Memory Management with Bitmaps

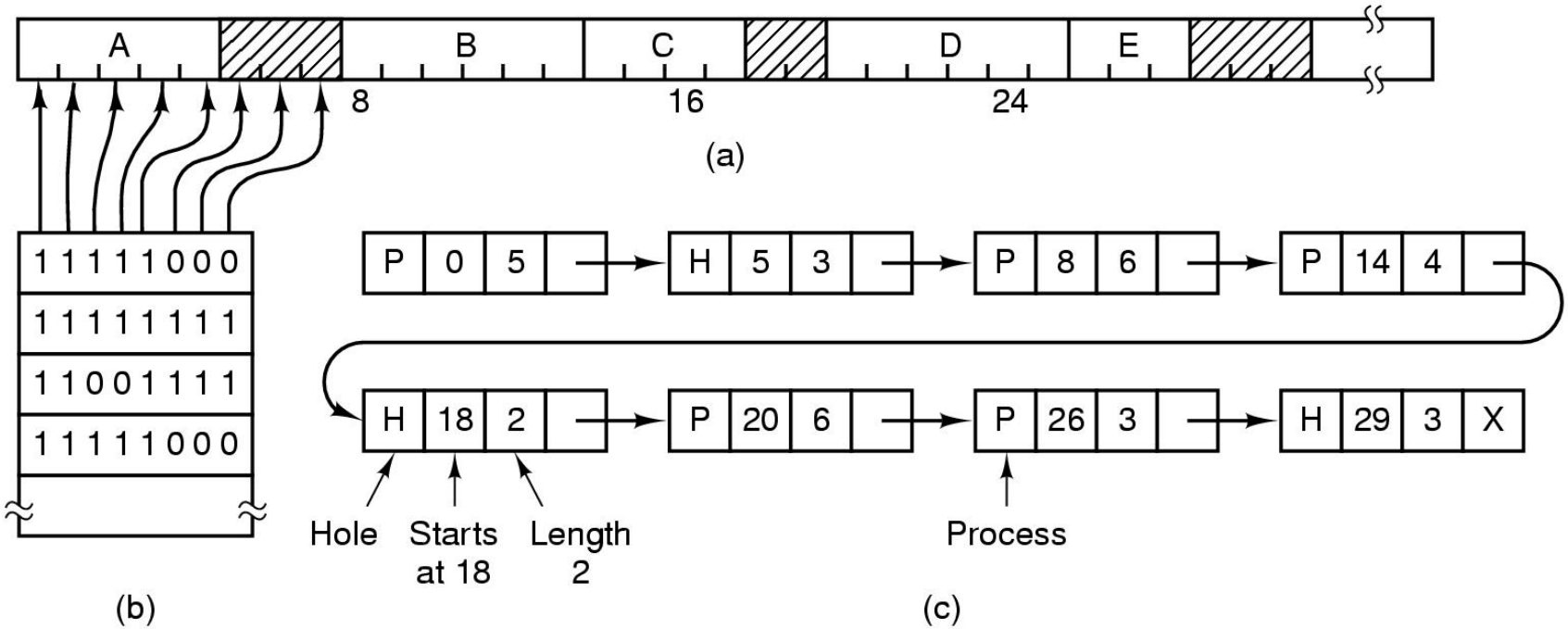


Figure 3-6. (a) A part of memory with five processes and three holes. The tick marks show the memory allocation units. The shaded regions (0 in the bitmap) are free. (b) The corresponding bitmap. (c) The same information as a list.

Memory Management with Linked Lists

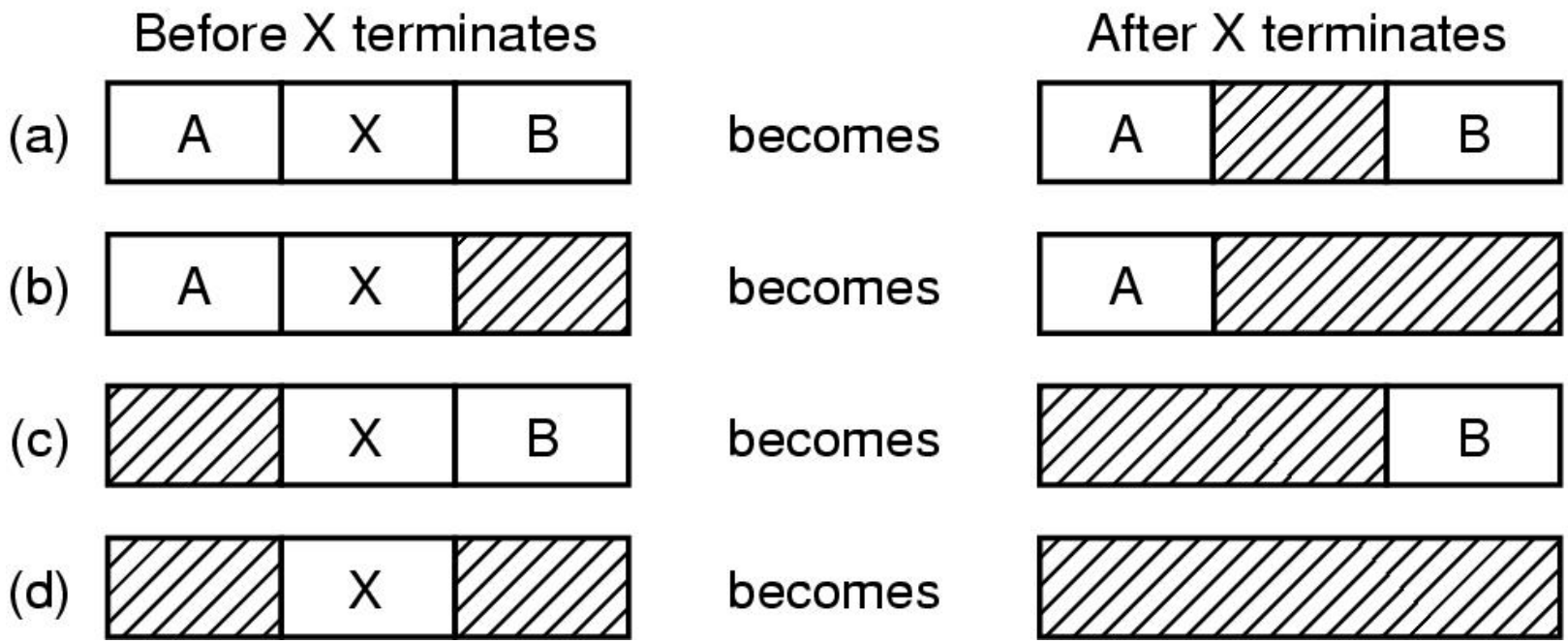


Figure 3-7. Four neighbor combinations for the terminating process, X.

Virtual Memory – Paging (1)

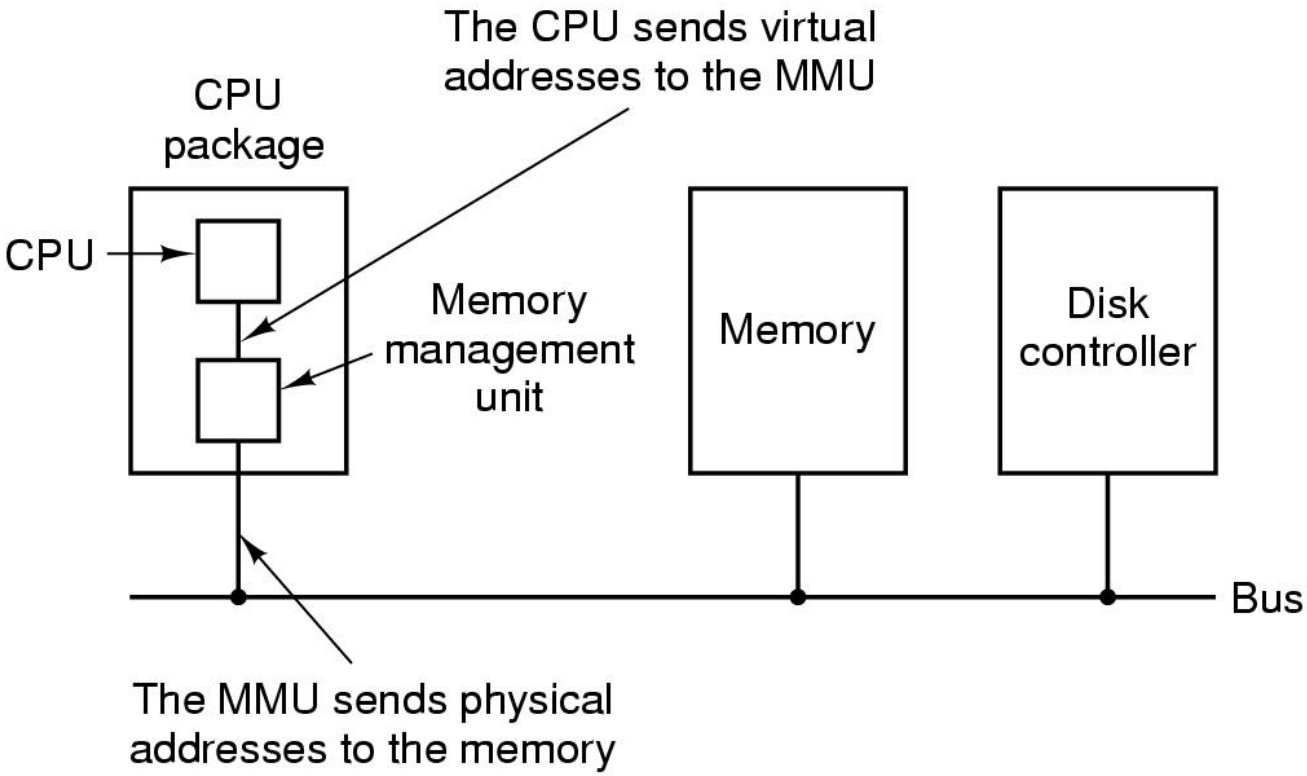


Figure 3-8. The position and function of the MMU – shown as being a part of the CPU chip (it commonly is nowadays). Logically it could be a separate chip, was in years gone by.

Paging (2)

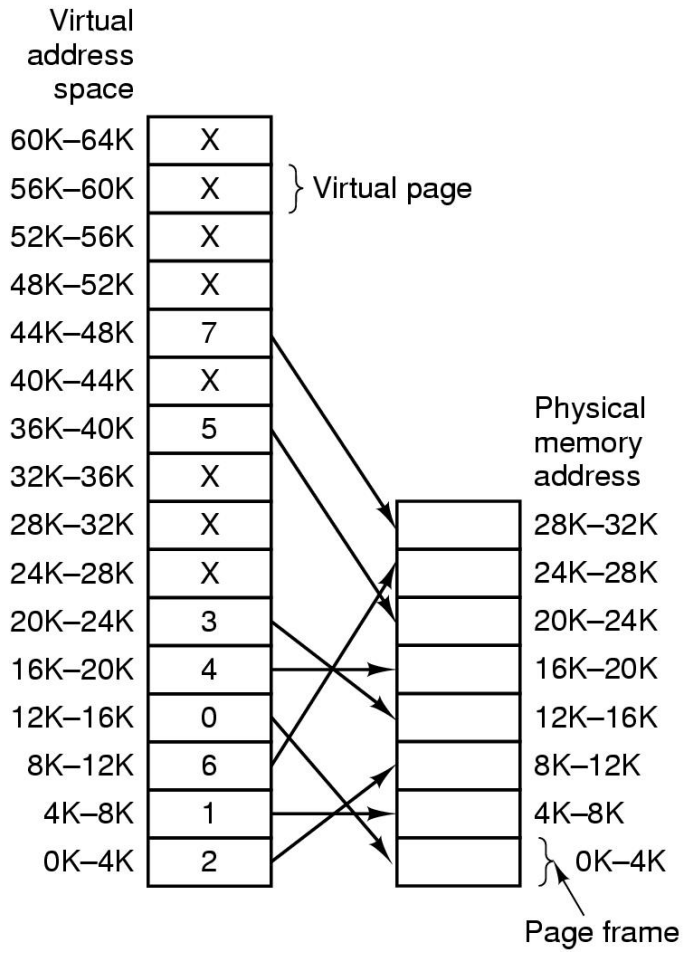


Figure 3-9. Relation between virtual addresses and physical memory addresses given by page table.

Paging (3)

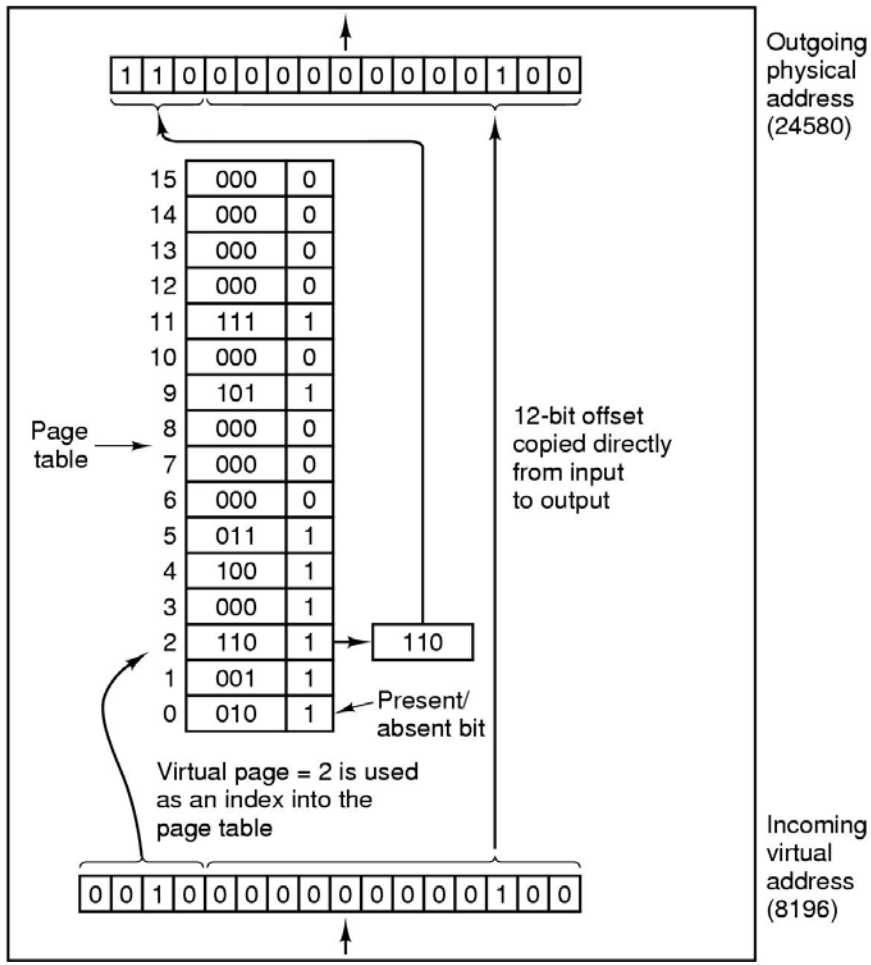


Figure 3-10. The internal operation of the MMU with 16 4-KB pages.

Structure of Page Table Entry

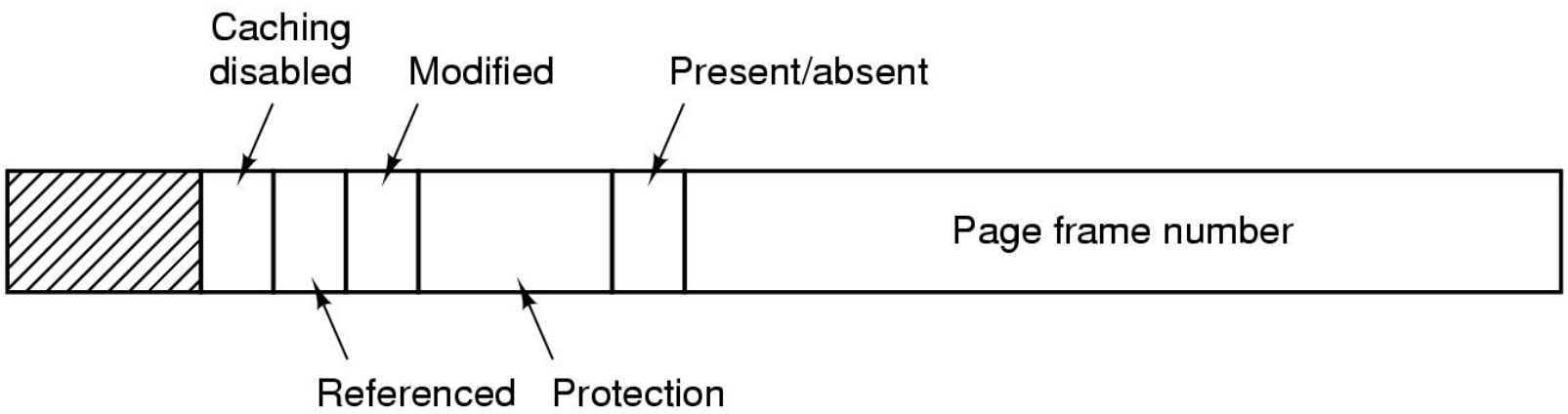
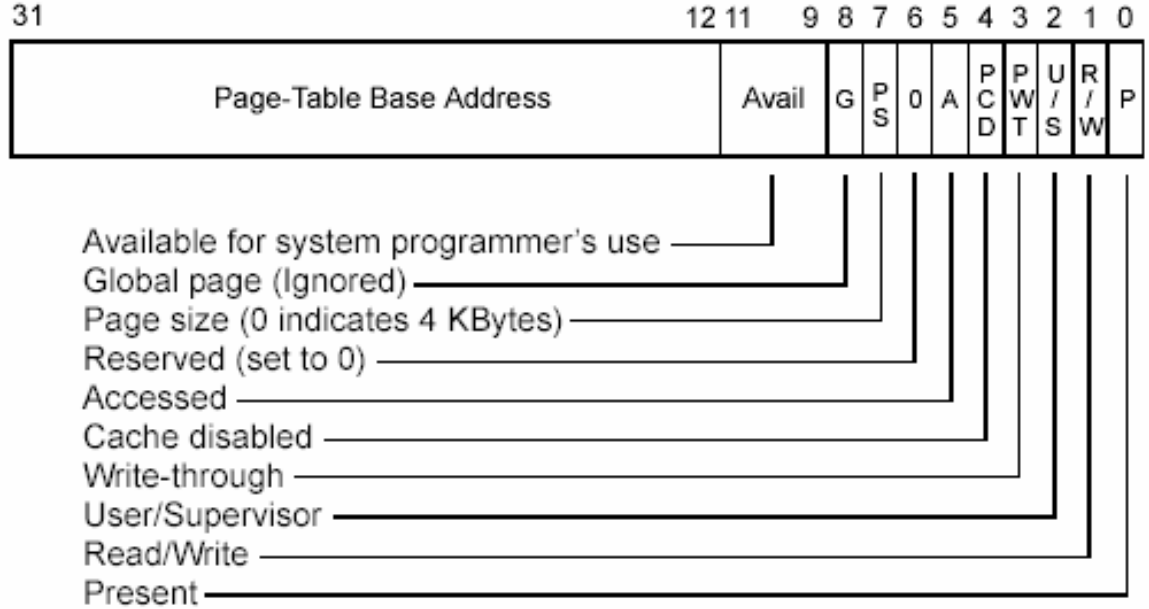
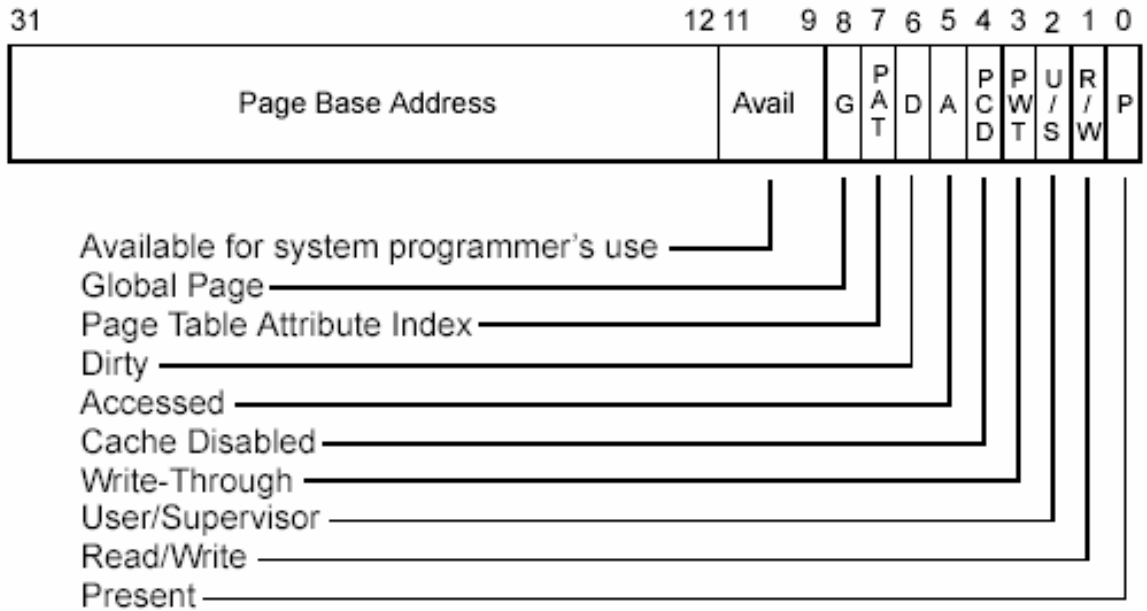


Figure 3-11. A typical page table entry.

Page-Directory Entry (4-KByte Page Table)



Page-Table Entry (4-KByte Page)



PTE Control Bits

- Present flag
  - If it is set, the referred-to page (or Page Table) is contained in main memory.
- Accessed flag
  - Set each time the paging unit addresses the corresponding page frame.
- Dirty flag
  - Applies only to the Page Table entries.
- Read/Write flag
  - Contains the access right (Read/Write or Read) of the page or of the Page Table
- User/Supervisor flag
  - Contains the privilege level required to access the page or Page Table
- PCD and PWT flags
  - Controls the way the page or Page Table is handled by the hardware cache
- Page Size flag
  - Applies only to Page Directory entries.
- Global flag
  - Applies only to Page Table entries.

Speeding Up Paging

Paging implementation issues:

- The mapping from virtual address to physical address must be fast.
- If the virtual address space is large, the page table will be large.

Translation Lookaside Buffers

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Figure 3-12. A TLB to speed up paging.

Multilevel Page Tables

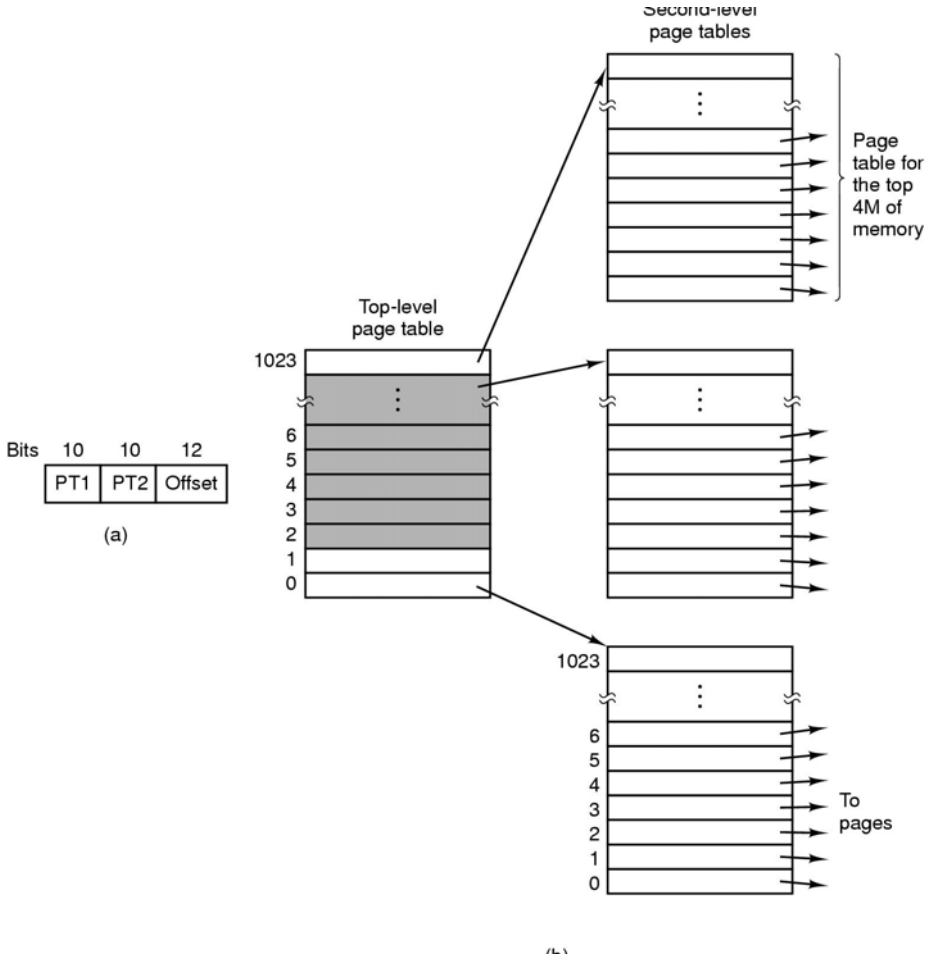


Figure 3-13. (a) A 32-bit address with two page table fields. (b) Two-level page tables.

Page Replacement Algorithms

- Optimal page replacement algorithm
- Not recently used page replacement
- First-In, First-Out page replacement
- Second chance page replacement
- Clock page replacement
- Least recently used page replacement
- Working set page replacement
- WSClock page replacement

Second Chance Algorithm

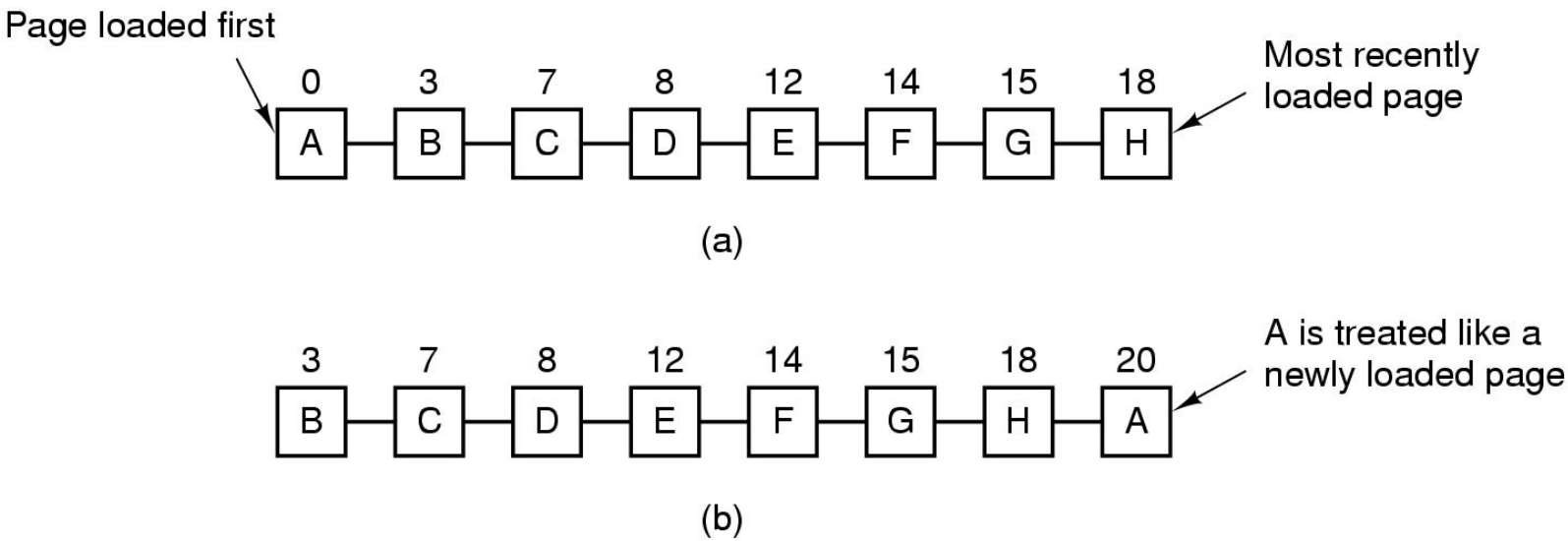


Figure 3-15. Operation of second chance.
(a) Pages sorted in FIFO order.
(b) Page list if a page fault occurs at time 20 and A has its R bit set. The numbers above the pages are their load times.

The Clock Page Replacement Algorithm

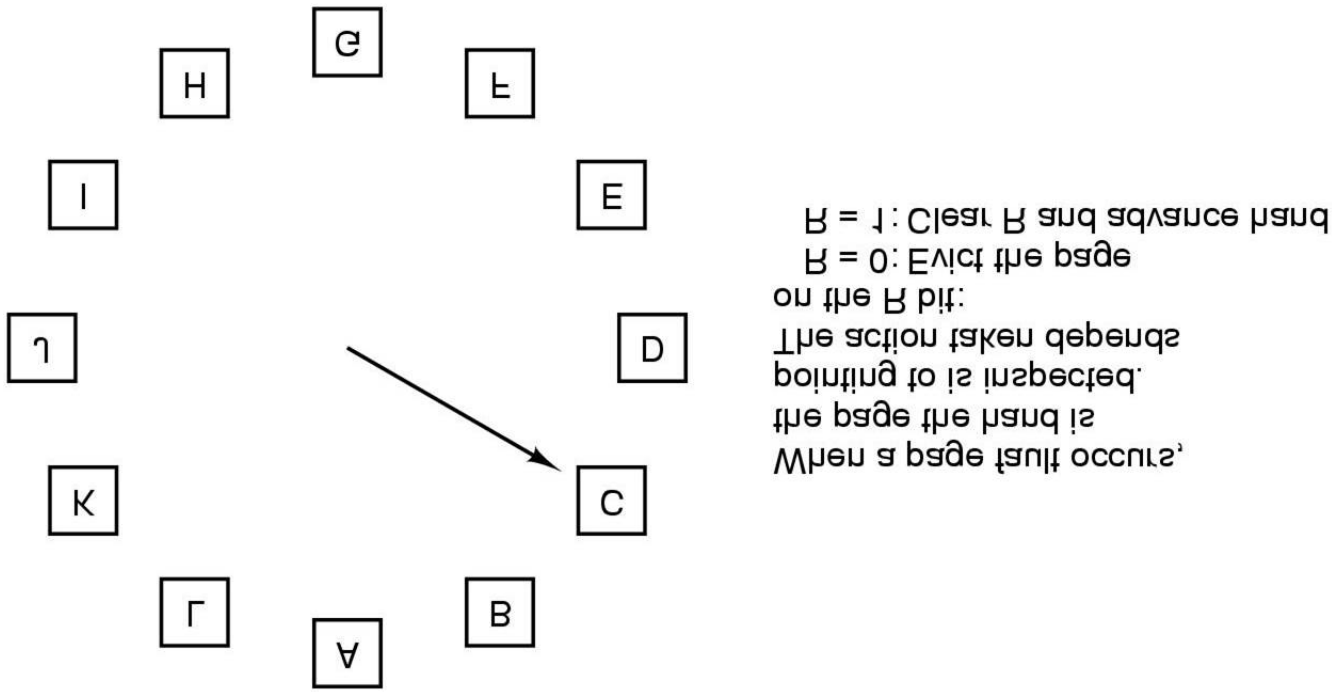


Figure 3-16. The clock page replacement algorithm.

LRU Page Replacement Algorithm

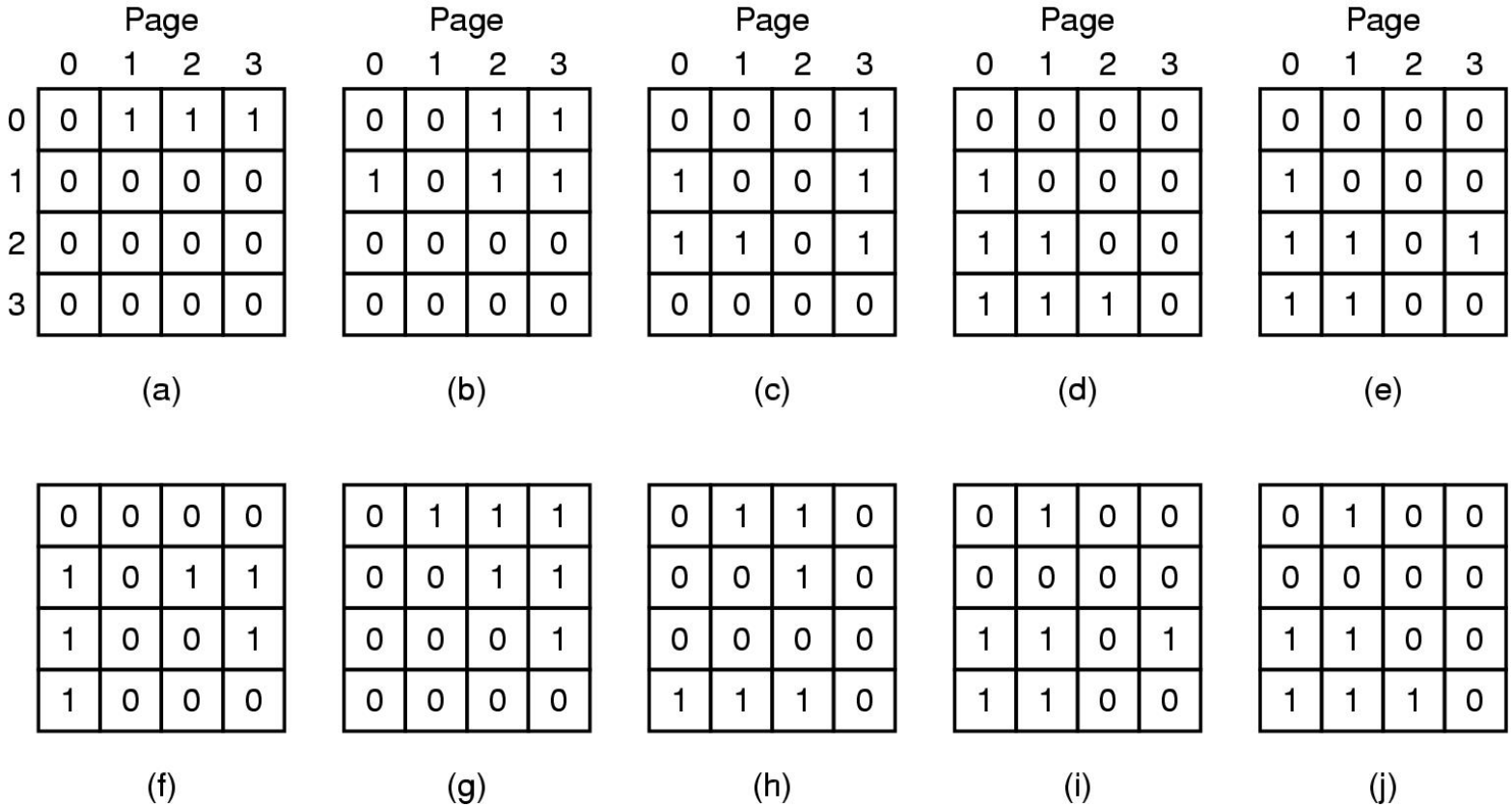


Figure 3-17. LRU using a matrix when pages are referenced in the order 0, 1, 2, 3, 2, 1, 0, 3, 2, 3.

Simulating LRU in Software

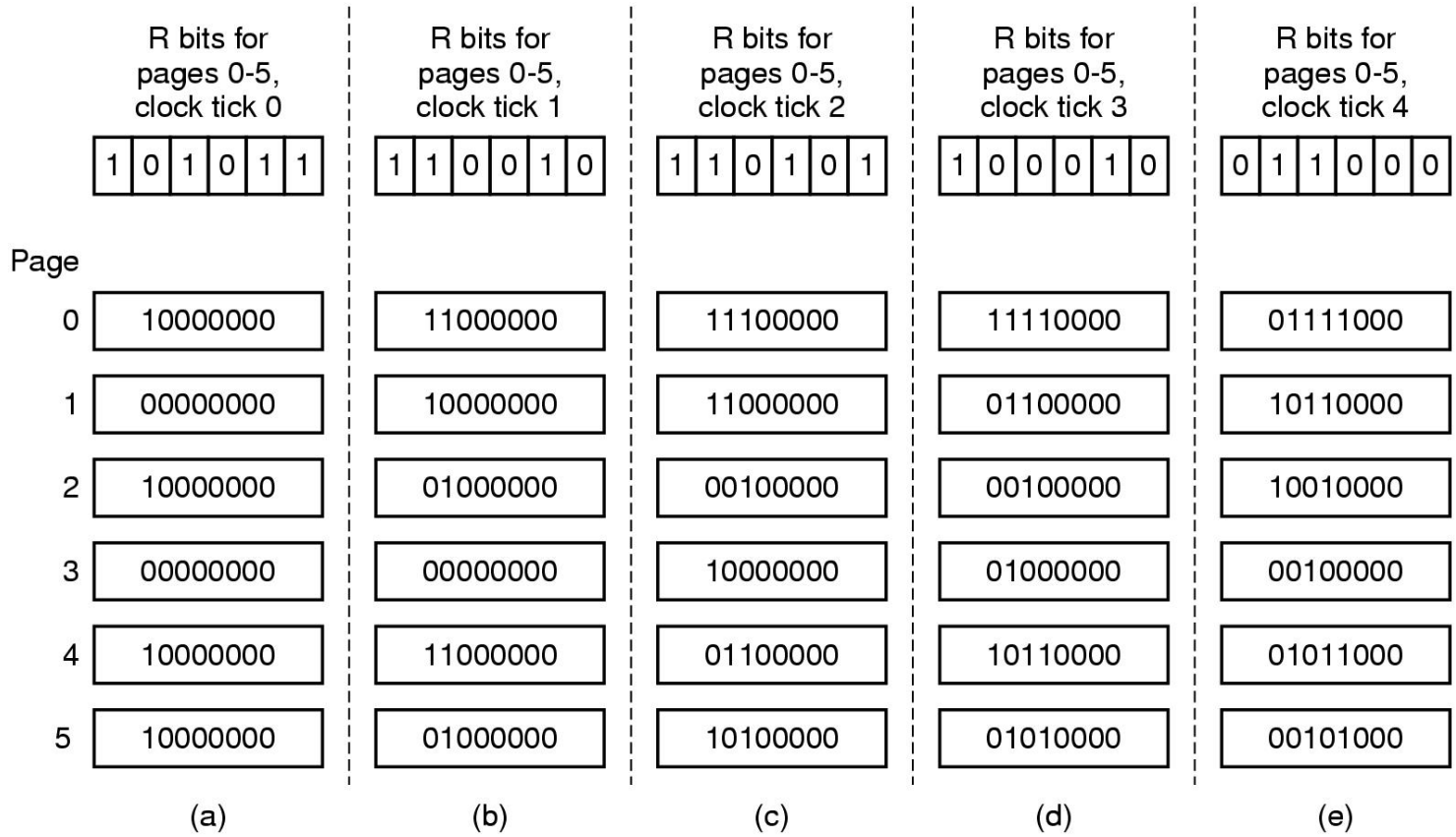


Figure 3-18. The aging algorithm simulates LRU in software. Shown are six pages for five clock ticks. The five clock ticks are represented by (a) to (e).

Working Set Page Replacement (1)

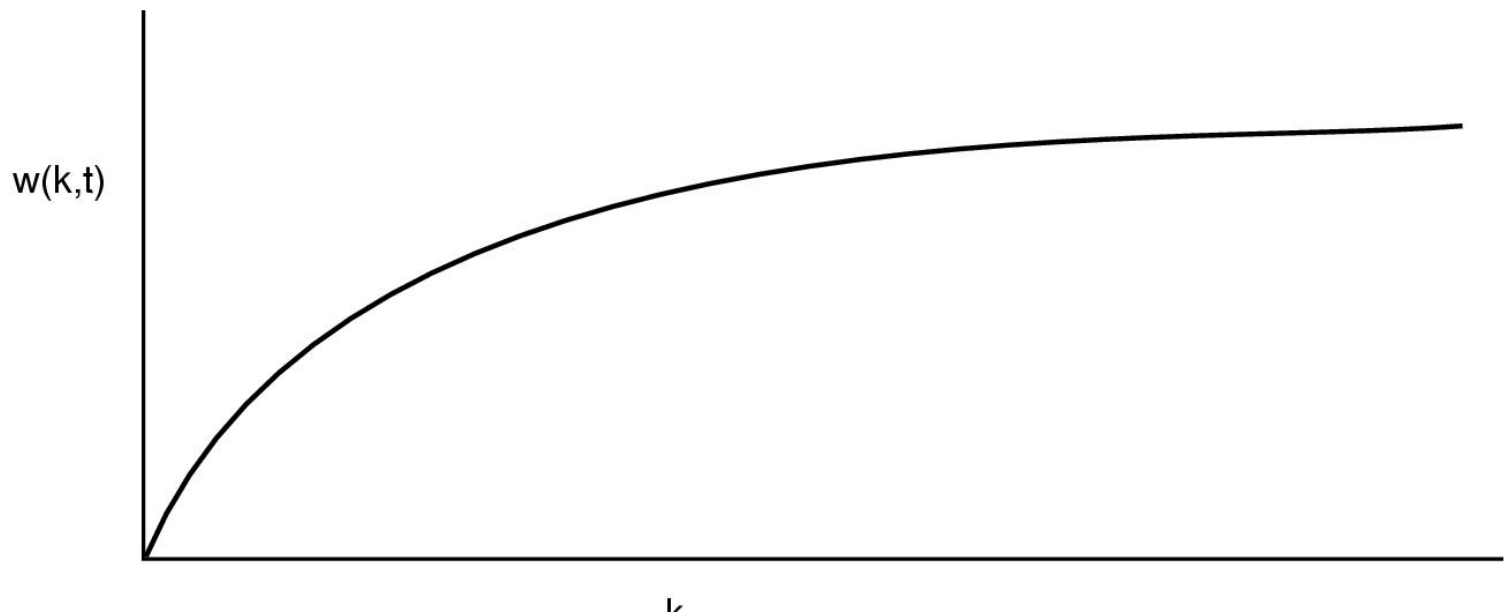


Figure 3-19. The working set is the set of pages used by the k most recent memory references. The function w(k, t) is the size of the working set at time t.

Working Set Page Replacement (2)

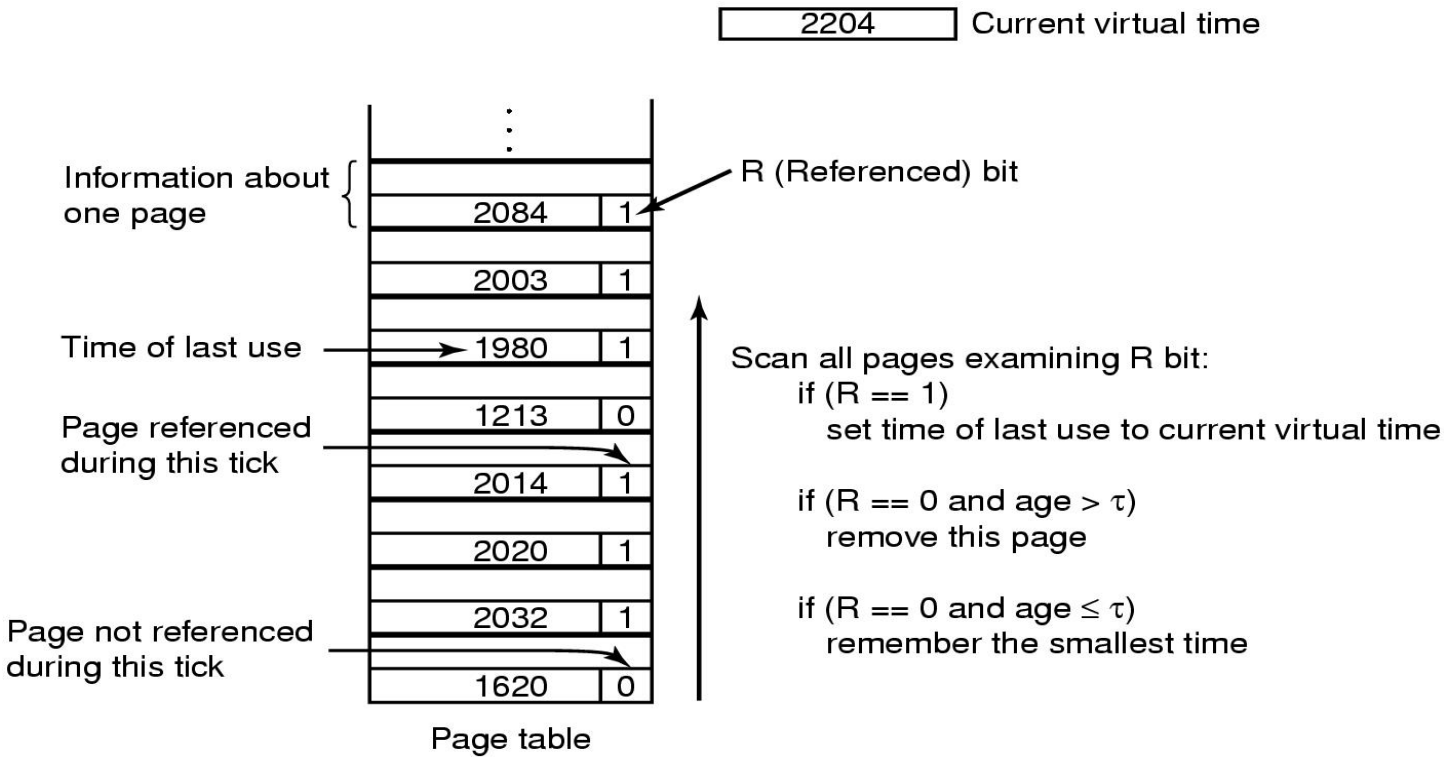


Figure 3-20. The working set algorithm.

The WSClock Page Replacement Algorithm (1)

When the hand comes all the way around to its starting point there are two cases to consider:

- At least one write has been scheduled.
- No writes have been scheduled.

The WSClock Page Replacement Algorithm (2)

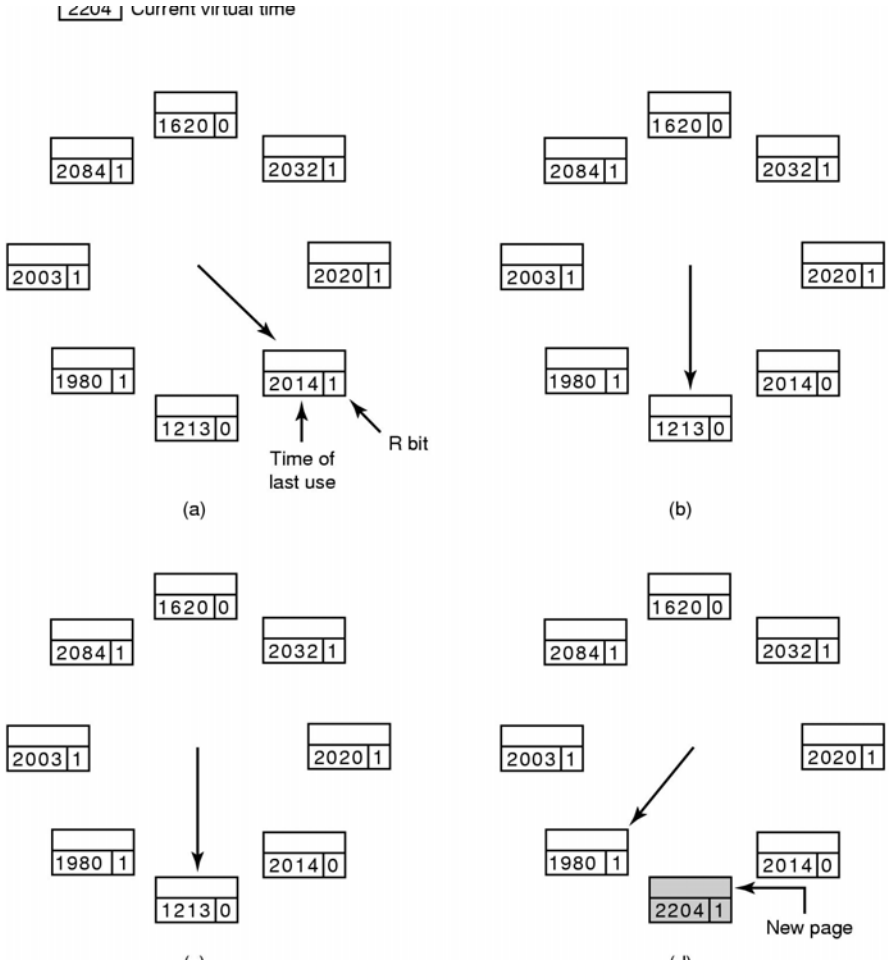


Figure 3-21. Operation of the WSClock algorithm. (a) and (b) give an example of what happens when R = 1.

The WSClock Page Replacement Algorithm (3)

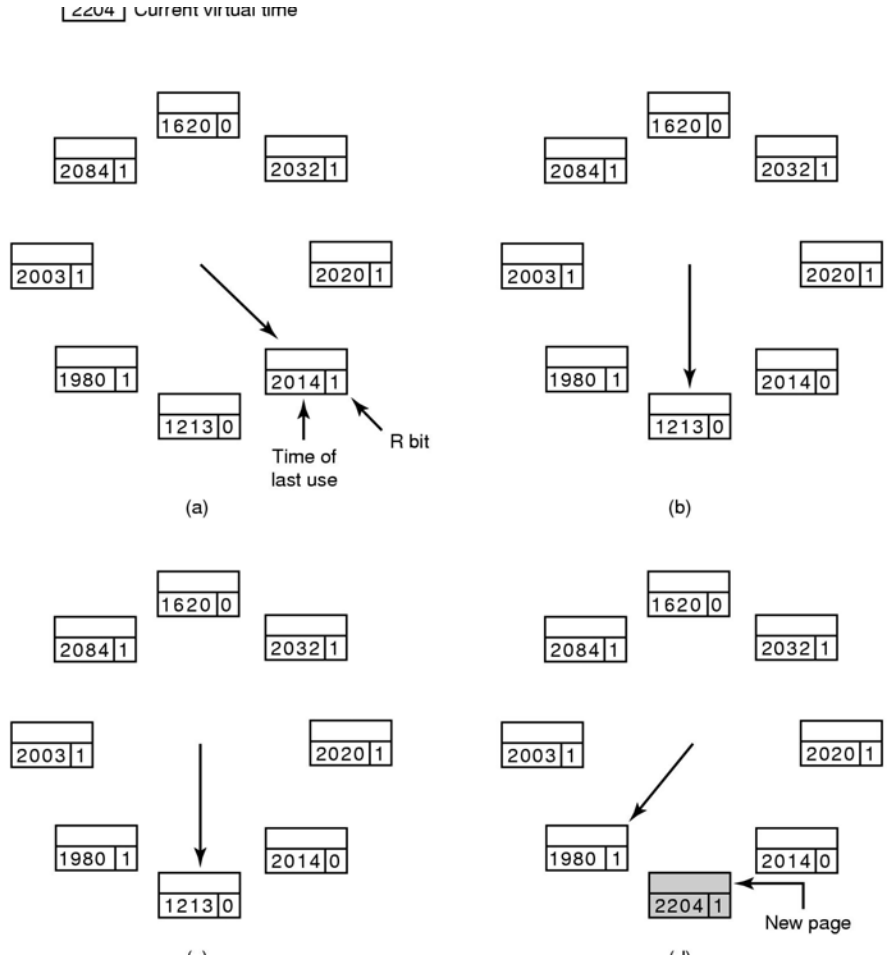


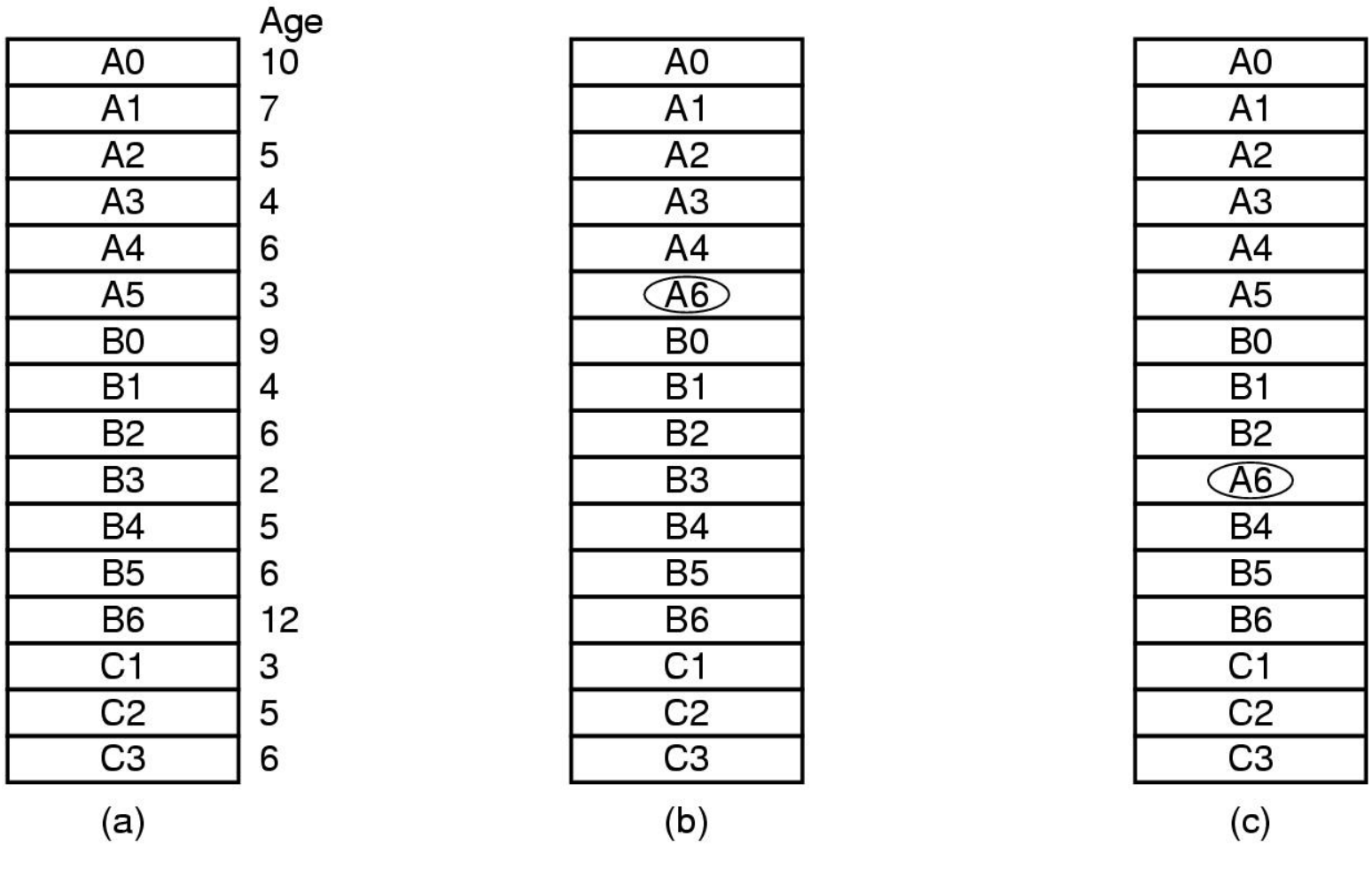
Figure 3-21. Operation of the WSClock algorithm. (c) and (d) give an example of R = 0.

Summary of Page Replacement Algorithms

Algorithm	Comment
Optimal	Not implementable, but useful as a benchmark
NRU (Not Recently Used)	Very crude approximation of LRU
FIFO (First-In, First-Out)	Might throw out important pages
Second chance	Big improvement over FIFO
Clock	Realistic
LRU (Least Recently Used)	Excellent, but difficult to implement exactly
NFU (Not Frequently Used)	Fairly crude approximation to LRU
Aging	Efficient algorithm that approximates LRU well
Working set	Somewhat expensive to implement
WSClock	Good efficient algorithm

Figure 3-22. Page replacement algorithms discussed in the text.

Local versus Global Allocation Policies (1)



(a) Original configuration. (b) Local page replacement. (c) Global page replacement.

Load Control

- Despite good designs, system may still thrash
- When PFF algorithm indicates
  - some processes need more memory
  - but no processes need less
- Solution :
  Reduce number of processes competing for memory
  - swap one or more to disk, divide up pages they held
  - reconsider degree of multiprogramming

Page Size (1)

Small page size

- Advantages
  - less internal fragmentation
  - better fit for various data structures, code sections
  - less unused program in memory
- Disadvantages
  - programs need many pages, larger page tables

Local versus Global Allocation Policies (2)

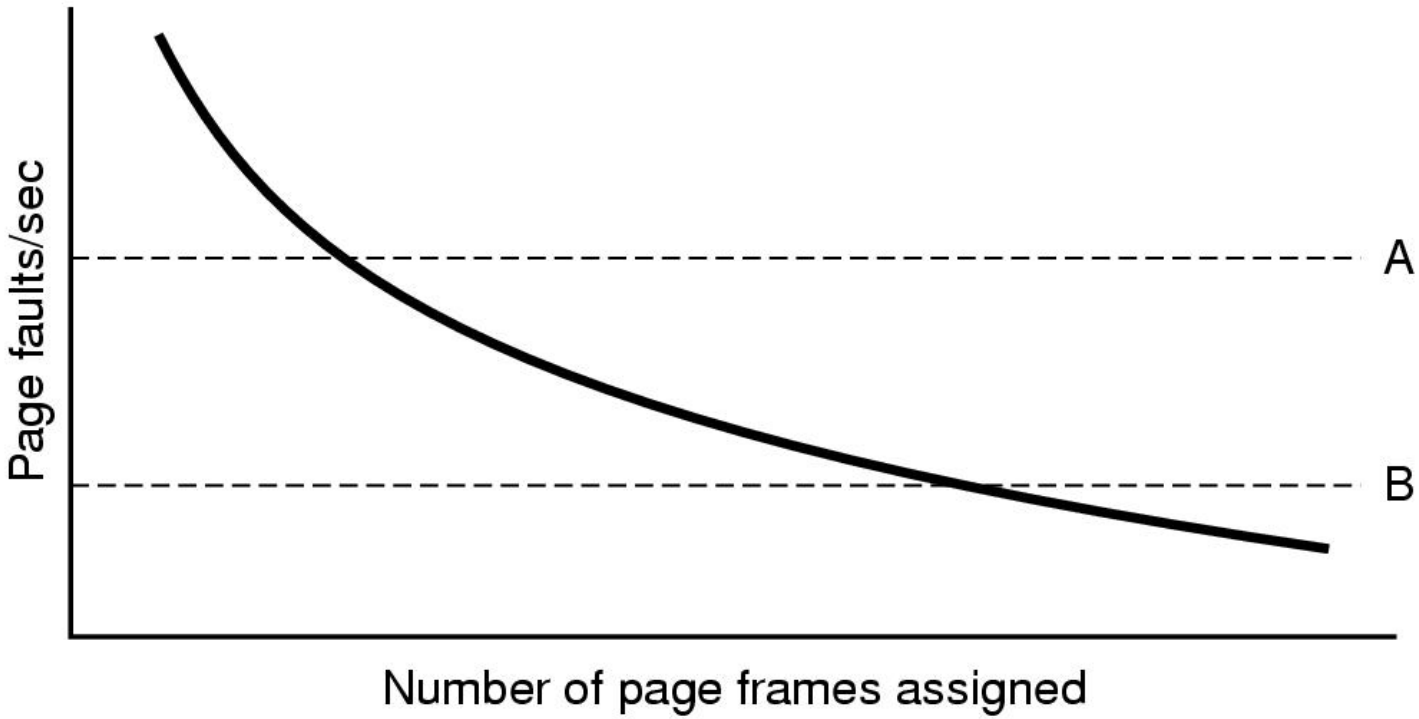


Figure 3-24. Page fault rate as a function of the number of page frames assigned.

Separate Instruction and Data Spaces

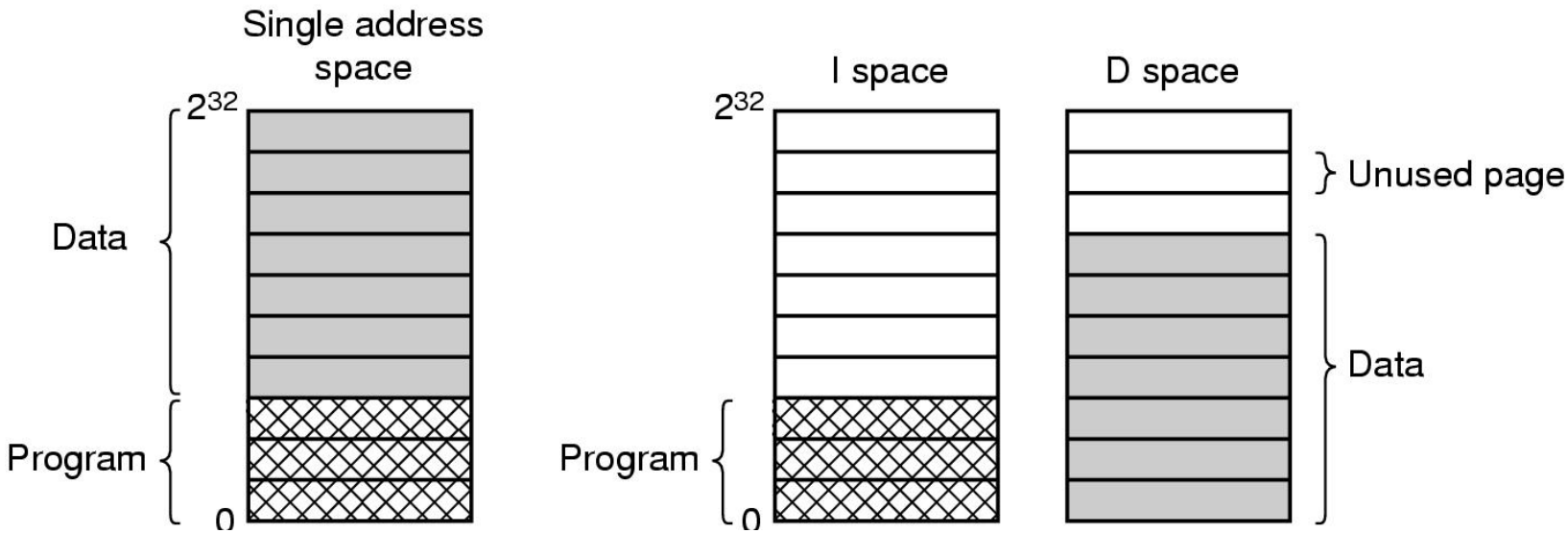


Figure 3-25. (a) One address space.
(b) Separate I and D spaces.

Shared Pages

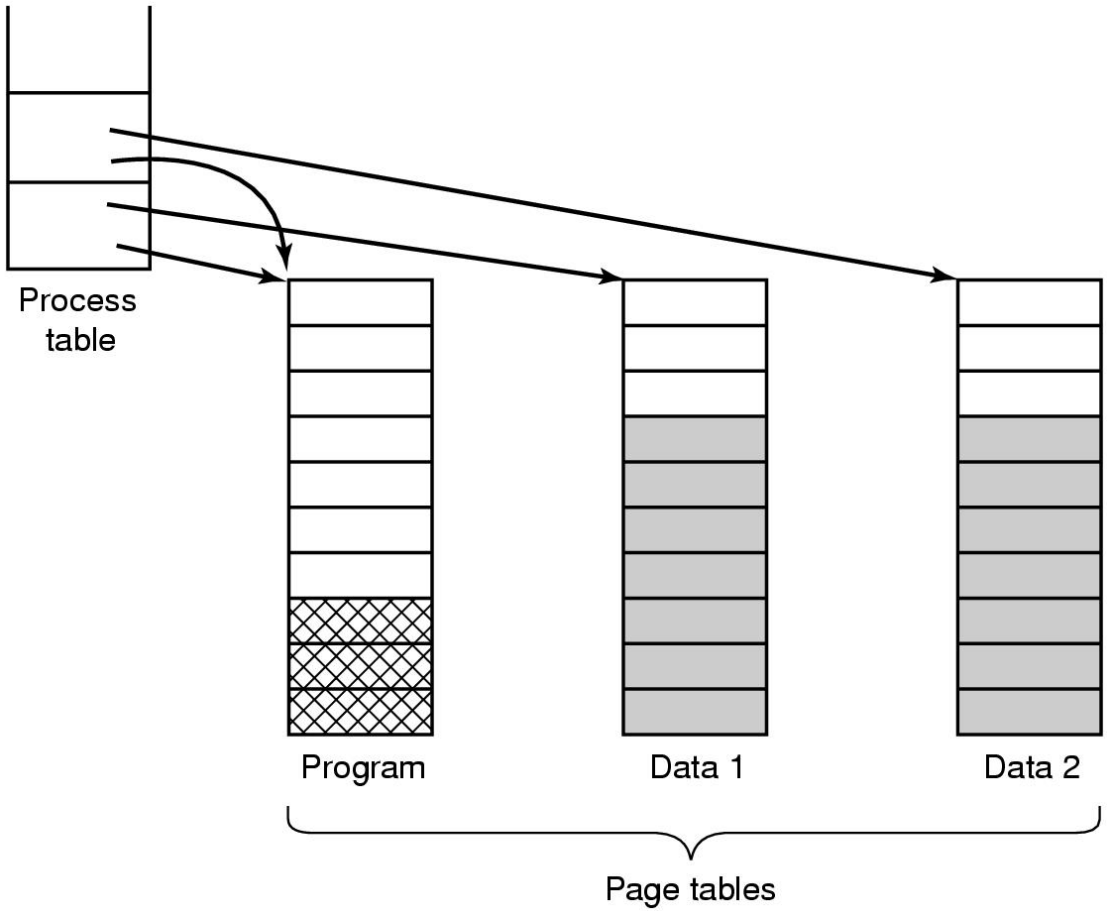


Figure 3-26. Two processes sharing the same program sharing its page table.

Shared Libraries

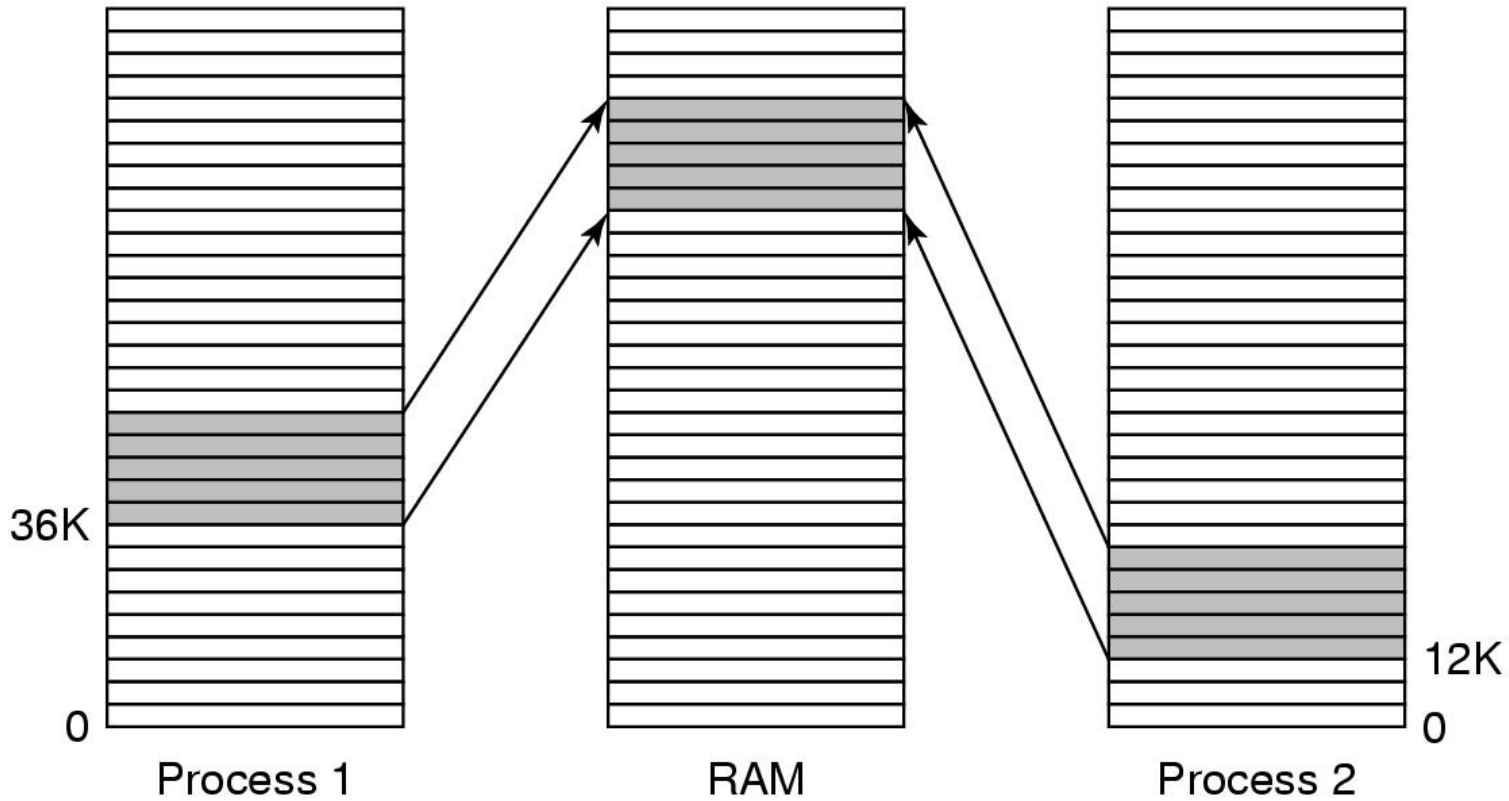


Figure 3-27. A shared library being used by two processes.

Page Fault Handling (1)

- The hardware traps to the kernel, saving the program counter on the stack.
- An assembly code routine is started to save the general registers and other volatile information.
- The operating system discovers that a page fault has occurred, and tries to discover which virtual page is needed.
- Once the virtual address that caused the fault is known, the system checks to see if this address is valid and the protection consistent with the access

Page Fault Handling (2)

- If the page frame selected is dirty, the page is scheduled for transfer to the disk, and a context switch takes place.
- When page frame is clean, operating system looks up the disk address where the needed page is, schedules a disk operation to bring it in.
- When disk interrupt indicates page has arrived, page tables updated to reflect position, frame marked as being in normal state.

Page Fault Handling (3)

- Faulting instruction backed up to state it had when it began and program counter reset to point to that instruction.
- Faulting process scheduled, operating system returns to the (assembly language) routine that called it.
- This routine reloads registers and other state information and returns to user space to continue execution, as if no fault had occurred.

Instruction Backup

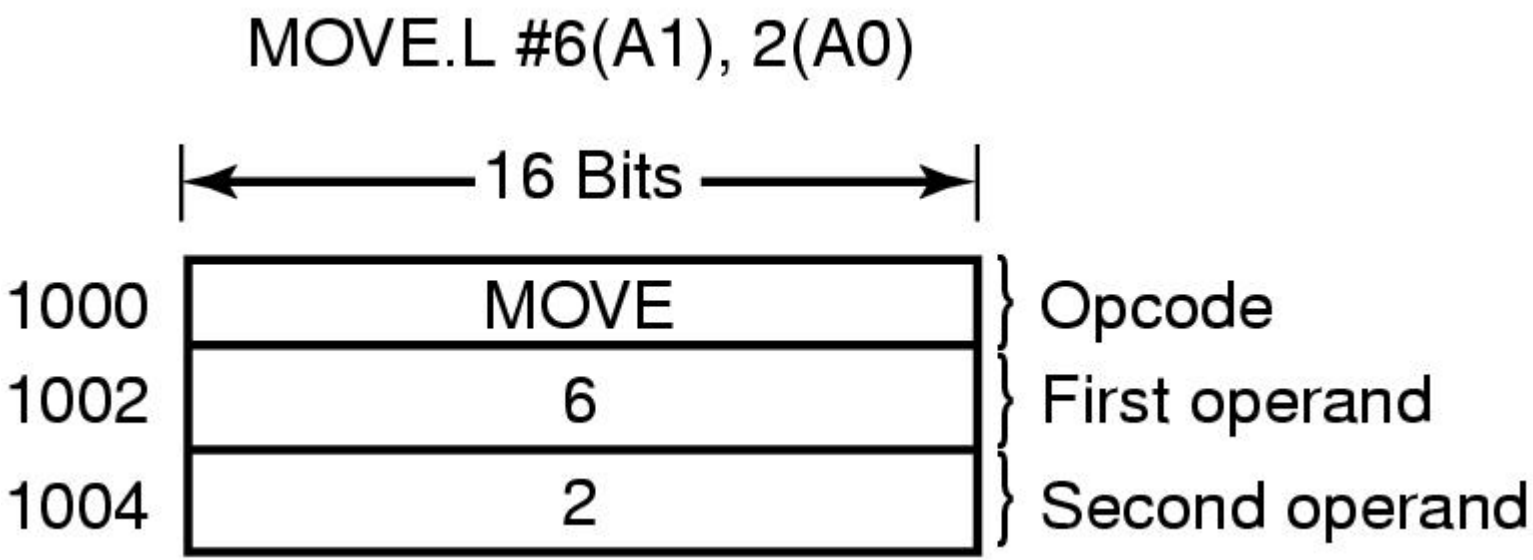


Figure 3-28. An instruction causing a page fault.

Backing Store (1)

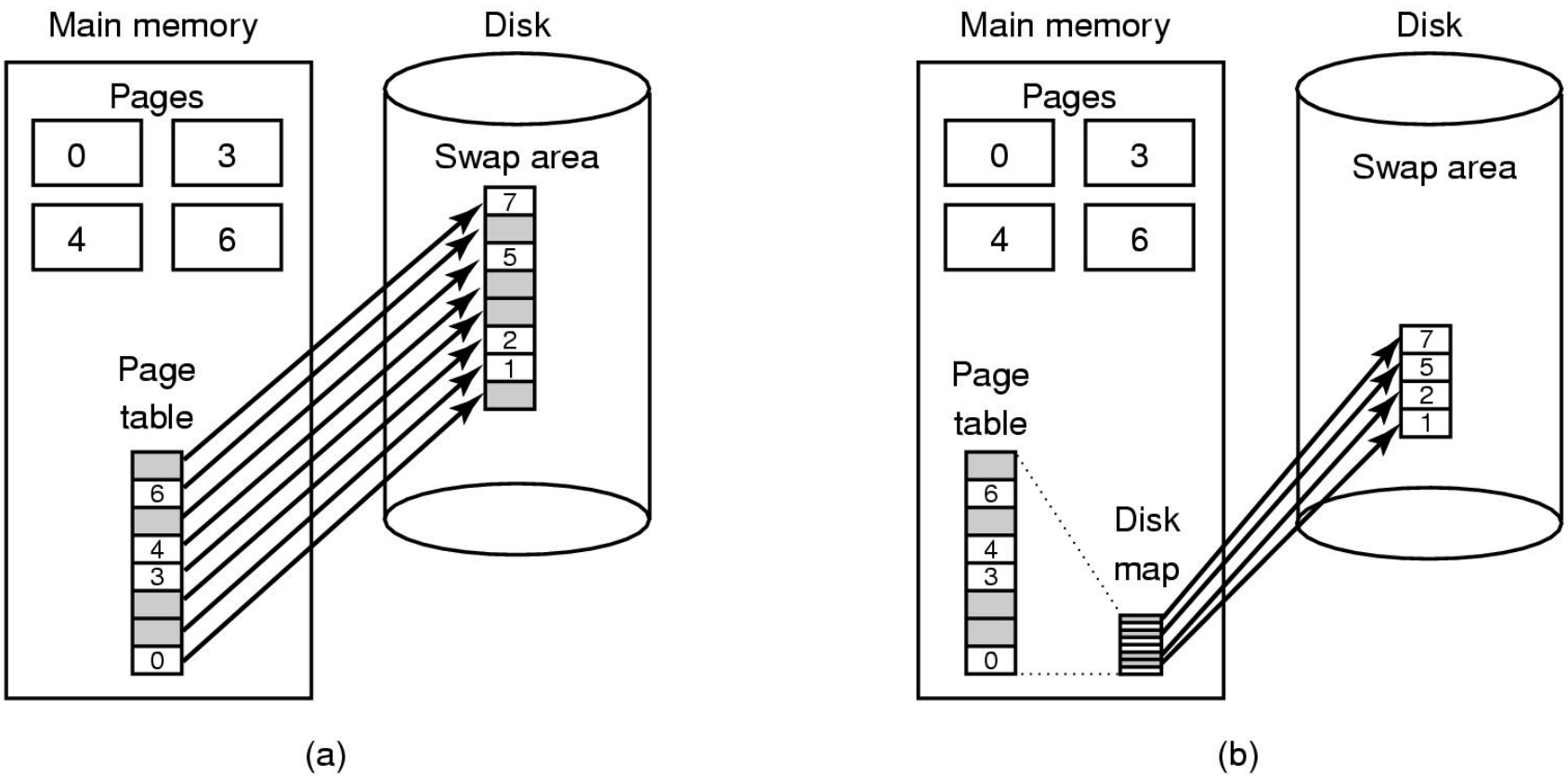


Figure 3-29. (a) Paging to a static swap area.

Backing Store (2)

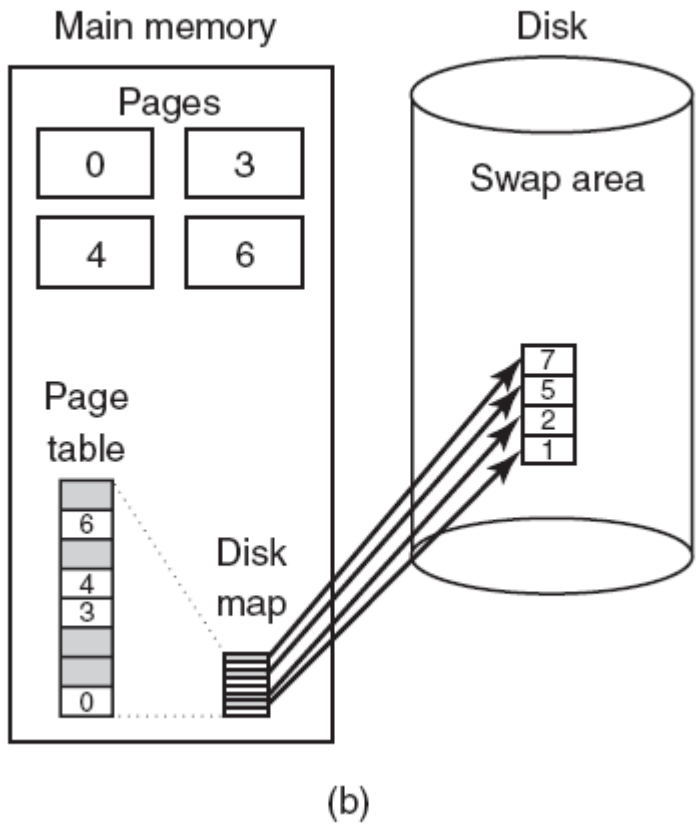


Figure 3-29. (b) Backing up pages dynamically.

Separation of Policy and Mechanism (1)

Memory management system is divided into three parts:

- A low-level MMU handler.
- A page fault handler that is part of the kernel.
- An external pager running in user space.

Separation of Policy and Mechanism (2)

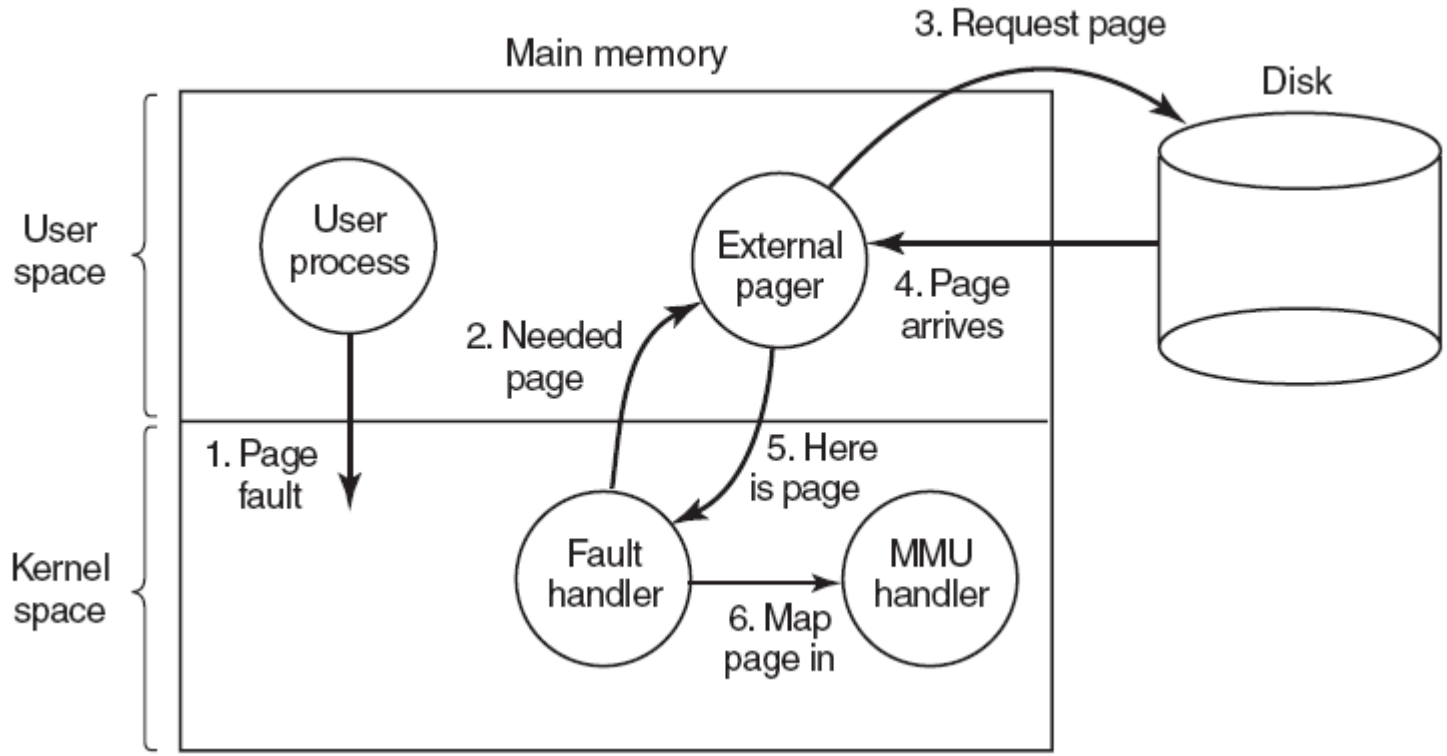


Figure 3-30. Page fault handling with an external pager.