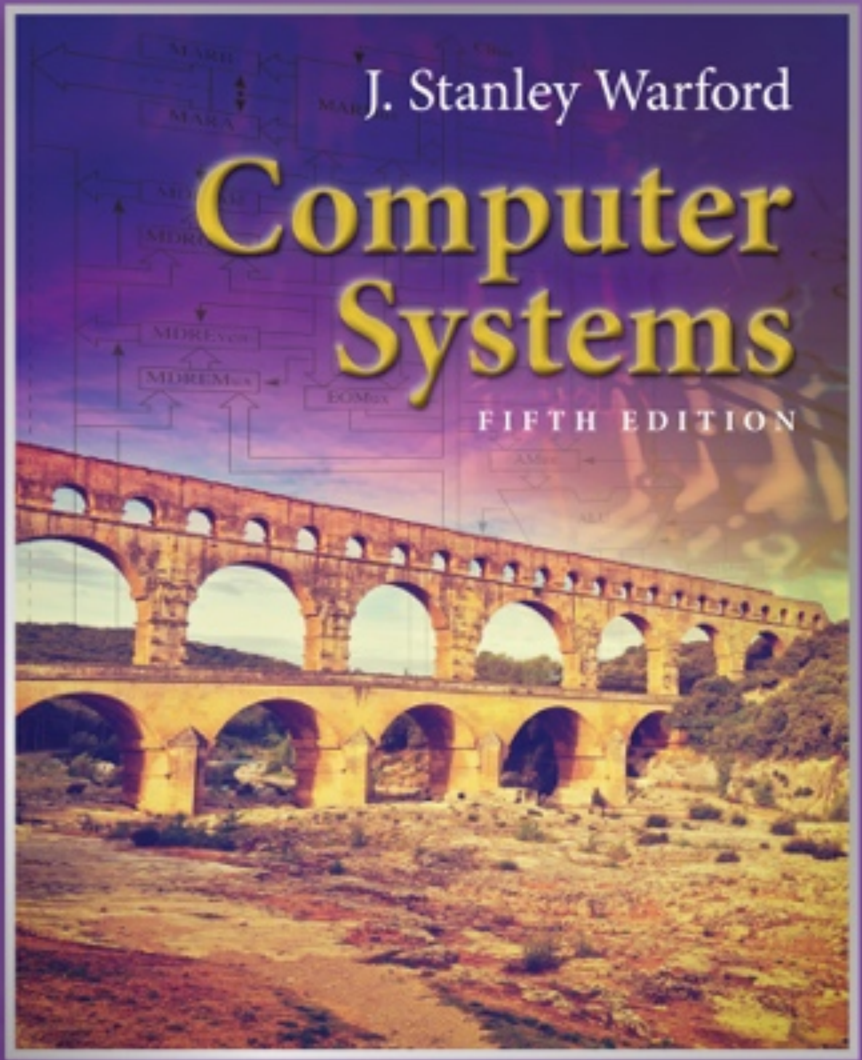
Chapter 9

Storage Management



Memory allocation techniques

- Uniprogramming
- Fixed-partition multiprogramming
- Variable-partition multiprogramming
- Paging
- Virtual memory

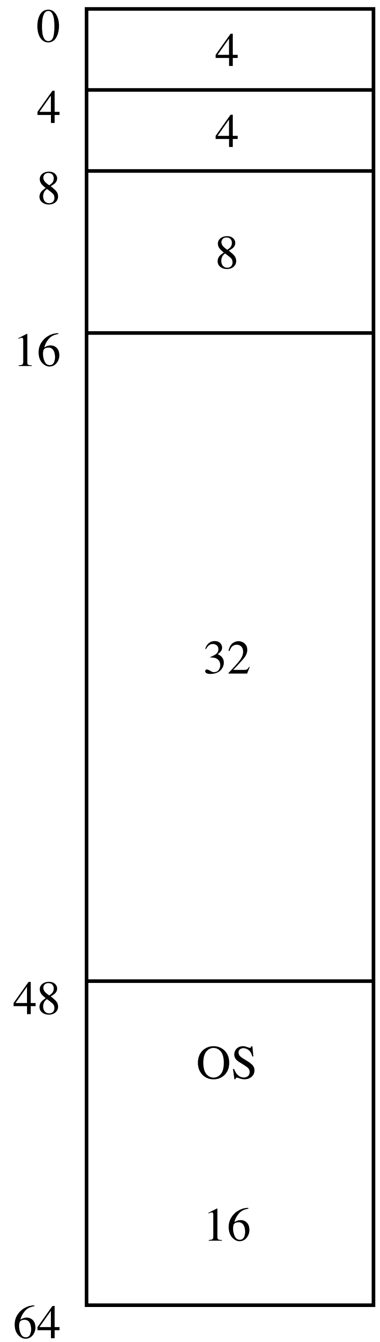
Uniprogramming

- Operating system resides at one end of memory
- Application at the other end
- System only executes one job at time
- Example: Pep/8 operating system
- Disadvantages: Inflexible, CPU time wasted waiting for I/O

Fixed-partition multiprogramming

- Operating system in one fixed reserved partition of memory
- Multiple processes in fixed partitions of memory
- Must solve the address problem

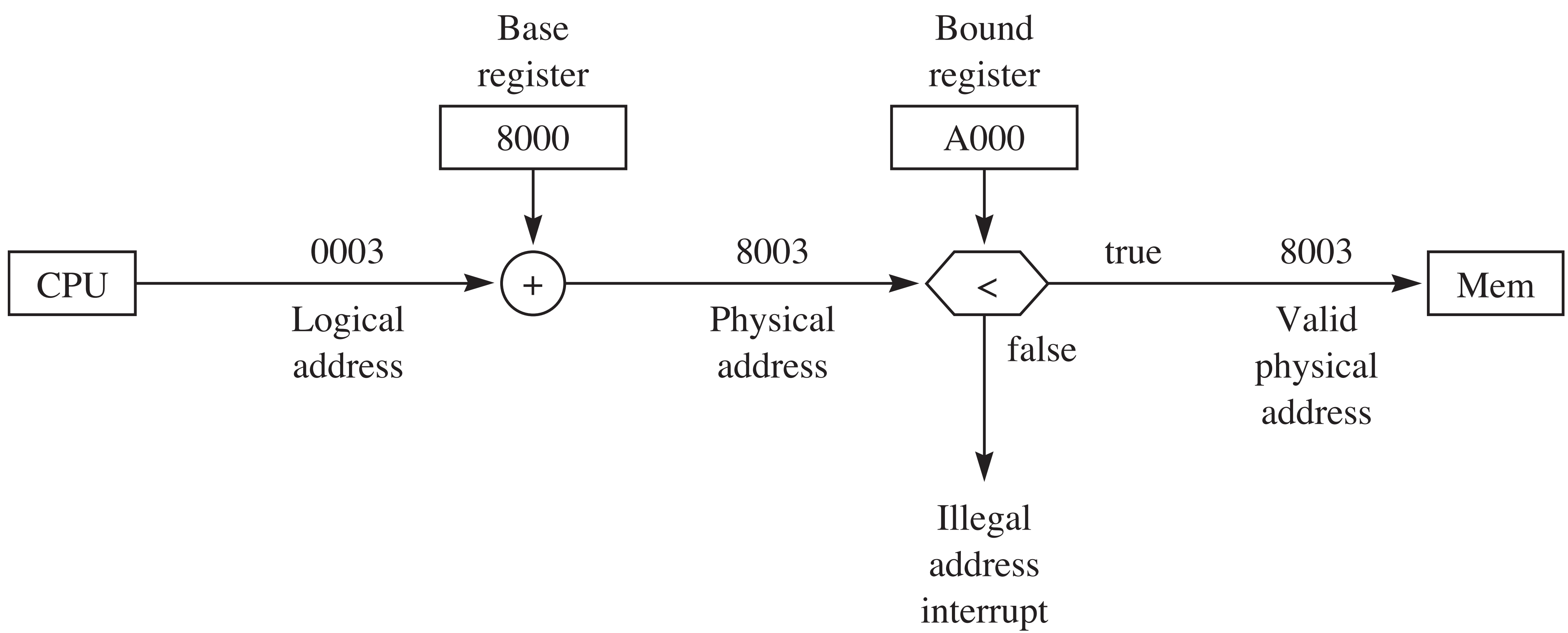
Figure 9.1



Logical addresses

- Logical address is the address generated by the assembler assuming the program begins at address 0
- Physical address =

  logical address + partition address
- Base register converts from logical to physical
- Bound register keeps program isolated

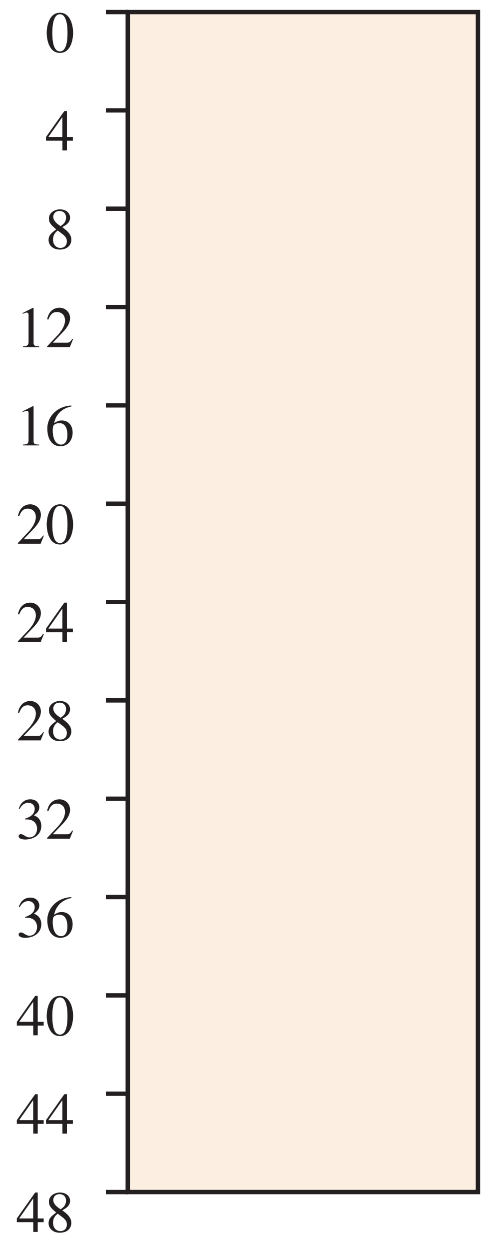

Base
register
8000
Bound
register
A000
CPU
0003
Logical
address
+
8003
Physical
address
<
true
8003
Valid
physical
address
Mem
false
Illegal
address
interrupt

Problems with fixed-partitions

- Scheduling a small job in a large partition because the small partitions are all used
- Determining the optimal partition in the first place

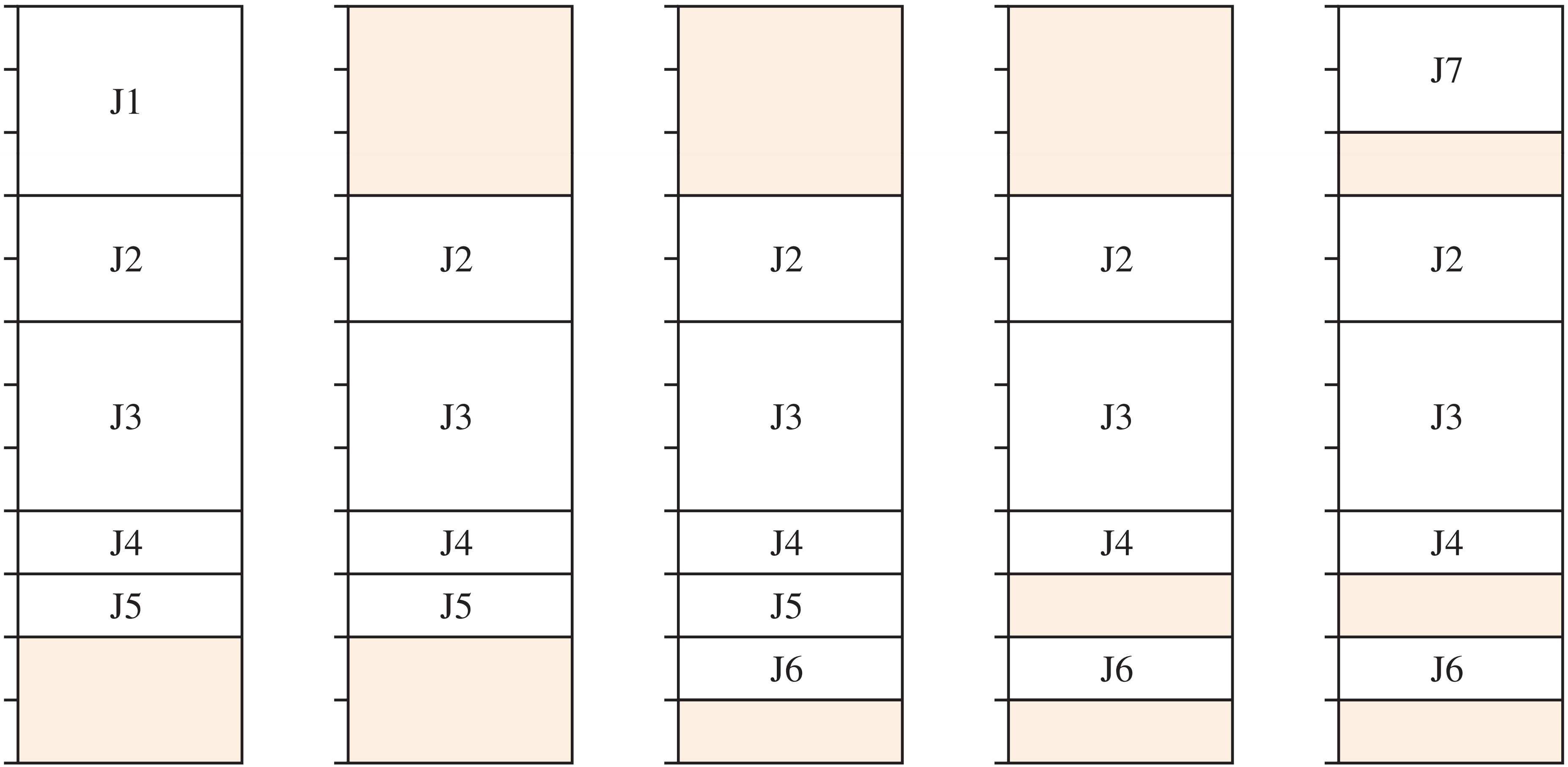
Variable-partition multiprogramming

- Establish a partition only when a job is loaded into memory
- The size of the partition can match the size of the job
- A region available for use by an incoming job is a *hole*


0
4
8
12
16
20
24
28
32
36
40
44
48

Job	Size	Action
J1	12	Start
J2	8	Start
J3	12	Start
J4	4	Start
J5	4	Start
J1	12	Stop
J6	4	Start
J5	4	Stop
J7	8	Start
J8	8	Start

Best-fit algorithm



(a) J1 to J5 starts.

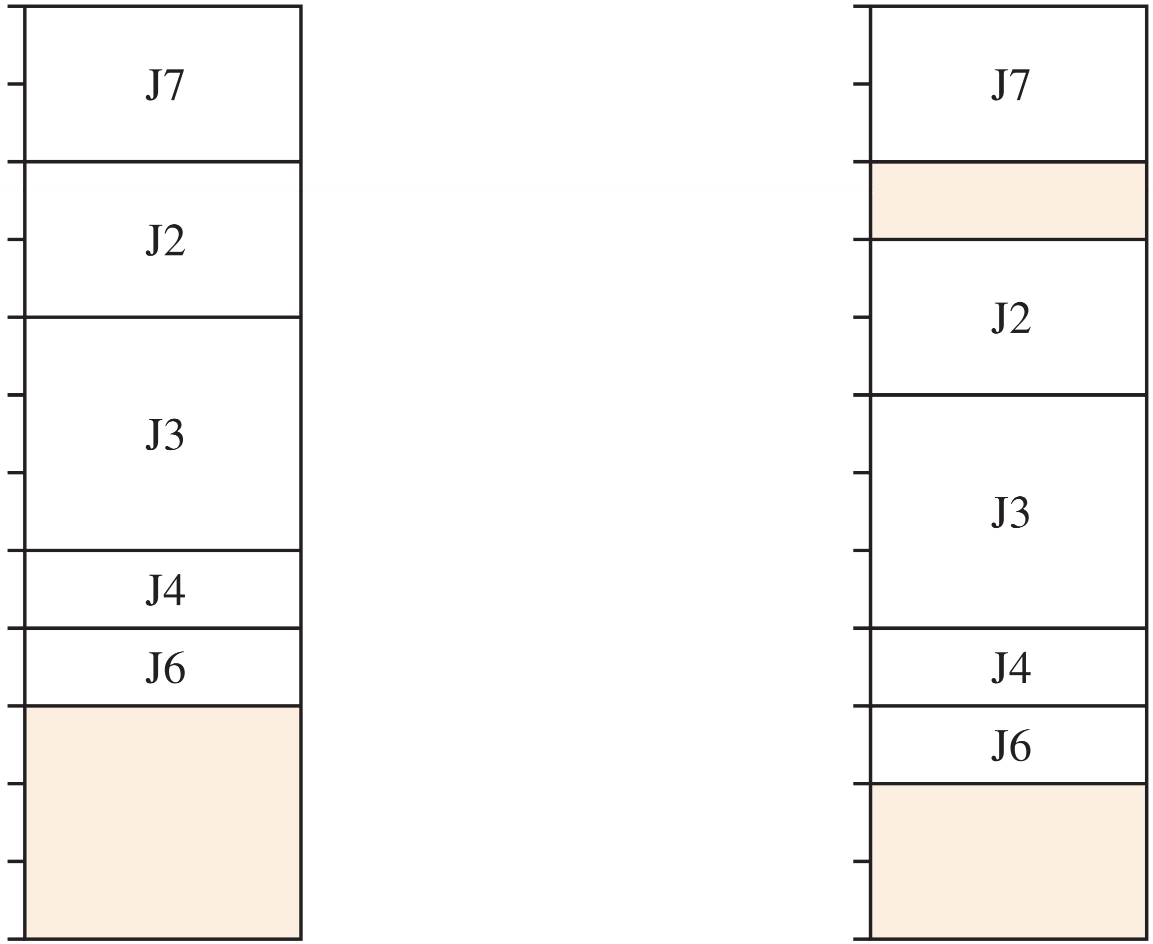
(b) J1 stops.

(c) J6 starts.

(d) J5 stops.

(e) J7 starts.

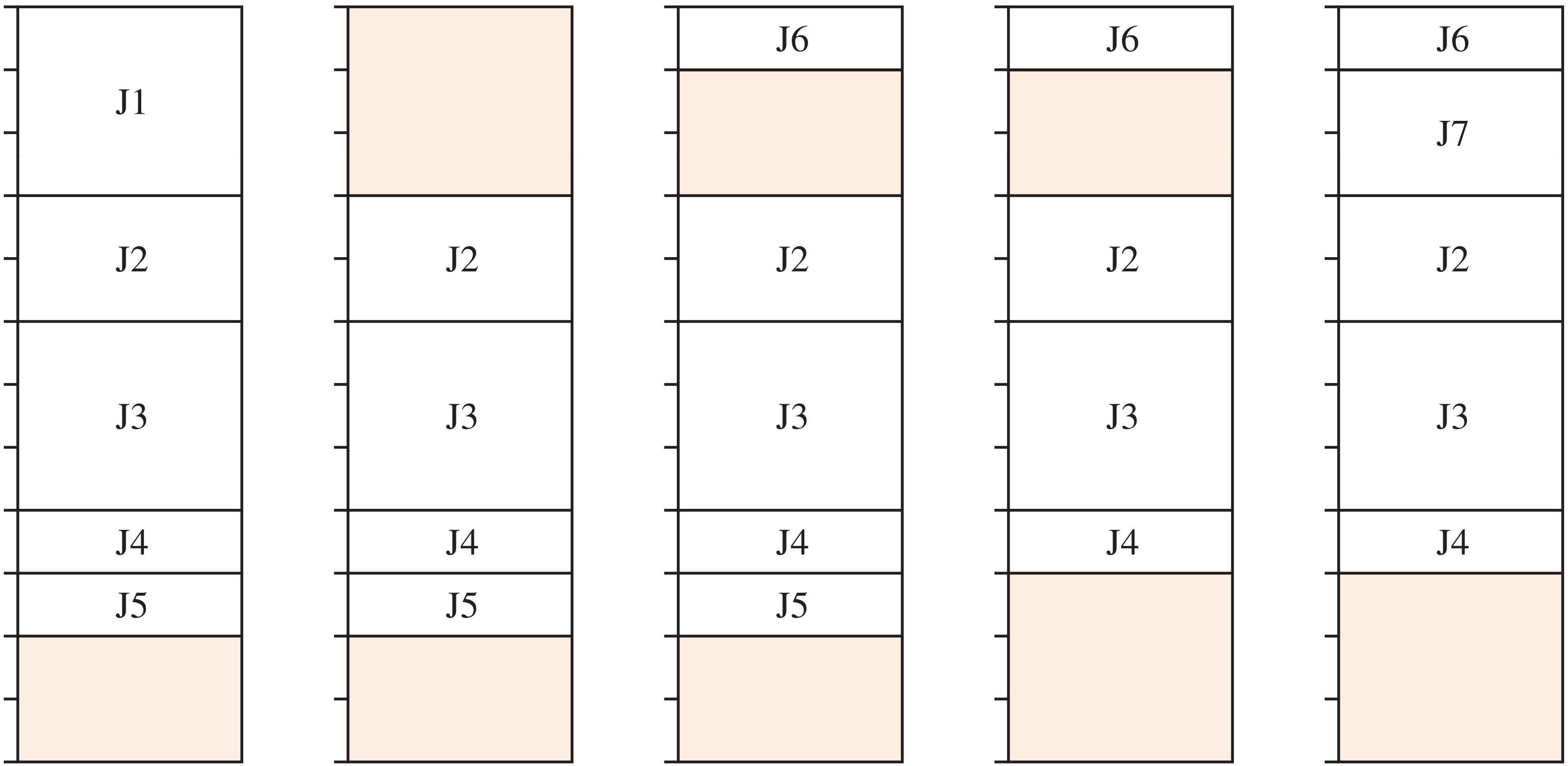
Compacting main memory



(a) Shifting all jobs to the top.

(b) Shifting only J6.

First-fit algorithm



(a) J1 to J5 starts.

(b) J1 stops.

(c) J6 starts.

(d) J5 stops.

(e) J7 starts.

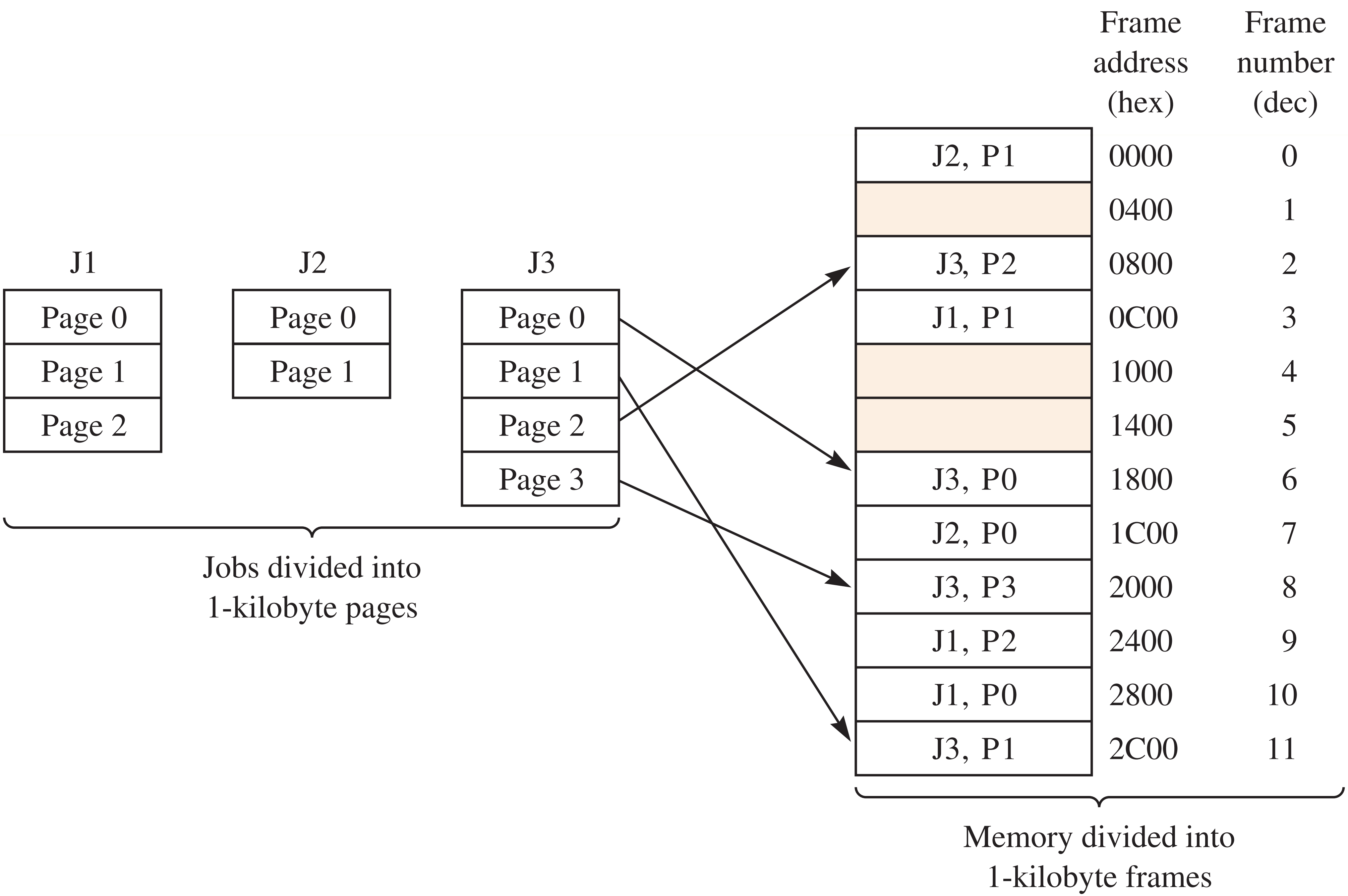
Problems with variable partitions

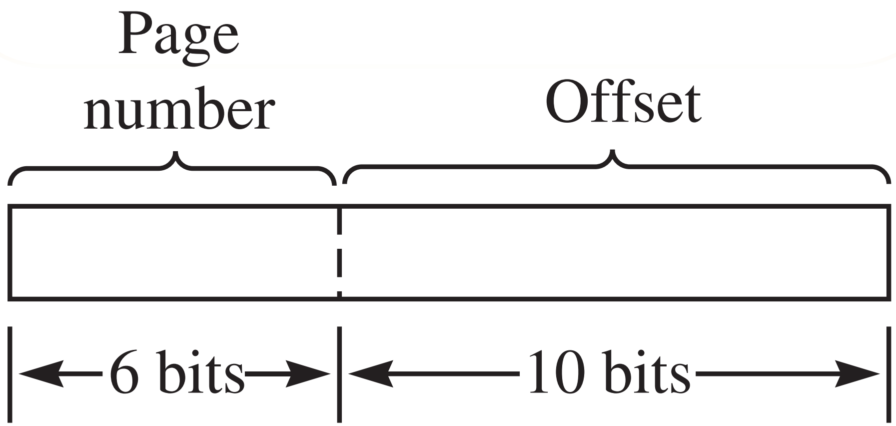
- Fragmentation
- Consolidating holes is time-consuming

Paging

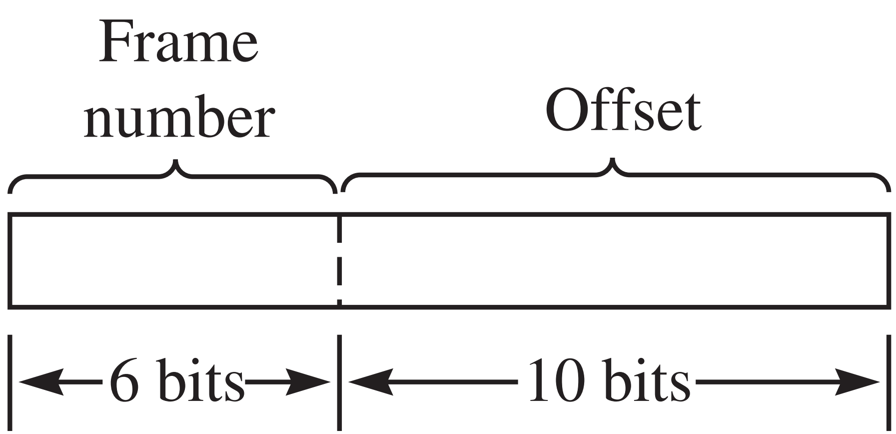
- Rather than coalesce several small holes to fit the program, fragment the program to fit the holes
- A job is divided into pages
- Main memory is divided into frames, each one the same size as a page
- No coalescing of holes is ever required

Figure 9.8

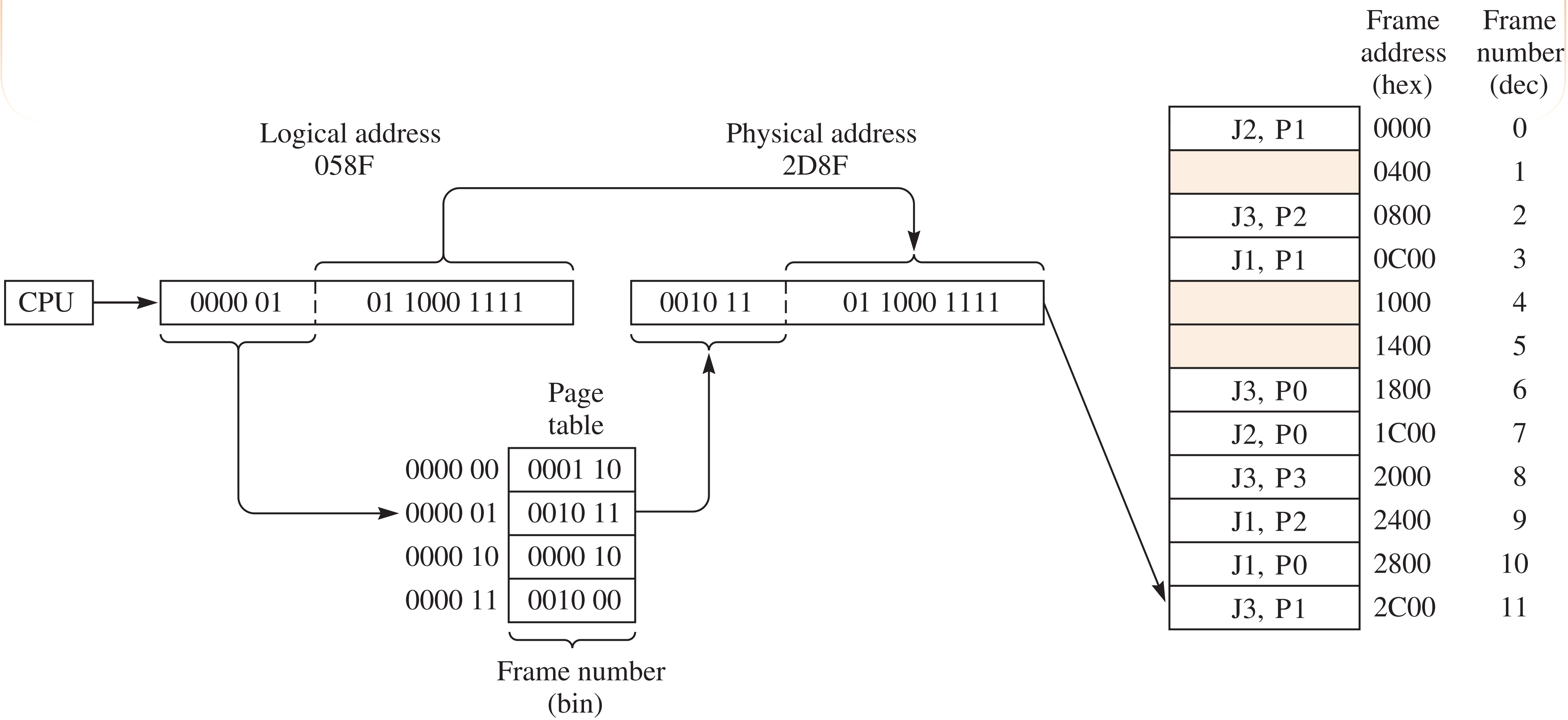

J1
Page 0
Page 1
Page 2
J2
Page 0
Page 1
J3
Page 0
Page 1
Page 2
Page 3
Jobs divided into
1-kilobyte pages
Frame
address
(hex)
Frame
number
(dec)
J2, P1
0000
0
0400
1
J3, P2
0800
2
J1, P1
0C00
3
1000
4
1400
5
J3, P0
1800
6
J2, P0
1C00
7
J3, P3
2000
8
J1, P2
2400
9
J1, P0
2800
10
J3, P1
2C00
11
Memory divided into
1-kilobyte frames



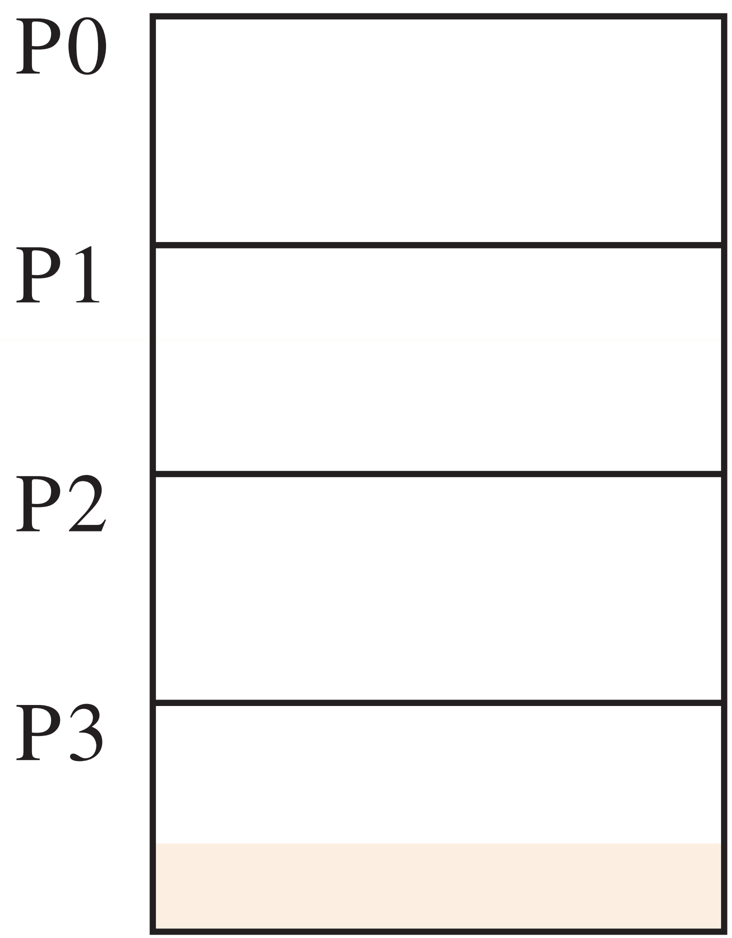
(a) Logical address.



(b) Physical address.


Logical address
058F
Physical address
2D8F
CPU
0000 01
01 1000 1111
0010 11
01 1000 1111
Page
table
0000 00
0001 10
0000 01
0010 11
0000 10
0000 10
0000 11
0010 00
Frame number
(bin)
Frame
address
(hex)
Frame
number
(dec)
J2, P1
0000
0
0400
1
J3, P2
0800
2
J1, P1
0C00
3
1000
4
1400
5
J3, P0
1800
6
J2, P0
1C00
7
J3, P3
2000
8
J1, P2
2400
9
J1, P0
2800
10
J3, P1
2C00
11

Internal fragmentation



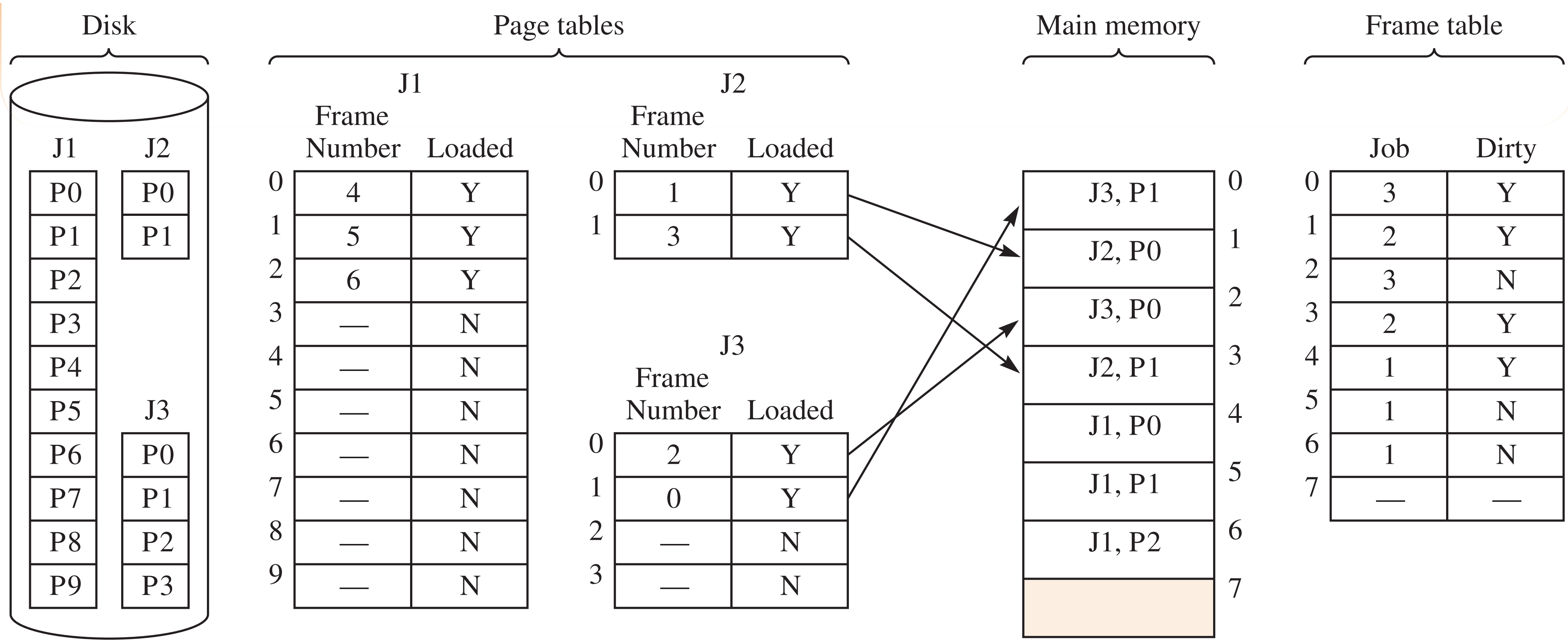
Problem with paging

- To execute a program, the entire program must be loaded into memory
- Most large programs have many sections of code that never execute
- Memory is used inefficiently with parts of the program taking up main memory unnecessarily

Virtual memory

- Cycle pages of the program from disk into memory only when they need to be executed
- The page that is executing in memory together with the pages in memory that have recently been executed is the program's *working set*
- As the program progresses, pages enter and leave the working set

Figure 9.12


Disk
J1
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
J2
P0
P1
J3
P0
P1
P2
P3
Page tables
J1
Frame Number
Loaded
0 4 Y
1 5 Y
2 6 Y
3 — N
4 — N
5 — N
6 — N
7 — N
8 — N
9 — N
J2
Frame Number
Loaded
0 1 Y
1 3 Y
J3
Frame Number
Loaded
0 2 Y
1 0 Y
2 — N
3 — N
Main memory
J3, P1 0
J2, P0 1
J3, P0 2
J2, P1 3
J1, P0 4
J1, P1 5
J1, P2 6
7
Frame table
Job
Dirty
0 3 Y
1 2 Y
2 3 N
3 2 Y
4 1 Y
5 1 N
6 1 N
7 — —

Page tables

- One page table for each program
- Converts logical address to physical address as in paging
- Loaded bit is 1 if the page is in memory
- A *page fault* occurs if the program needs to read or write a page that is not in memory
  - Page is loaded into an empty frame
  - If no empty frames, then a page is replaced

Frame tables

- One frame table with an entry for each frame
- Dirty bit initialized to 0 when page is first loaded into memory
- Set to 1 on a `STWr` to the frame, not on a `LDWr` from the frame
- When a page is replaced it is written back to disk only if the dirty bit is set to 1

Frame allocation

- Before a job starts executing, how many frames should be allocated for that job?
- System can allocate frames proportional to the physical size of the code

Page replacement

- First in, first out (FIFO)
  - On a page fault, select the page to be replaced as the one that first entered the set of loaded pages
- Least recently used (LRU)
  - On a page fault, select the page to be replaced as the one that was least recently read from or written to

First In, First Out (FIFO)

Page references		6	8	3	8	6	0	3	6	3	5	3	6
Loaded pages	—	6	8	3	3	3	0	0	6	6	5	3	3
	—	—	6	8	8	8	3	3	0	0	6	5	5
	—	—	—	6	6	6	8	8	3	3	0	6	6
Page fault		F	F	F			F		F		F	F	

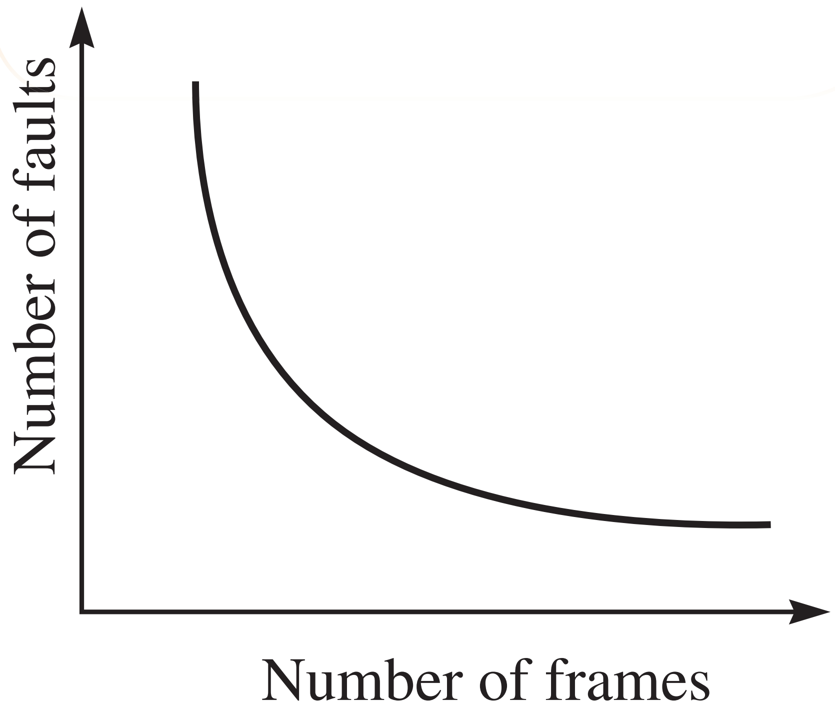
First In, First Out (FIFO)

Page references		6	8	3	8	6	0	3	6	3	5	3	6
Loaded pages	—	6	8	3	3	3	0	0	0	0	5	5	6
	—	—	6	8	8	8	3	3	3	3	0	0	5
	—	—	—	6	6	6	8	8	8	8	3	3	0
	—	—	—	—	—	—	6	6	6	6	8	8	3
Page fault		F	F	F			F				F		F

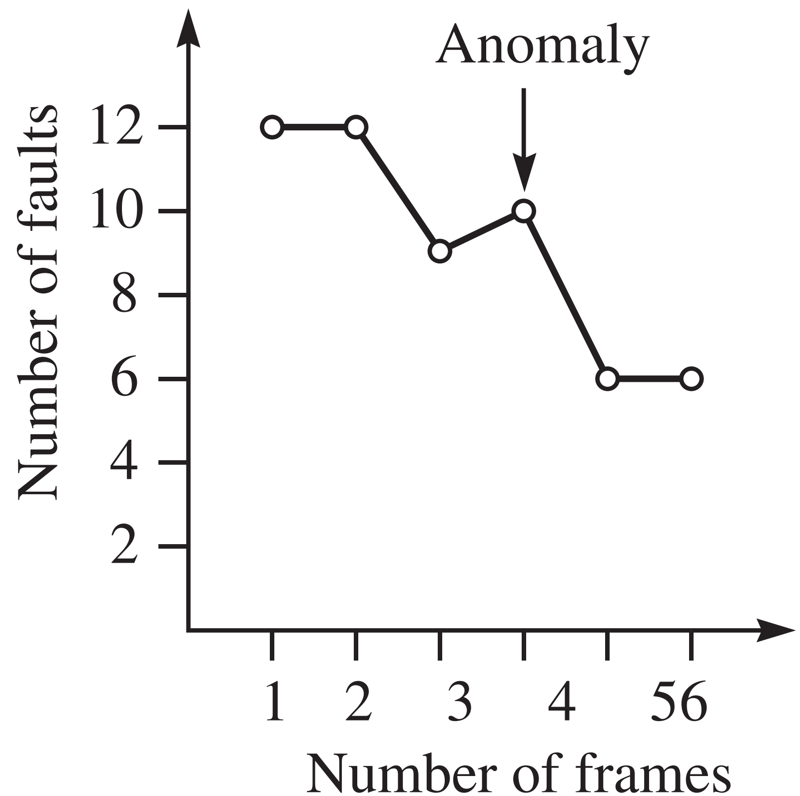
Bélády's anomoly

- In general, the greater the number of frames allocated to a program, the fewer the number of page faults
- In a few cases with FIFO, an increase in the number of frames increases the number of page faults
- Example page reference sequence

0, 1, 2, 3, 0, 1, 4, 0, 1, 2, 3, 4



(a) Expected effect of more frames on the number of page faults.



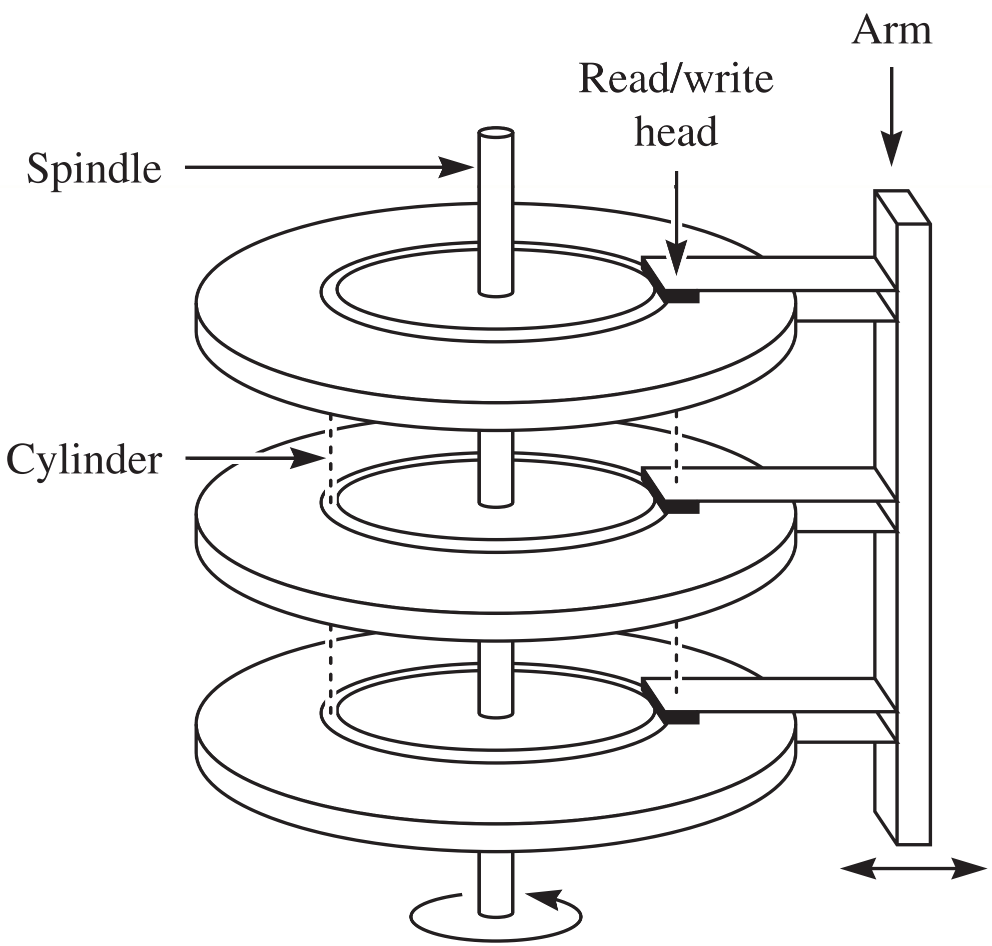
(b) Bélády's anomaly with the FIFO replacement algorithm.

Least Recently Used (LRU)

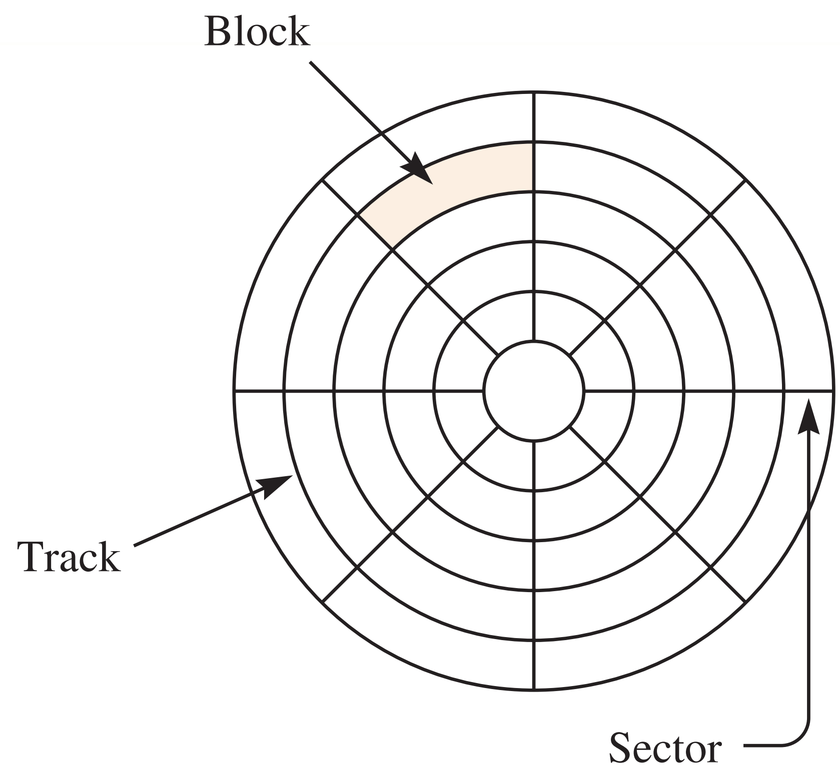
	6	8	3	8	6	0	3	6	3	5	3	6
—	6	8	3	8	6	0	3	6	3	5	3	6
—	—	6	8	3	8	6	0	3	6	3	5	3
—	—	—	6	6	3	8	6	0	0	6	6	5
	F	F	F			F	F			F		

File management

- Create a new file
- Delete a file
- Rename a file
- Open a file for editing
- Read the next data item from the file



(a) A hard disk drive.



(b) A single disk.

Contributions to the disk access time

- Seek time
  - Time for head to reach cylinder
- Latency
  - Time for start of sector to rotate to head
- Transmission time
  - Time for sector to pass under head

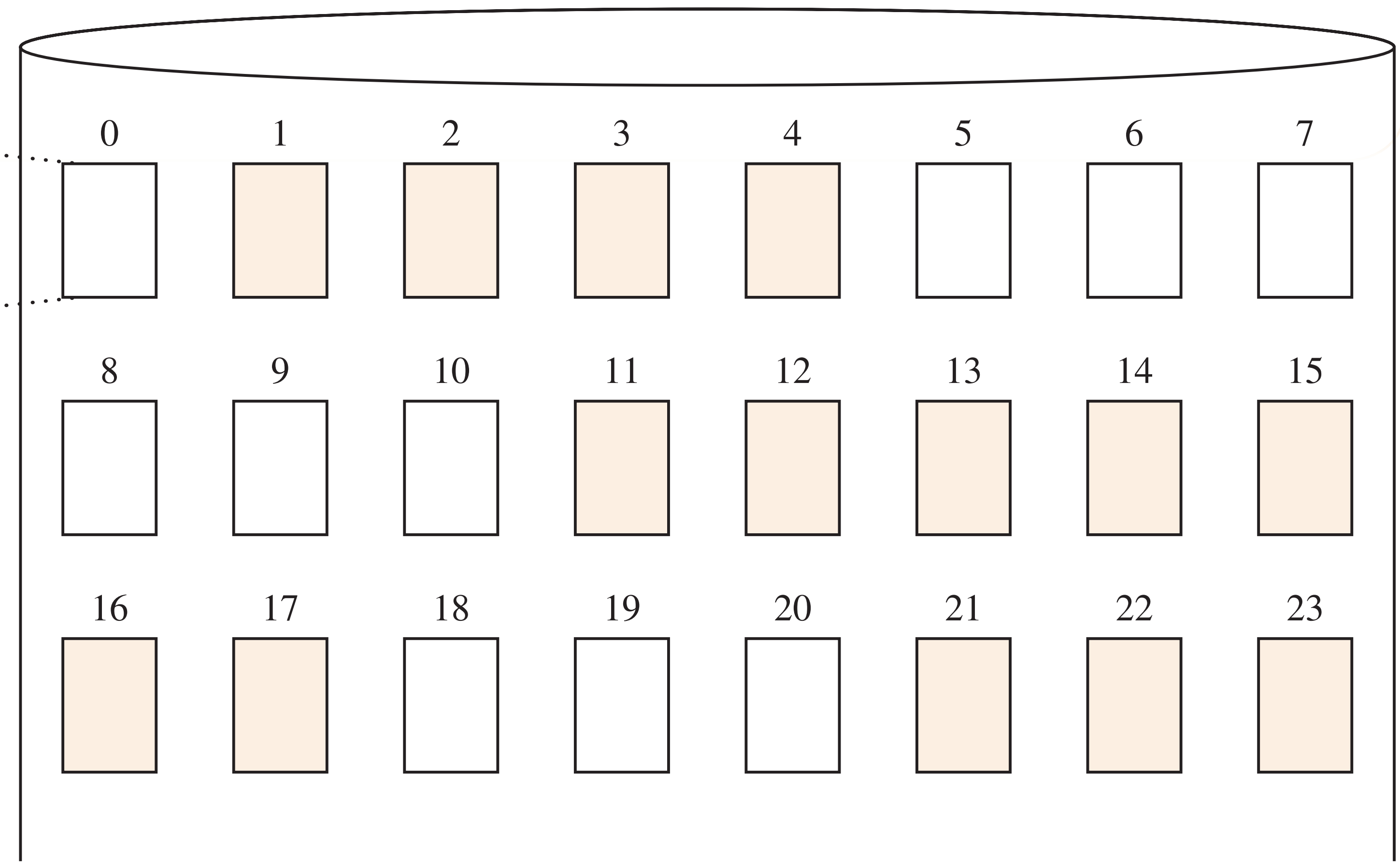
File allocation techniques

- Contiguous
- Linked
- Indexed

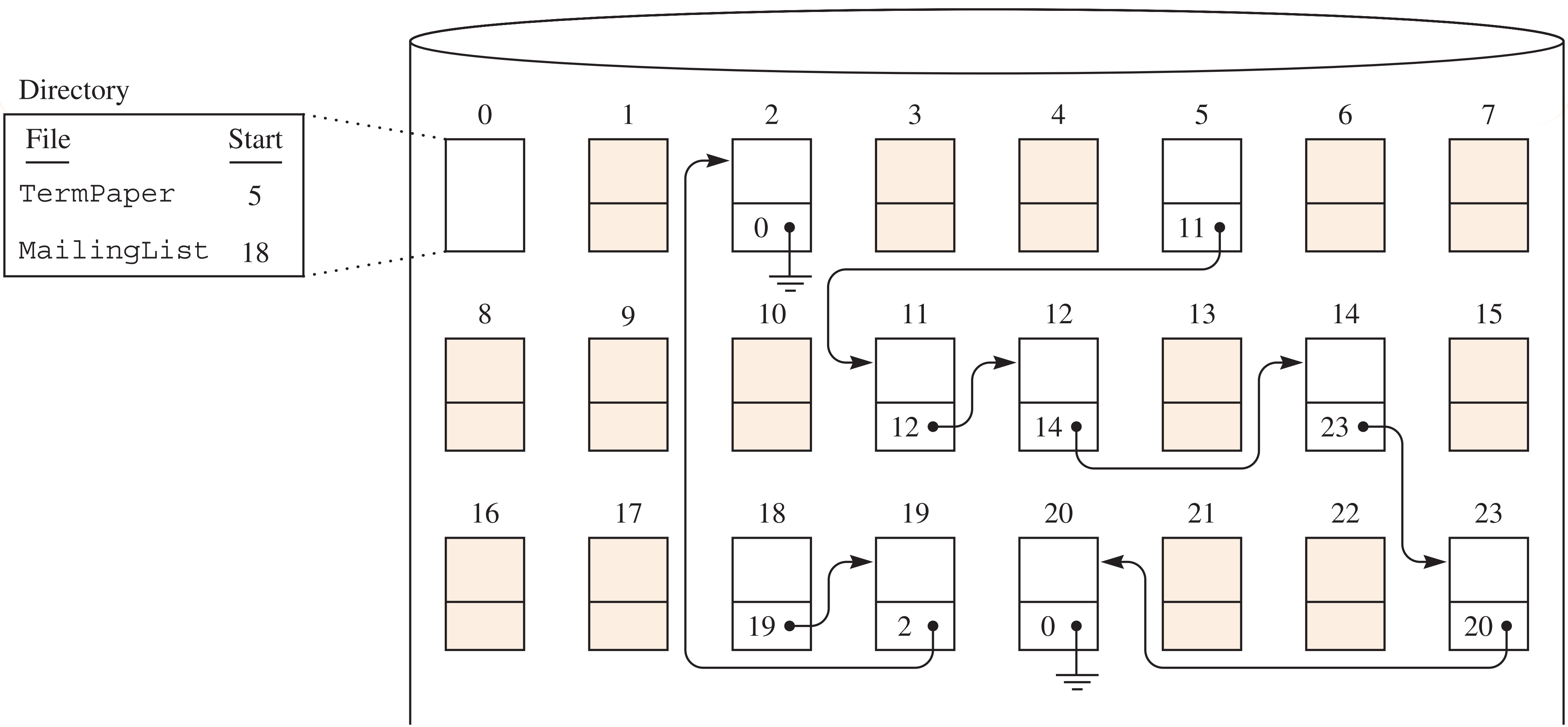
Contiguous allocation

Directory

File	Start	Size
TermPaper	5	6
MailingList	18	3



Linked allocation



Indexed allocation