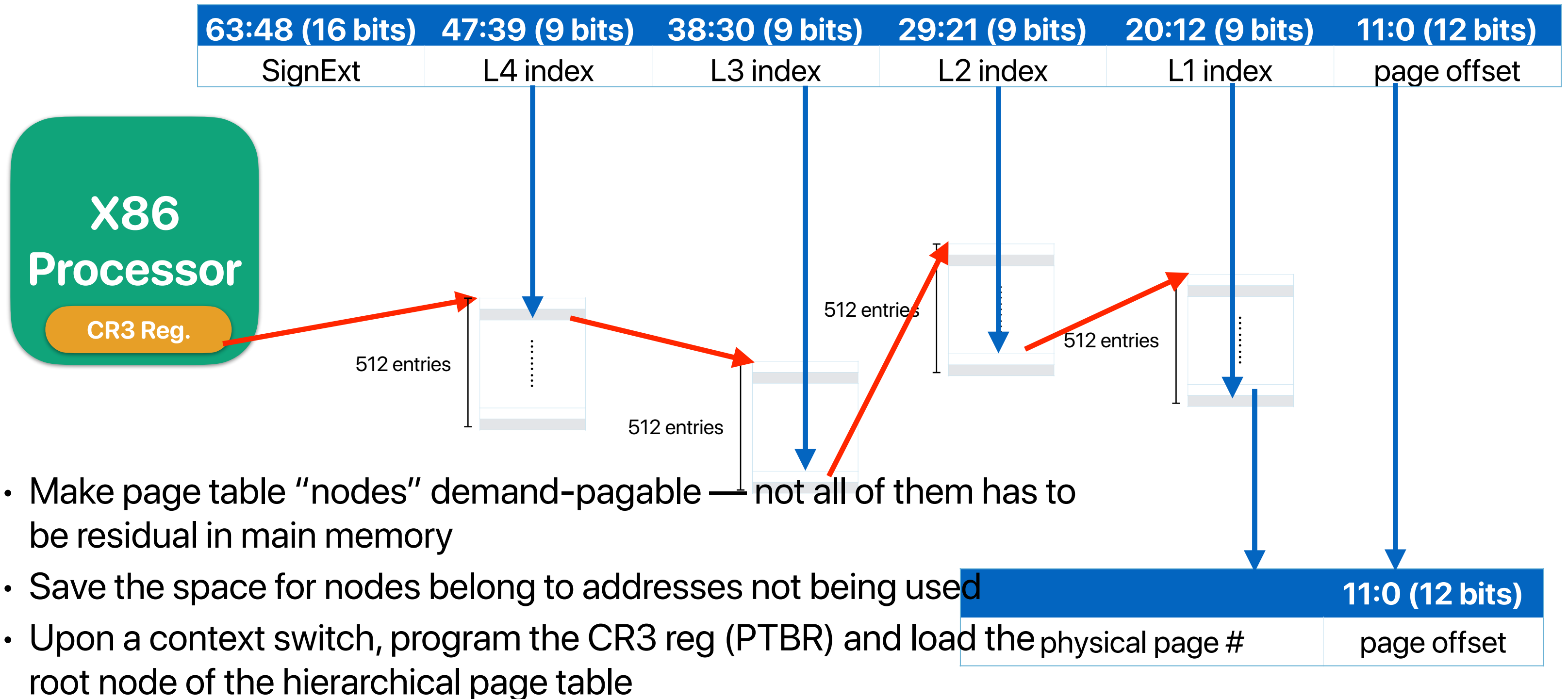
Recap: Virtual Memory



Recap: Demand paging + Swapping



Recap: Hierarchical page table to make paging feasible






Recap: Page replacement policy

- Goal: Identify page to remove that will avoid future page faults (i.e. utilize locality as much as possible)
- Implementation Goal: Minimize the amount of software and hardware overhead
 - Example:
 - Memory (i.e. RAM) access time: 100ns
 - Disk access time: 10ms
 - P_f : probability of a page fault
 - Effective Access Time = $10^{-7} + P_f * 10^{-3}$
 - When $P_f = 0.001$:
Effective Access Time = 10,100ns
 - **Takeaway: Disk access tolerable only when it is extremely rare**

What VAX/VMS proposed to achieve these goals?

- Considering the optimization goals and the proposed VAX/VMS mechanisms, which of the following combinations is incorrect?

| Goal | | Optimization | |
|--|----------------------------------|--------------|---|
|  A | Process startup cost | W | Demand-zero & copy-on-reference |
|  B | Process performance interference | X | Process-local replacement |
| C | Page table lookup overhead | Y | Page clustering also helps reduce disk loads |
|  D | Paging load on disks | Z | Page caching |

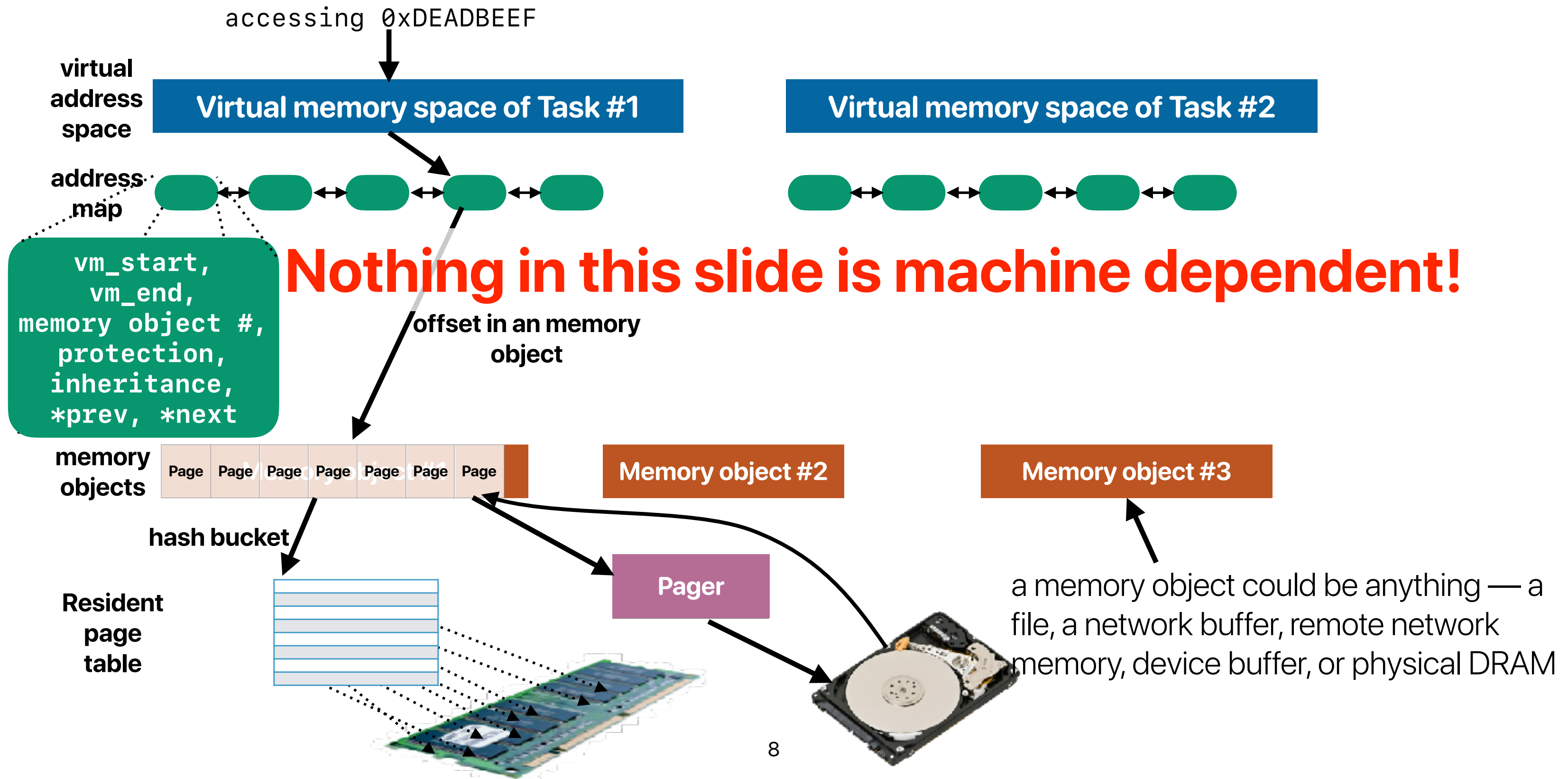
- We're still using their proposed techniques almost everyday!
- It's basically the baseline UNIX VM design

Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures

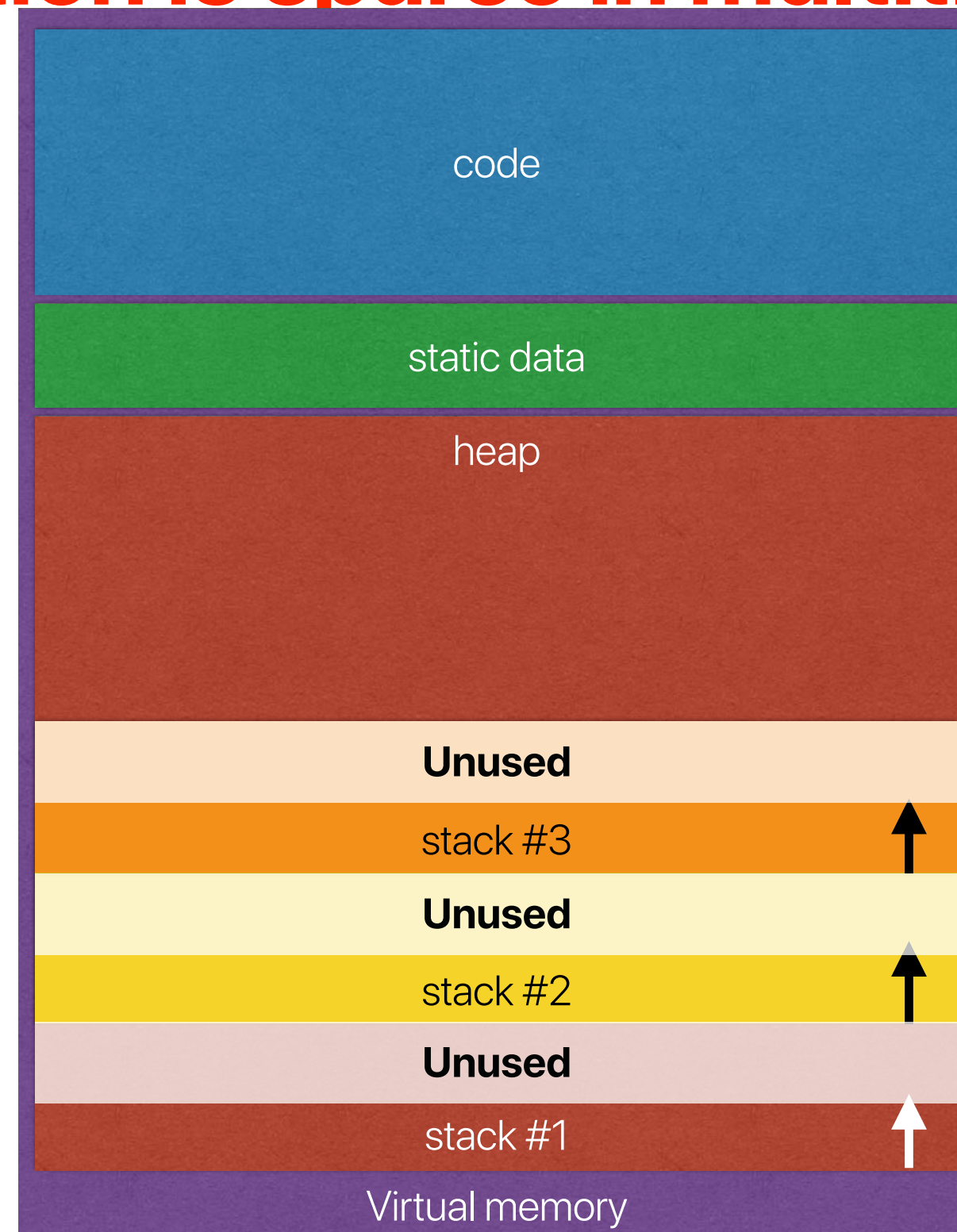
**Richard Rashid, Avadis Tevanian, Michael Young, David Golub, Robert Baron, David Black,
William Bolosky, and Jonathan Chew**

Carnegie-Mellon University, NeXT, University of Rochester

Overview of Mach's VM



Address allocation is sparse in multithreading model!



address
map



Outline

- Page replacement policies
- Page replacement policy once used in UNIX: Converting a Swap-Based System to do Paging in an Architecture Lacking Page-Reference Bits
- Another popular page replacement policy: WSClock – A Simple and Effective Algorithm for Virtual Memory Management

Page replacement policies from textbooks

Page replacement policy

- We need to determine:
 - Which page(s) to remove
 - When to remove the page(s)
- Goals
 - Identify page to remove that will avoid future page faults (i.e. utilize locality as much as possible)
 - Minimize the amount of software and hardware overhead

Page replacement algorithms

- FIFO: Replace the oldest page
- LRU: Replace page that was the least recently used (longest since last use)

FIFO v.s. LRU

- Assume your OS uses FIFO policy when handle page faults. Also assume that we have 3 physical memory pages available. Compared with the same machine using an OS with LRU page replacement police, how many more page faults will you see for the FIFO based OS in the following page reference sequence?

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|
| Page # | 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |

- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

FIFO v.s. LRU

- Assume your OS uses FIFO policy when handle page faults. Also assume that we have 3 physical memory pages available. Compared with the same machine using an OS with LRU page replacement police, how many more page faults will you see for the FIFO based OS in the following page reference sequence?

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|
| Page # | 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |

- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

FIFO v.s. LRU

- Assume your OS uses FIFO policy when handle page faults. Also assume that we have 3 physical memory pages available. Compared with the same machine using an OS with LRU page replacement police, how many more page faults will you see for the FIFO based OS in the following page reference sequence?

A. 0

B. 1

C. 2

D. 3

E. 4

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|
| Page # | 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |
| FIFO | 2 | 2 | 2 | 2 | 5 | 5 | 5 | 5 | 3 | 3 | 3 | 3 |
| | | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 5 | 5 |
| | | | | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 | 2 |
| LRU | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
| | | 3 | 3 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| | | | | 1 | 1 | 1 | 4 | 4 | 4 | 2 | 2 | 2 |

FIFO v.s. LRU

| | FIFO | LRU |
|--------------------|----------------------------|---|
| Implementation | Easy — circular queue | May require hardware support or linked list or additional timestamps in page tables |
| Execution overhead | Low | High — you need to manipulate the list or update every counter |
| Performance | Usually not as good as LRU | Usually better than FIFO |

Converting a Swap-Based System to do Paging in an Architecture Lacking Page-Reference Bits

Özalp Babaoglu and William Joy*

Cornell University and University of California, Berkeley

The VMS/Old UNIX VM

- Regarding the original UNIX VM (basically the VMS), please identify how many of the following statements are correct.
 - ① VAX machine provides no hardware support for page replacement policies
 - ② VMS implements FIFO policy for page replacement
 - ③ A process's resident set cannot be adjusted even though that process is the only process in the system
 - ④ VMS swaps out all memory page belong to a process when that process is switched out
- A. 0
B. 1
C. 2
D. 3
E. 4

The VMS/Old UNIX VM

- Regarding the original UNIX VM (basically the VMS), please identify how many of the following statements are correct.
 - ① VAX machine provides no hardware support for page replacement policies
 - ② VMS implements FIFO policy for page replacement
 - ③ A process's resident set cannot be adjusted even though that process is the only process in the system
 - ④ VMS swaps out all memory page belong to a process when that process is switched out
- A. 0
B. 1
C. 2
D. 3
E. 4

The VMS/Old UNIX VM

- Regarding the original UNIX VM (basically the VMS), please identify how many of the following statements are correct.
 - ① VAX machine provides no hardware support for page replacement policies
 - ② VMS implements FIFO policy for page replacement
 - ③ A process's resident set cannot be adjusted even though that process is the only process in the system
 - ④ VMS swaps out all memory page belong to a process when that process is switched out

— Really inefficient if you have frequent context switches or if you have many applications in-fly

Whenever a process is removed from memory, its entire resident set is written to the swap file, along with some

A. 0

B. 1

C. 2

D. 3

E. 4

tions) base their decisions on. Without even this minimal page reference information, the only reasonable algorithms for replacing pages are the First-In-First-Out (FIFO) and the Random (RAND)

In VMS, the vendor-supplied operating system for the VAX, the solution to the replacement decision is simple. Each process is assigned a fixed-size memory partition, called a resident set, that is managed according to the FIFO policy. Pages

The Why of Babaoglu new UNIX VM

- The original UNIX is a **swap-based** system
 - Whenever you have a context switch, swap the whole process out from the memory
 - Really inefficient if you have frequent context switches or if you have many applications in-fly
 - Imply that the **modern** UNIX or Linux **does not** do this
- Efficient page replacement policies and other virtual optimization techniques cannot be implemented easily without appropriate hardware support

The page replacement policy proposed

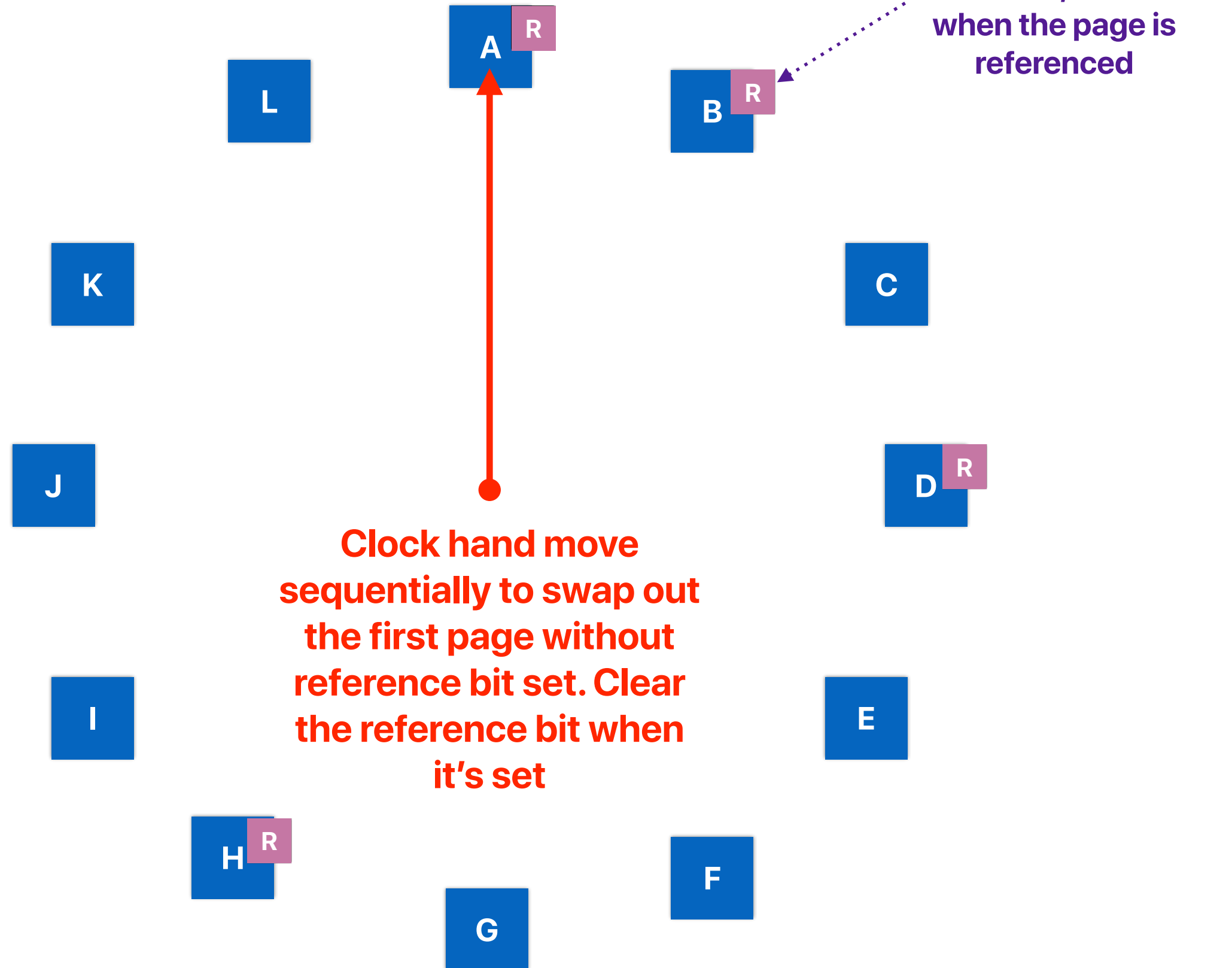
- How many of following statements fit the page replacement policy that the paper implements?
 - ① It uses LRU (least-recently-used) as the page replacement policy
 - ② Page replacement policy are only triggered whenever a page fault occurs
 - ③ It attaches a timestamp to each page table entry instead of using the reference bit from hardware
 - ④ Processes are allocated a fixed set of pages and swap in/out to/from those pages
 - ⑤ The page replacement policy helps to guarantee the response time of short programs
- A. 0
B. 1
C. 2
D. 3
E. 4

The page replacement policy proposed

- How many of following statements fit the page replacement policy that the paper implements?
 - ① It uses LRU (least-recently-used) as the page replacement policy
 - ② Page replacement policy are only triggered whenever a page fault occurs
 - ③ It attaches a timestamp to each page table entry instead of using the reference bit from hardware
 - ④ Processes are allocated a fixed set of pages and swap in/out to/from those pages
 - ⑤ The page replacement policy helps to guarantee the response time of short programs

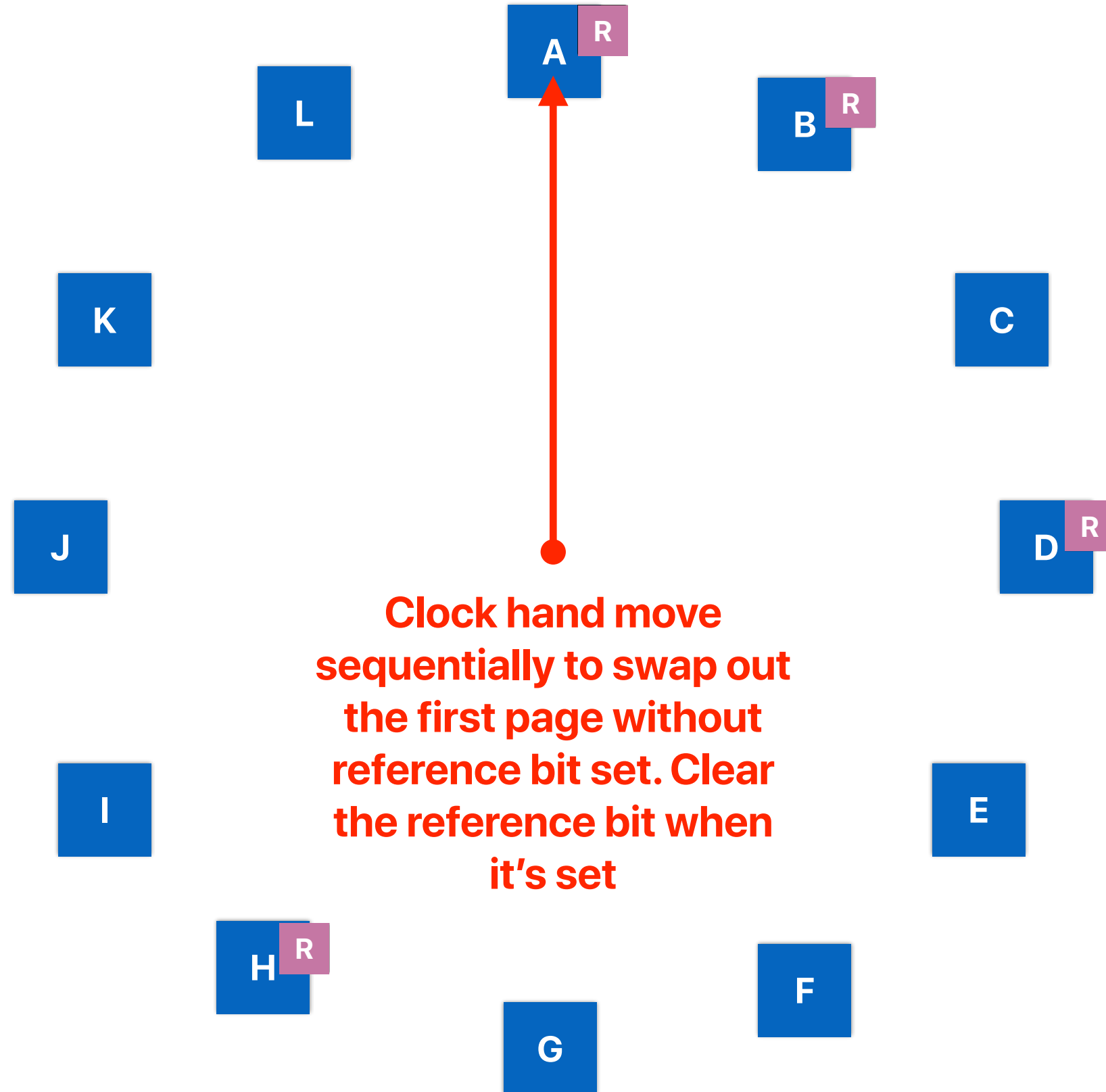
A. 0
B. 1
C. 2
D. 3
E. 4

Clock algorithm



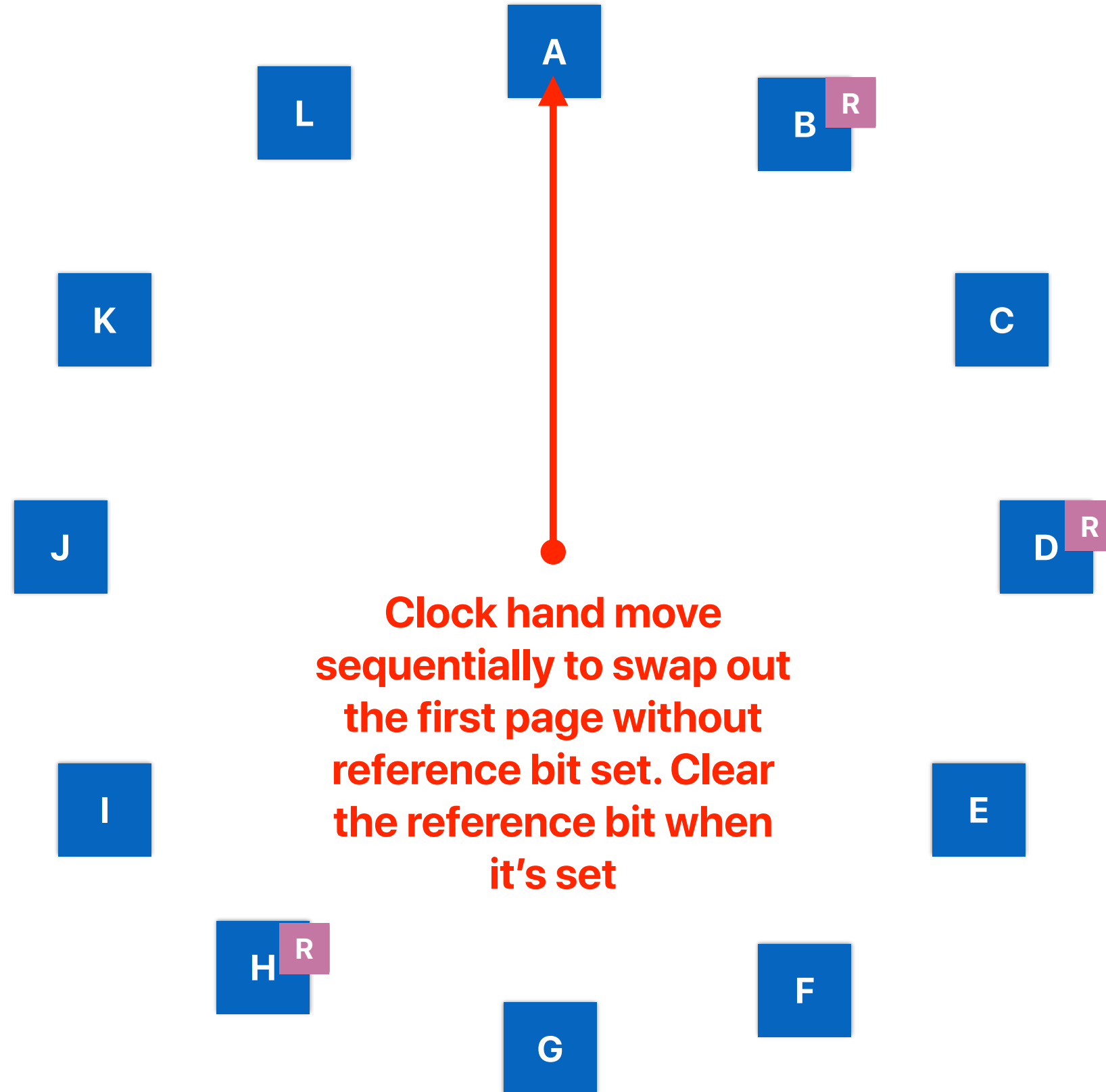
Clock algorithm in motion

Where to put **M** ?



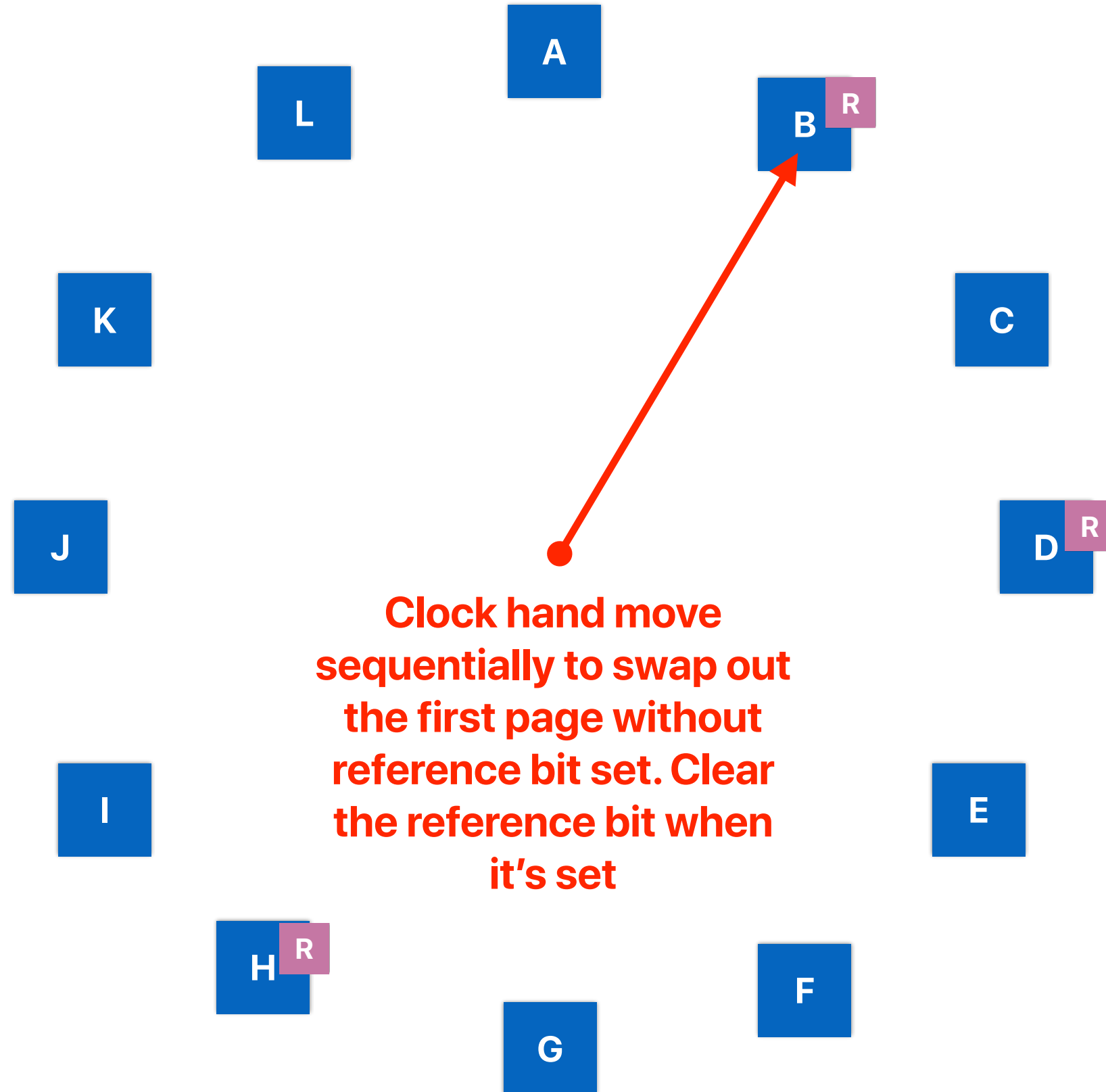
Clock algorithm in motion

Where to put **M** ?



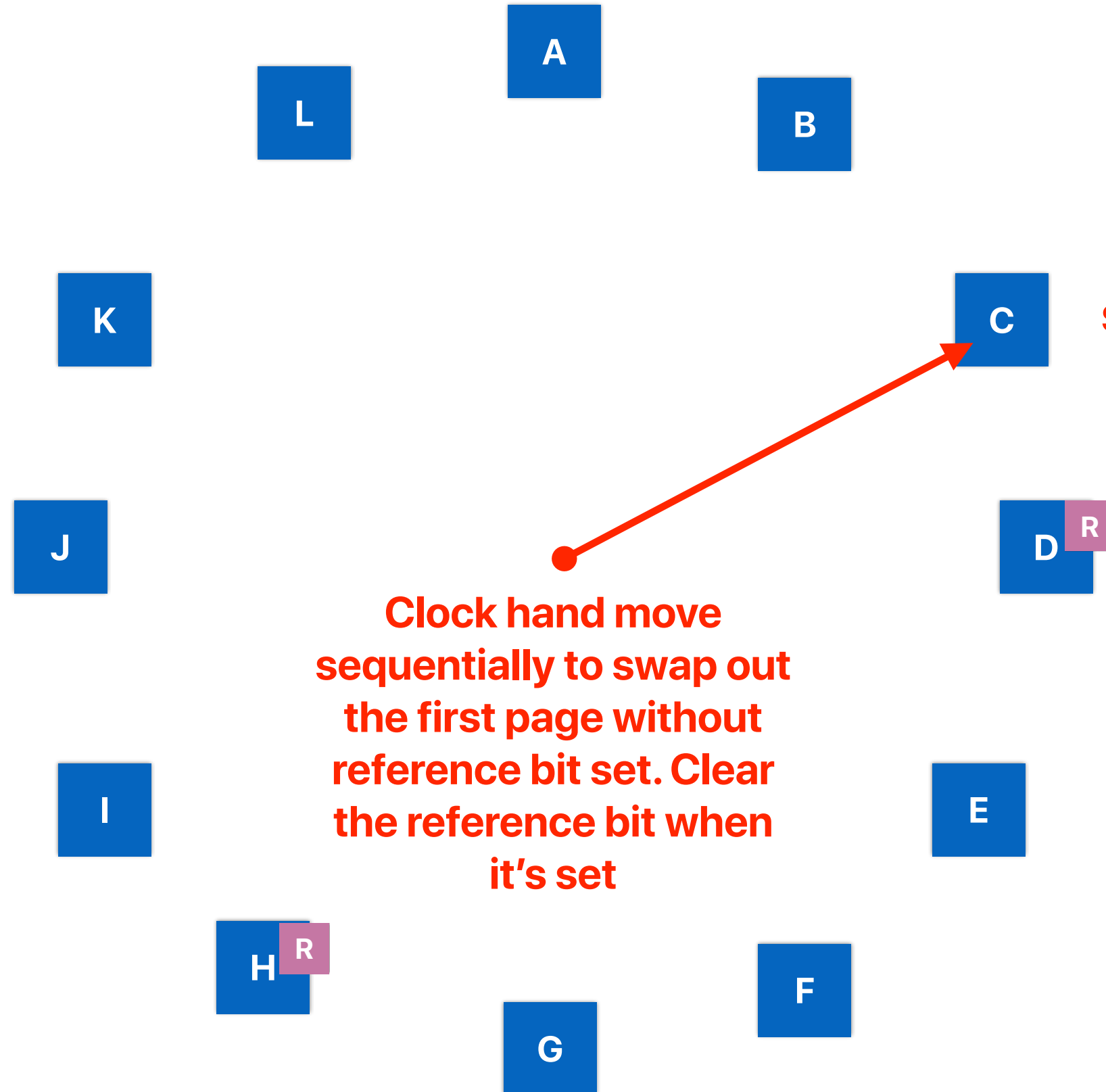
Clock algorithm in motion

Where to put **M** ?



Clock algorithm in motion

Where to put M ?



C will be selected to swap out, but Rs of A and B are cleared

Clock hand move sequentially to swap out the first page without reference bit set. Clear the reference bit when it's set

Recap: LRU

- Assume your OS uses LRU policy when handle page faults. Also assume that we have 3 physical memory pages available. How many page faults will you see in the following page reference sequence?

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|
| Page # | 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |

A. 5

B. 6

C. 7

D. 8

E. 9

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
| | 3 | 3 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| | | | 1 | 1 | 1 | 4 | 4 | 4 | 2 | 2 | 2 |

How good is clock?

- Assume your OS uses the clock policy when handle page faults. Also assume that we have 3 physical memory pages available. How many page faults will you see in the following page reference sequence?

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|
| Page # | 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |

- A. 5
- B. 6
- C. 7
- D. 8
- E. 9

How good is clock?

- Assume your OS uses the clock policy when handle page faults. Also assume that we have 3 physical memory pages available. How many page faults will you see in the following page reference sequence?

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|
| Page # | 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |

- A. 5
- B. 6
- C. 7
- D. 8
- E. 9

How good is clock?

- Assume your OS uses the clock policy when handle page faults. Also assume that we have 3 physical memory pages available. How many page faults will you see in the following page reference sequence?

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|
| Page # | 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |

A. 5

B. 6

C. 7

D. 8

E. 9

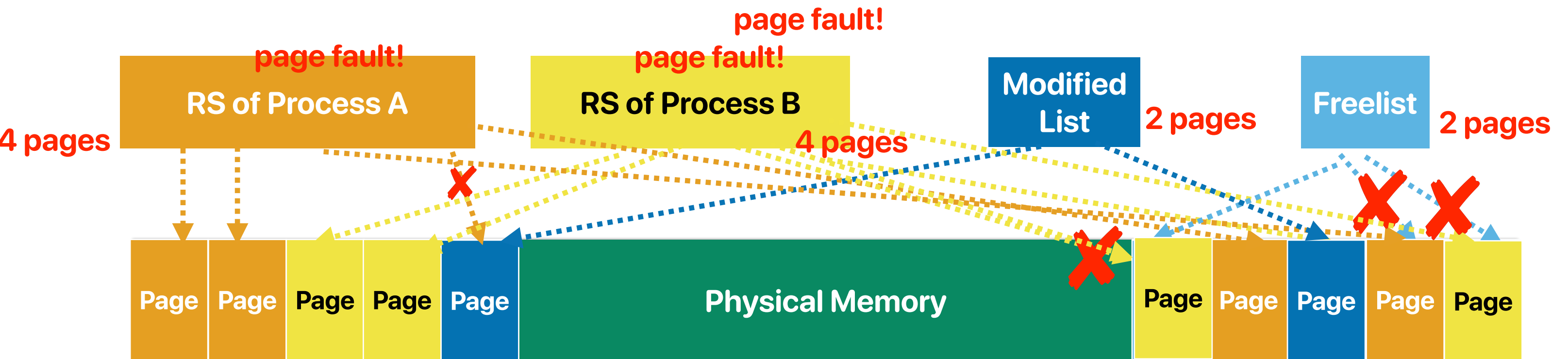
| | | | | | | | | | | | |
|---|----|-----|-----|----|----|----|-----|----|-----|----|-----|
| 2 | 2* | 2*+ | 2*+ | 2 | 2+ | 2+ | 2*+ | 2* | 2*+ | 2* | 2*+ |
| | 3 | 3 | 3 | 5 | 5 | 5 | 5+ | 5 | 5 | 5+ | 5+ |
| | | | 1 | 1* | 1* | 4 | 4 | 3 | 3 | 3 | 3 |

+ means the reference bit is set

* means the current hand

Recap: Page caching to cover the performance loss

- Evicted pages will be put into one of the lists in DRAM
 - Free list: clean pages
 - Modified list: dirty pages — needs to copy data to the disk
- Page fault to any of the page in the lists will bring the page back
 - Reduces the demand of accessing disks



Free list in Babaoglu's UNIX

- How many of the following statements regarding the "free list" is/are correct?
 - ① It can improve the latency of a page fault
 - ② It can reduce the latency of swapping out a page
 - ③ It can incur disk accesses without page faults
 - ④ It doesn't allow a page in the list to be used for other purpose

A. 0

B. 1

C. 2

D. 3

E. 4

Free list in Babaoglu's UNIX

- How many of the following statements regarding the "free list" is/are correct?
 - ① It can improve the latency of a page fault
 - ② It can reduce the latency of swapping out a page
 - ③ It can incur disk accesses without page faults
 - ④ It doesn't allow a page in the list to be used for other purpose

A. 0

B. 1

C. 2

D. 3

E. 4

Free list in Babaoglu's UNIX

- How many of the following statements regarding the "free list" is/are correct?
 - ① It can improve the latency of a page fault
— instead of swapping a page during the page fault, just take one from the free list
 - ② It can reduce the latency of swapping out a page
— No! This completely depend on how fast your disk/storage is!
 - ③ It can incur disk accesses without page faults
— Do you remember how UNIX page replacement is triggered?
 - ④ It doesn't allow a page in the list to be used for other purpose
— No! You can use those pages as disk caches!
- A. 0
- B. 1
- C. 2**
- D. 3
- E. 4

Free list

- So far, we need to trigger clock policy and swap in/out on each page fault
- Why don't we prepare more free pages each time so that we can feed page faults with pages from the list?
- Free list
 - When we need a page, take one from the free list
 - Have a daemon running the background, managing this free list — you can do this when system is not loaded
 - If size of free list gets too small, trigger the clock algorithm to add pages into the free list (by swapping out to disk)
 - Free list can be used as a disk cache

The page replacement policy proposed

- How many of following statements fit the page replacement policy that the paper implements?
 - ① It uses LRU (~~least recently used~~) as the page replacement policy
clock
 - ② Page replacement policy are only triggered whenever a ~~page fault occurs~~
free list is under a threshold
 - ③ It attaches a ~~timestamp~~ to each page table entry instead of using the reference bit
reference bit from hardware
 - ④ Processes are allocated a fixed set of pages and swap in/out to/from those pages
 - ⑤ The page replacement policy helps to guarantee the response time of short programs
- A. 0
B. 1
C. 2
D. 3
E. 4

The page replacement policy proposed

- How many of following statements fit the page replacement policy that the paper implements?
 - ① It uses LRU (~~least recently used~~) as the page replacement policy
clock
 - ② Page replacement policy are only triggered whenever a ~~page fault occurs~~
free list is under a threshold
 - ③ It attaches a ~~timestamp~~ to each page table entry instead of using the reference bit from hardware
reference bit
 - ④ ~~Processes are allocated a fixed set of pages and swap in/out to/from those pages~~
Process just get a page from the free list whenever it needs
 - ⑤ The page replacement policy helps to guarantee the response time of short programs

A. 0

B. 1

C. 2

D. 3

E. 4

formance implications. Lazowska [LAZO 79] reports that in his measurements based on a real workload, system performance was significantly improved by increasing the minimum size of the free list (a system generation parameter). An unfortunate

(ii) The projected workload for the system had no requirement of guaranteed response times as in real-time applications.

WSClock - A Simple and Effective Algorithm for Virtual Memory Management

Richard Carr and John Hennessy

Brief recap: what policies are used?

- Local: select one page from the same process' physical pages for storing the demanding page when swapping is necessary
 - VAX/VMS
 - Original UNIX
- Global: select any page that was previously belong to any process when swapping is necessary
 - UNIX after Babaoglu
 - Mach

Degree of parallelism and performance

`accomplish most tasks-- processes are cheap. These`

- How many of the following would happen in Babaoglu's UNIX VM if we keep increase the amount of concurrent processes and assume each process uses some virtual memory in the system?
 - ① The CPU utilization will keep increasing and stay at 100%
 - ② The system may spend more time in context switching than real computation
 - ③ The system may spend more time in swap in/out than real computation
 - ④ Some process may not respond due to the high paging overhead
- A. 0
B. 1
C. 2
D. 3
E. 4

Degree of parallelism and performance

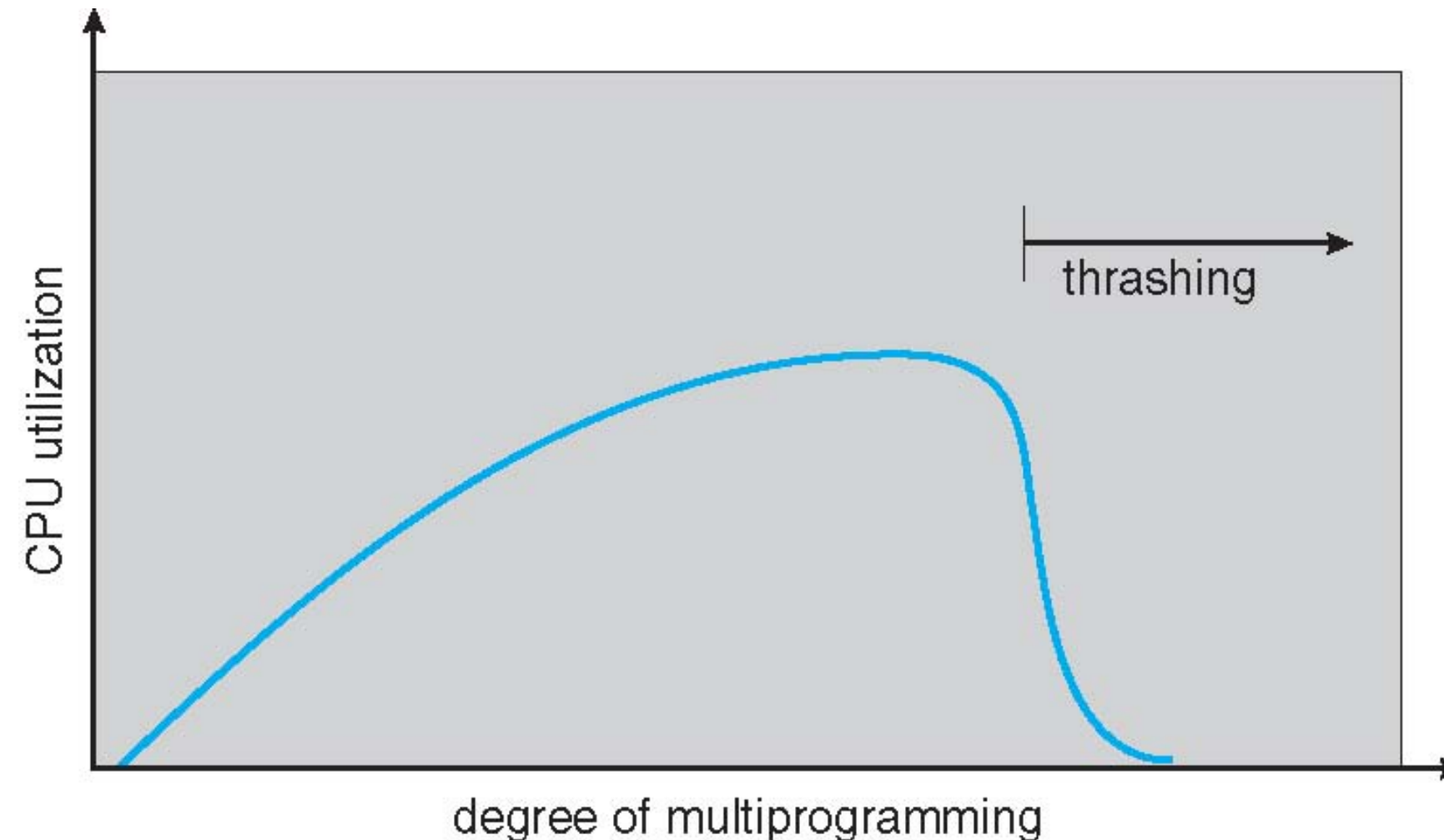
accomplish most tasks-- processes are cheap. These

- How many of the following would happen in Babaoglu's UNIX VM if we keep increase the amount of concurrent processes and assume each process uses some virtual memory in the system?
 - ① The CPU utilization will keep increasing and stay at 100%
 - ② The system may spend more time in context switching than real computation
 - ③ The system may spend more time in swap in/out than real computation
 - ④ Some process may not respond due to the high paging overhead
- A. 0
B. 1
C. 2
D. 3
E. 4

Thrashing: Paging overhead

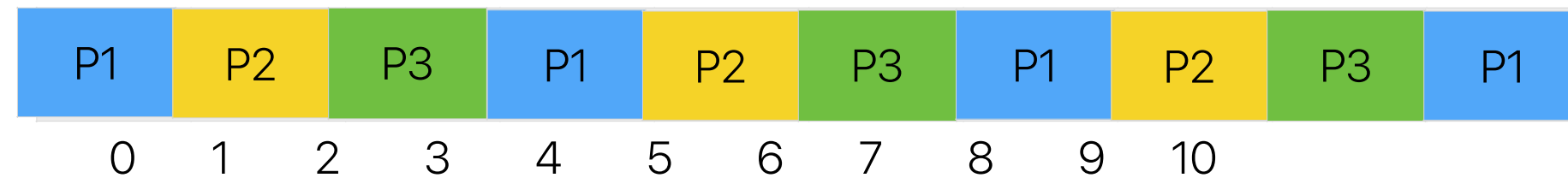
- The system overcommitted memory to tasks
- The system spends most time in paging, instead of making meaningful progress

**Previously, we have seen how scheduling policies can help improving "saturation".
Now, let's see how page replacement policies can address this "thrashing"**



Saturation: Context Switch Overhead

You think round robin should act like this —



But the fact is —



- Your processor utilization can be very low if you switch frequently
- No process can make sufficient amount of progress within a given period of time
- It also takes a while to reach your turn

Thrashing v.s. Saturation

- Thrashing — when memory are overcommitted
 - The system is busy paging
 - The processor is idle waiting
- Saturation — when processors are overcommitted
 - The system is busy context switching and scheduling
 - The processor is busy but not contributing to the running program

Degree of parallelism and performance

- How many of the following would happen in Babaoglu's UNIX VM if we keep increase the amount of concurrent processes and assume each process uses some virtual memory in the system?
 - ☒ ① The CPU utilization will keep increasing and stay at 100%
 - ② The system may spend more time in context switching than real computation
 - ③ The system may spend more time in swap in/out than real computation
 - ④ Some process may not respond due to the high paging overhead
- A. 0
B. 1
C. 2
D. 3
E. 4

Why WS-Clock

- Take advantages from both local and global page replacement policies
 - Global — simplicity, adaptive to process demands
 - Local — prevent thrashing

Working Set Algorithm

- Working set: the set of pages used in a certain number of recent accesses
- Assume these recently referenced pages are likely to be referenced again soon (temporal locality)
- Evict pages that are not referenced in a certain period of time
 - Swap out may occur even if there is no page faults
- A process is allowed to be executed only if the working set size fits in the physical memory

WSClock

- Use **working set** policy to decide how many pages can a process use
 - Return a page to the free list if there exists a page in the process' working set that hasn't been access for a certain period of time
- If the free list is lower than a threshold
 - Trigger the **clock** policy to select pages from any process
- On a page fault
 - Take a page from the free list

WSClock

- Wherever you need to reclaim a page —
 1. Examine the PTE pointed to by clock hand.
 2. If reference bit is set
 1. Clear reference bit;
 2. Advance clock hand;
 3. Goto Done.
 3. If reference bit is not set
 1. If the timestamp of the PTE is older than a threshold
 1. Write the page to disk if it's dirty and use this page
 2. Goto Done
 2. Otherwise
 1. Advance clock hand
 2. Goto 1.
 4. Done
 5. If no victim page is chosen, randomly pick one

The impact of WSClock

- One of the most important page replacement policies in practice

Sample Midterm

Disclaimer

- This is just a sample midterm for you to practice.
- Questions listed in multiple choices can be transformed as free answer or short answer questions in the midterm.
 - Same thing for sample free answer questions and short answer questions

Regarding the "true" midterm

- Rules
 - Cheating is not allowed and you will receive an F once we identified that
 - Will release on 2/10/2021 0:00am and due on 2/15/2021 11:59:00pm
 - You will have to find a consecutive, non-stop 80-minute slot with this period
 - **One time, cannot reinitiate** — please make sure you have a stable system and network
 - **No late submission is allowed**
- Format
 - 15 multiple choices — 30%
 - 4 short answer questions — 38%
— You have to explain everything within 30 words.
 - 3 free answer questions — 32%

What OS must track for a process?

- Which of the following information does NOT the OS need to track?
 - A. Stack pointer
 - B. Program counter
 - C. Scheduling information
 - D. Registers
 - E. None of the above

What do we need for parallel processing

- Which process component(s) must we replicate in order to take advantage of multiple cores/CPU's?
 - A. The address space (i.e. memory)
 - B. Misc. resources (e.g. open files)
 - C. Execution state (e.g. PC, registers, stack pointer)
 - D. More than one of the above
 - E. None of the above (explain).

Is hierarchical design a good idea?

- How many the following is/are true regarding the proposed hierarchical design in Dijkstra's THE.
 - ① Hierarchical design facilitates debugging
 - ② Hierarchical design makes verification of system components easier
 - ③ Hierarchical design reduces the overhead of running a single process
 - ④ The proposed hierarchical design allows layer 0 to schedule I/O & peripherals
- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

Rejection of layering

- What's the main reason why HYDRA rejects the layering proposed in Dijkstra's THE?
 - A. Enhance functionality
 - B. Support concurrency
 - C. Facilitate debugging
 - D. Improve flexibility
 - E. Boost performance

Understanding about HYDRA

- Regarding HYDRA, how many of the following statements is/are correct?
 - ① The procedure call overhead of using capability is larger than without using capability
 - ② A procedure can have a different capability from the caller
 - ③ If the caller does not have the permission of an operation on an object, the caller may still perform that operation by calling a procedure
 - ④ Only the kernel can alter capabilities
- A. 0
B. 1
C. 2
D. 3
E. 4

Processes and threads

- Regarding processes (or say tasks in Mach) and threads, how many of the following statements is/are correct?
 - ① Threads within a process share the same address space
 - ② Threads within a process can communicate and synchronize using shared memory
 - ③ Processes can communicate through messages
 - ④ Two processes may only be able to communicate through messages

A. 0
B. 1
C. 2
D. 3
E. 4

Protection

- Regarding the protection in UNIX, how many of the followings is/are correct?
 - ① The same file may have different permissions for different user-id
 - ② The owner of the file may not have the permission of writing a file
 - ③ If the user does not have a permission to access a device, set-user-id will guarantee that the user will not be able to access that device
 - ④ In the UNIX system described in this paper, if the file owner is "foo", then the user "bar" will have the same permission as another user (e.g. "xyz").
- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

Why not microkernels?

- Although Mach's design strongly influenced modern operating systems, why most modern operating systems do not adopt the design of microkernels?
 - A. Microkernels are more difficult to extend than monolithic kernels
 - B. Microkernels are more difficult to maintain than monolithic kernels
 - C. Microkernels are less stable than monolithic kernels
 - D. Microkernels are not as competitive as monolithic kernels in terms of application performance
 - E. Microkernels are less flexible than monolithic kernels

What we learned about kernel

- Which of the following is true about kernel?
 - A. It executes as a process
 - B. It is always executing, in support of other processes
 - C. It should execute as little as possible.
 - D. A & B
 - E. B & C

Interrupt and Trap

- How many of the following statements is/are true regarding interrupt and trap?
 - ① Both interrupt and trap can incur context switch
 - ② Both interrupt and trap are raised from hardware
 - ③ Both interrupt and trap require OS kernel to handle
 - ④ Both interrupt and trap are machine dependent features
- A. 0
B. 1
C. 2
D. 3
E. 4

Did they achieve their goals?

- Considering the optimization goals and the proposed VAX/VMS mechanisms, which of the following combinations is incorrect?

| Goal | | Optimization | |
|------|----------------------------------|--------------|---------------------------------|
| A | Process startup cost | W | Demand-zero & copy-on-reference |
| B | Process performance interference | X | Process-local replacement |
| C | Page table lookup overhead | Y | Page clustering |
| D | Paging load on disks | Z | Page caching |

The role of the OS in virtual memory management

- How many of the following tasks in virtual memory management always requires the assistance of operating system?

- ① Address translation
- ② Growth of process address space
- ③ Tracking free physical memory locations
- ④ Maintaining mapping tables

A. 0

B. 1

C. 2

D. 3

E. 4

Address translation in x86-64

| 63:48 (16 bits) | 47:39 (9 bits) | 38:30 (9 bits) | 29:21 (9 bits) | 20:12 (9 bits) | 11:0 (12 bits) |
|-----------------|----------------|----------------|----------------|----------------|----------------|
| SignExt | L4 index | L3 index | L2 index | L1 index | page offset |

- The above shows the address partition in x86-64. According to this information, how many of the following is/are true?
 - ① x86-64 provides 16EB virtual memory space
 - ② each node in the hierarchical page contains 512 entries
 - ③ the default page size is 4KB
 - ④ if only three level indexes are used, x86-64 can support 2MB page size
- A. 0
B. 1
C. 2
D. 3
E. 4

Free list

- How many of the following statements regarding the “free list” is/are correct?
 - ① It can improve the latency of a page fault
 - ② It can reduce the latency of swapping out a page
 - ③ It can incur disk accesses without page faults
 - ④ It doesn't allow a page in the list to be used for other purpose

A. 0

B. 1

C. 2

D. 3

E. 4

User-level v.s kernel threads

- Comparing user-level threads and kernel threads, please identify how many of the following statements are correct.
 - ① The overhead of switching threads is smaller for user-level threads
 - ② The OS scheduler can directly control the scheduling of kernel thread, but not for user-level threads
 - ③ A user-level thread can potentially block all other threads in the same process
 - ④ Implementing the user-level thread library can be achieved without modifying the OS kernel
- A. 0
B. 1
C. 2
D. 3
E. 4

Answer the following within 30 words

- Differences among 4 types of kernels
- Differences between threads and processes
- The pros and cons between using threads and processes to parallelize an application
- Differences between global and local page replacement policy. Examples of using global/local page replacement policies?
- What is "free list"? How is it used in the system?
- Differences among VAX/VMS and Mach's VM. Why initial version of UNIX's VM resembles VAX/VMS?
- Differences between UNIX in the 70's paper and now. Why?
- What is segmentation fault? What is page fault?
- What is thrashing? What's saturation?

Think deep

- What problem does multi-level scheduling policy try to address? What they proposed to address the issue?
- What will happen during a context switch? How expensive is a context?

Page replacement policy

- Assume your OS uses LRU policy when handle page faults. Also assume that we have 3 physical memory pages available. How many page faults will you see in the following page reference sequence?

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|
| Page # | 9 | 4 | 8 | 7 | 9 | 4 | 8 | 7 | 9 | 4 | 8 | 7 |

- What if we use FIFO?
- What if we use Clock policy?
- Can you propose a policy outperform both FIFO and Clock?

Bounded-buffer/producer-consumer problem

- What functions to use for P, Q, R, S in this problem?
- What variables to use for W, X, Y, Z in this problem?
- Can you use spinlock to achieve the same effect?
- What if we have multiple producers and consumers?

```
int main(int argc, char *argv[]) {
    pthread_t p;
    printf("parent: begin\n");
    // init here
    Pthread_create(&p, NULL, child, NULL);
    int in = 0;
    Sem_init(&filled, 0);
    Sem_init(&empty, BUFF_SIZE);
    while(TRUE) {
        int item = ...;
        P(&W);
        buffer[in] = item;
        in = (in + 1) % BUFF_SIZE;
        Q(&X);
    }
    printf("parent: end\n");
    return 0;
}
```

```
int buffer[BUFF_SIZE]; // shared global
sem_t filled, empty;
```

```
void *child(void *arg) {
    int out = 0;
    printf("child\n");
    while(TRUE) {
        R(&Y);
        int item = buffer[out];
        out = (out + 1) % BUFF_SIZE;
        // do something w/ item
        S(&Z);
    }
    return NULL;
}
```

More forks

- Consider the following code

```
fork();  
printf("moo\n");  
fork();  
printf("oink\n");  
fork();  
printf("baa\n");
```

What is the output?

Programming questions in C

- How to implement a simple shell that can launch a user program given from the command line?
- How to implement a shell that can redirect program output to a file?

Announcement

- No lecture this Thursday — relocate those 80 minutes to anytime you like before EOD 2/10.
- Reading quizzes due next Thursday
- New office hour
 - M 3p-4p and Th 9a-10a
 - Use the office hour Zoom link, not the lecture one
- Project released
 - Groups in 2
 - **Pull the latest version — had some changes for later kernel versions**
<https://github.com/hungweitseng/CS202-ResourceContainer>
 - **Install an x86-64 Ubuntu Linux 16.04.07 (or later) VM as soon as you can!**

Computer
Science &
Engineering

202



2021 新年快樂

