Memory Management

Memory Hierarchy

- Memory system consists of
  - Cache: volatile
  - Main memory (ROM+RAM): volatile
  - Hard Disk: non-volatile
  - CDs, DVD, Tape storage: non-volatile
  - Removable Media: volatile

Cost (per byte of storage)

Size (in bytes)

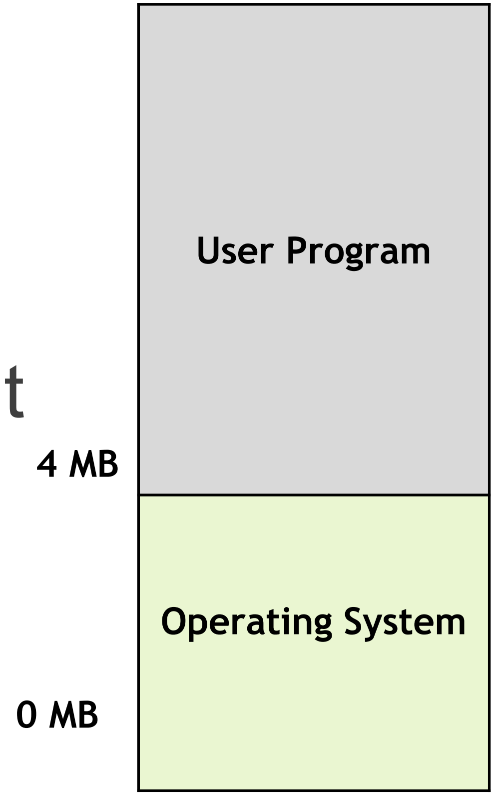
- CPU can directly access main memory and cache
- Unless data is brought to main memory, it is not accessed by CPU

Memory Management

- Transparency: Hide implementation details of memory hierarchy
- Efficiency: Manage memory in time-efficient, space-efficient manner
    - Keep track of parts of memory are in use,
    - Allocate memory to processes when they need it
    - Deallocate it when they are done.
- Protection:
    - Restricting access to memory allocated to OS
    - Isolation between memory allocated to other processes
- Facilitate shared memory
- Memory Virtualization: space sharing

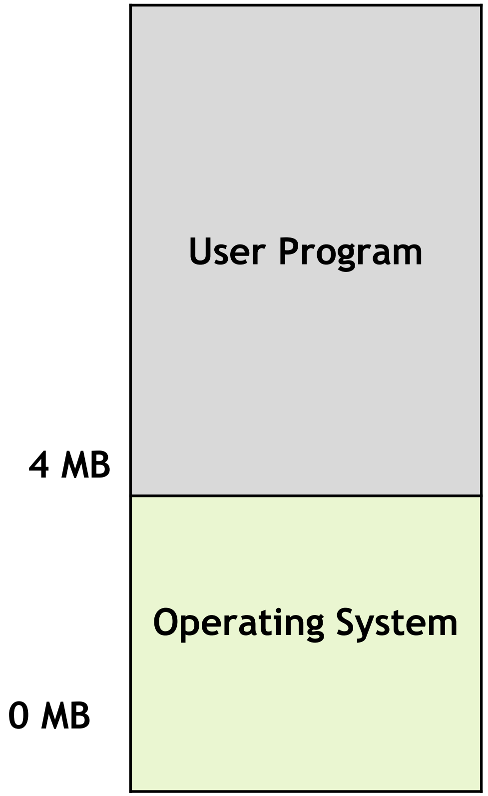
Early OS

- OS occupies lower memory address starting from 0000H.
- Rest of the memory could be used for user program.
- Single user contiguous memory management system
- Issues:
  - Support of multiprogramming?
  - How to adapt to OS growth?



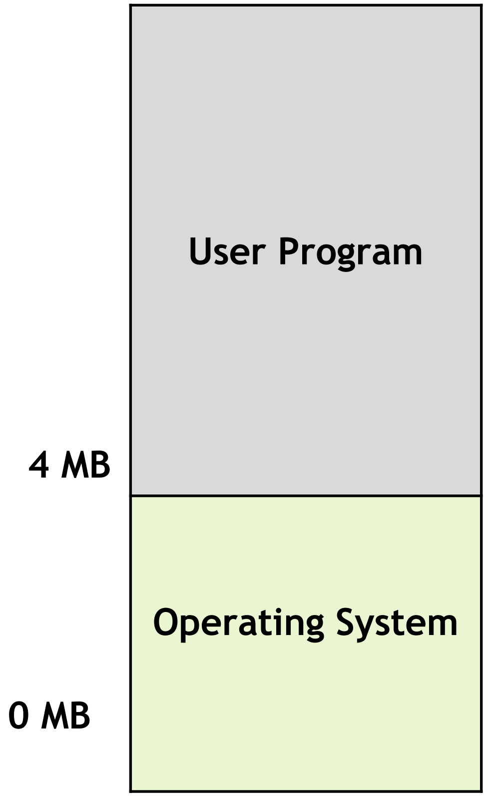
Single User Partition

- Multiprogramming implementation
  - Run a process for some time
  - Save it to secondary memory
  - Load another process into memory
  - Repeat above
- Time-inefficient implementation



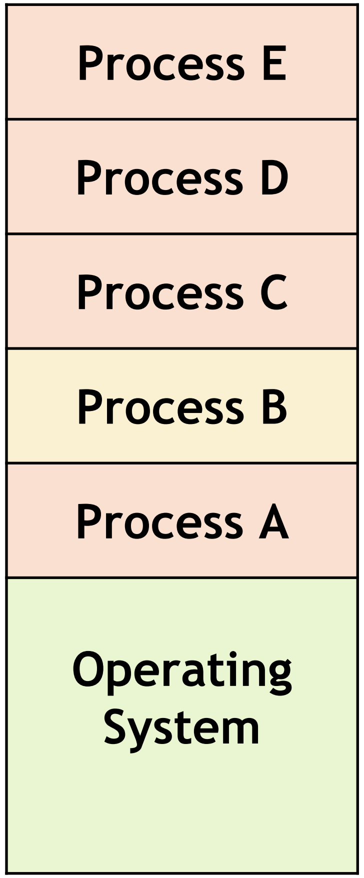
Single User Partition

- What if OS requires more memory
  - Change in OS version
  - Addition of new device drivers
  - Updation of existing device drivers
  - Utilities /system programs to be run
- Adapting to OS growth
  - Relocation
  - Why not use multiple partitions?
  - Solves multiprogramming too



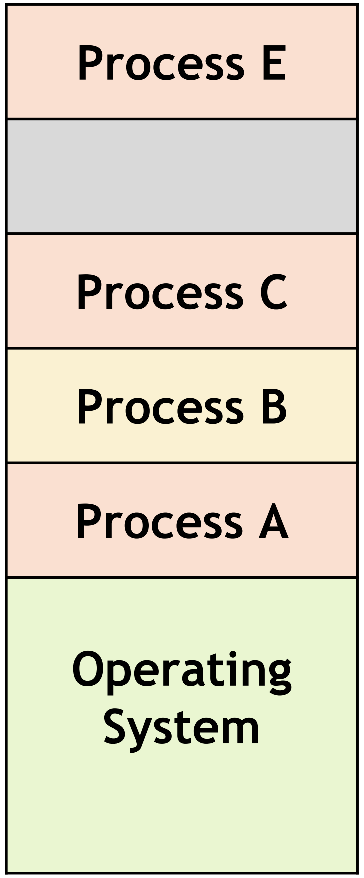
Multiple Partitions

- Space Sharing
  - Multiple partitions
  - Each partition for different user process
  - Relocation as per partition address
  - Implements multiprogramming
  - Caters to OS growth
- Memory partitions can be
  - Fixed Size (Static Partitions)
  - Variable Size (Dynamic Partitions)
- Memory virtualization



Fixed Size Partitions

- Partition sizes are fixed (same size)
- Degree of multiprogramming
  - Equals number of partitions
- Size of partition
  - Too small: processes can not be executed
  - Too large: memory is wasted
- When a process exits:
  - Partition becomes free
  - Fragmented
  - Internal fragmentation



Internal Fragmentation

Green: Memory space occupied by process

Grey: Unused memory space

Internal Fragmentation: An Example

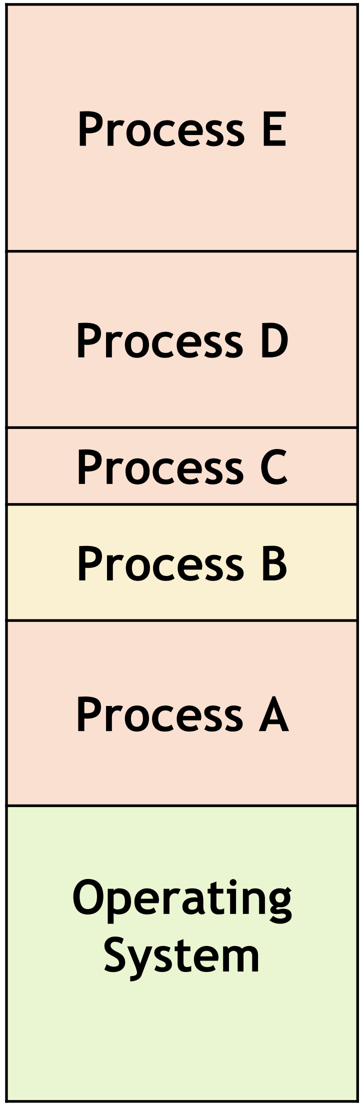
- Fixed Partition Size: 64 KB = 2^{16} bytes
- Allocated to a process of size: 58 KB
- Internal fragmentation = (64 – 58) KB = 6 KB
- Minimum Fragmentation = 1 byte
- Maximum Fragmentation = $(2^{16} – 1)$ bytes
- Average fragmentation = half the partition size
- Average fragmentation = 32 KB

Internal Fragmentation: An Example

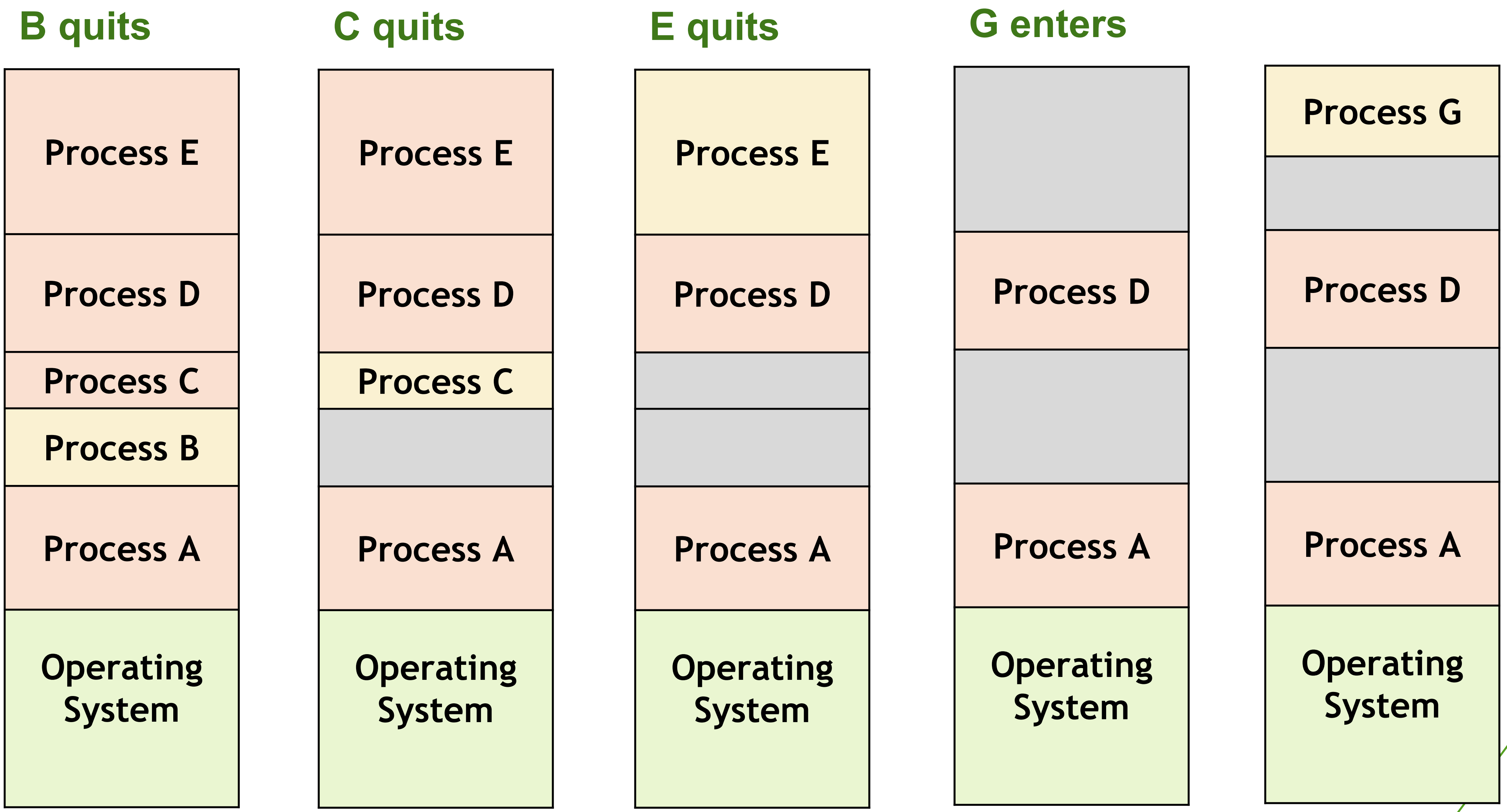
- Fixed Partition Size: 64 KB = 2^{16} bytes
- Memory size = 1 MB = 2^{20} bytes
- Memory allocated to OS = 512 KB = 2^{19} bytes
- Number of partitions = $(2^{20} - 2^{19}) / 2^{16} = 8$
- Process size (in KB): 62, 56, 42, 38, 60, 52, 48, 28
- Total Internal Fragmentation = (64-62)+ (64-56) + (64- 42) + (64- 38) + (64- 60) + (64- 52) + (64- 48) + (64- 28)
  = 2 + 8 + 22 + 26 + 4 + 12 + 16 + 36 = 126 KB
- Average Internal Fragmentation **= 15.75 KB**

Variable Size Partitions

- Partition sizes are variable
- Degree of multiprogramming
  - May vary
- Size of partition
  - Can be adjusted as per process needs
- When a process exits:
  - Partition becomes free (holes)
  - Fragmented: memory consists of allocated regions interspersed with holes



Variable Size Partitions



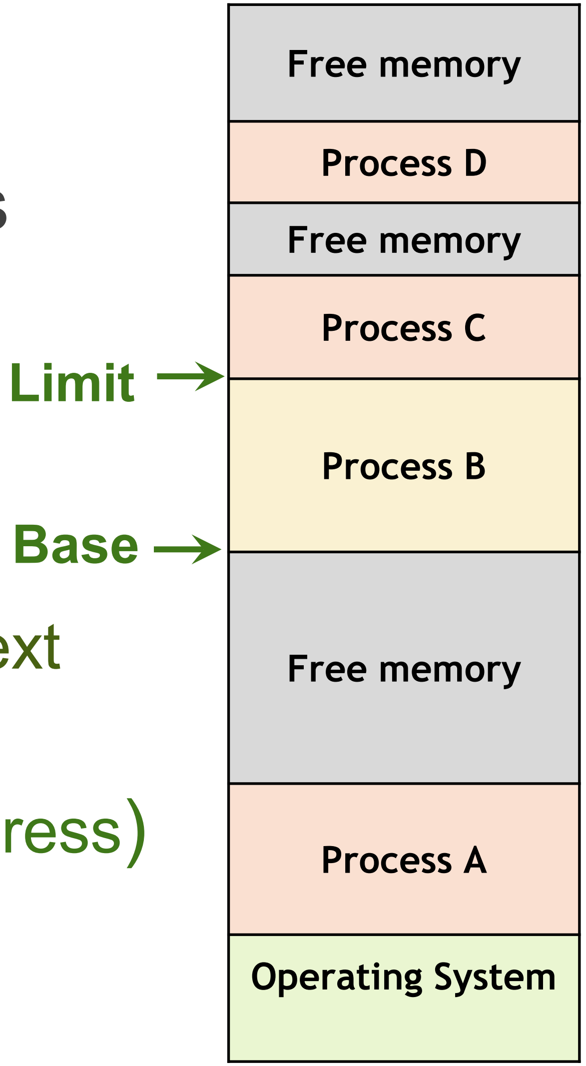
Fragmentation

- Variable Size partitioning leads to external fragmentation
- Internal Fragmentation
  - Unutilized memory of within partition
  - Characteristic of fixed size partitioning
- External Fragmentation
  - Wasted memory is outside partition
  - Happens in variable size partitioning

Hardware Support

- Multiple processes reside in memory
- Process may exit leading to free spaces
- Isolation between processes?
  - Hardware support needed
  - Base and limit registers used
  - Loaded with start and end address at context switch
- If $(\text{address} \leq \text{Content}_{\text{base}})$ or $(\text{Content}_{\text{limit}} \leq \text{address})$

  Segmentation fault



Partition Allocation

- First fit: allocate the first hole large enough to accommodate the process
- Best fit: allocate the hole that results in least fragmentation i.e. difference of hole size and process size if minimum
- Worst fit: allocate the largest hole (should be larger that the process size)
- Next fit: modified version of first fit but starts searching from last position and not from beginning

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 20, P_5: 40

72		34		68		24		56		48

- First Fit
- Best Fit
- Worst Fit
- Next Fit

First Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48

First Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48

40	P1		34		68		24		56		48

First Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48

40	P1		34		68		24		56		48

40	P1		34		P2		24		56		48

First Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48

40	P1		34		68		24		56		48

40	P1		34		P2		24		56		48

40	P1		34		P2		24		6	P3		48

First Fit

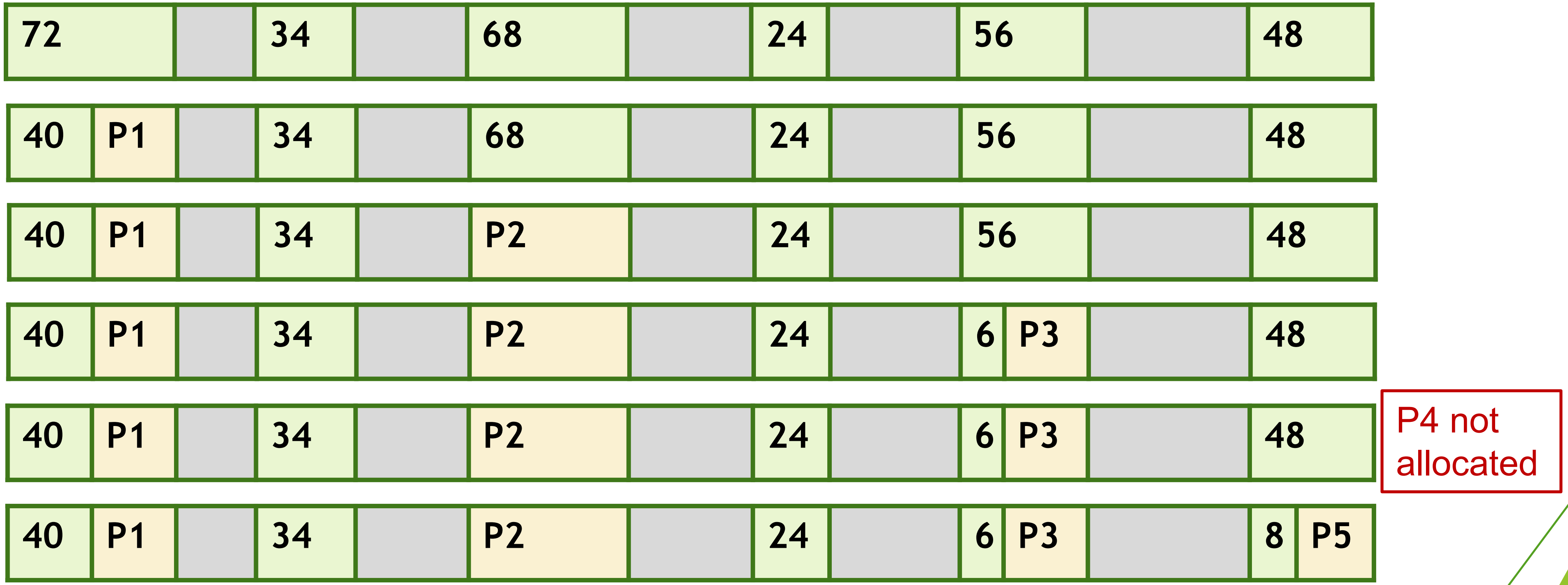
- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72			34		68		24		56			48
40	P1		34		68		24		56			48
40	P1		34		P2		24		56			48
40	P1		34		P2		24		6	P3		48
40	P1		34		P2		24		6	P3		48

P4 not allocated

First Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



First Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72			34		68		24		56			48		
40	P1		34		68		24		56			48		
40	P1		34		P2		24		56			48		
40	P1		34		P2		24		6	P3		48		
40	P1		34		P2		24		6	P3		48		P4 not allocated
40	P1		34		P2		24		6	P3		8	P5	

Best Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48

Best Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48

72		2	P1		68		24		56		48

Best Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34			68		24		56		48
72		2	P1		68		24		56		48
72		2	P1		P2		24		56		48

Best Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34			68		24		56			48
72		2	P1		68		24		56			48
72		2	P1		P2		24		56			48
70		2	P1		P2		24		6	P3		48

Best Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48
72		2 | P1		68		24		56		48
72		2 | P1		P2		24		56		48
70		2 | P1		P2		24		6 | P3		48
2 | P4		2 | P1		P2		24		6 | P3		48

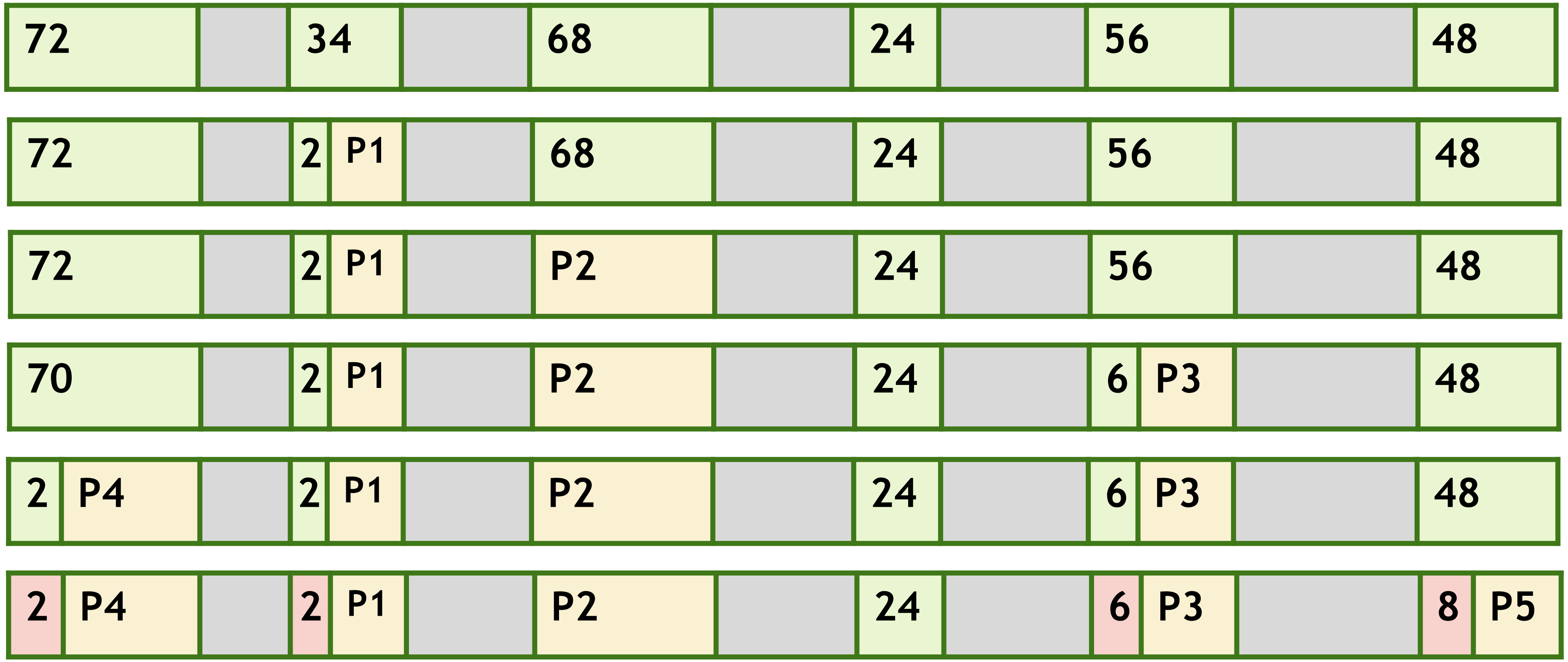
Best Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72			34			68		24		56			48	
72			2	P1		68		24		56			48	
72			2	P1		P2		24		56			48	
70			2	P1		P2		24		6	P3		48	
2	P4		2	P1		P2		24		6	P3		48	
2	P4		2	P1		P2		24		6	P3		8	P5

Best Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



Worst Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48

Worst Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48

40	P1		34		68		24		56		48

Worst Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48

40	P1		34		68		24		56		48

40	P1		34		P2		24		56		48

Worst Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72		34		68		24		56		48

40	P1		34		68		24		56		48

40	P1		34		P2		24		56		48

40	P1		34		P2		24		6	P3		48

Worst Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72			34		68		24		56			48
40	P1		34		68		24		56			48
40	P1		34		P2		24		56			48
40	P1		34		P2		24		6	P3		48
40	P1		34		P2		24		6	P3		48

P4 not allocated

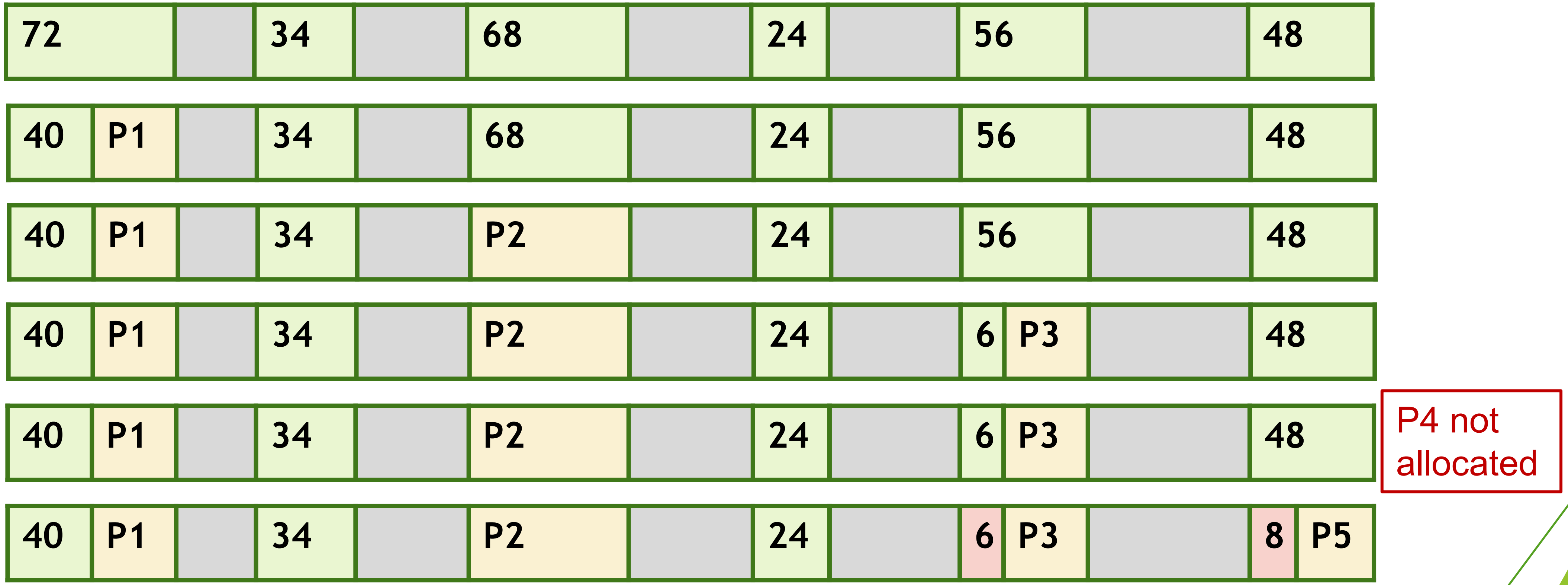
Worst Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

72			34		68		24		56			48		
40	P1		34		68		24		56			48		
40	P1		34		P2		24		56			48		
40	P1		34		P2		24		6	P3		48		
40	P1		34		P2		24		6	P3		48		P4 not allocated
40	P1		34		P2		24		6	P3		8	P5	

Worst Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

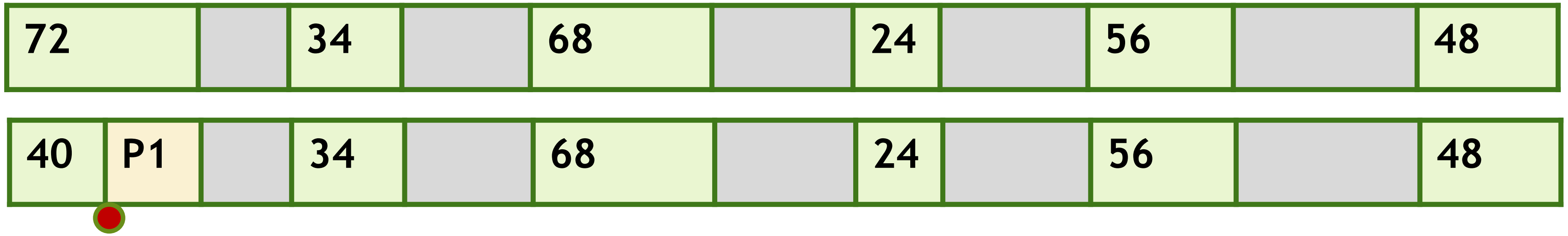


Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

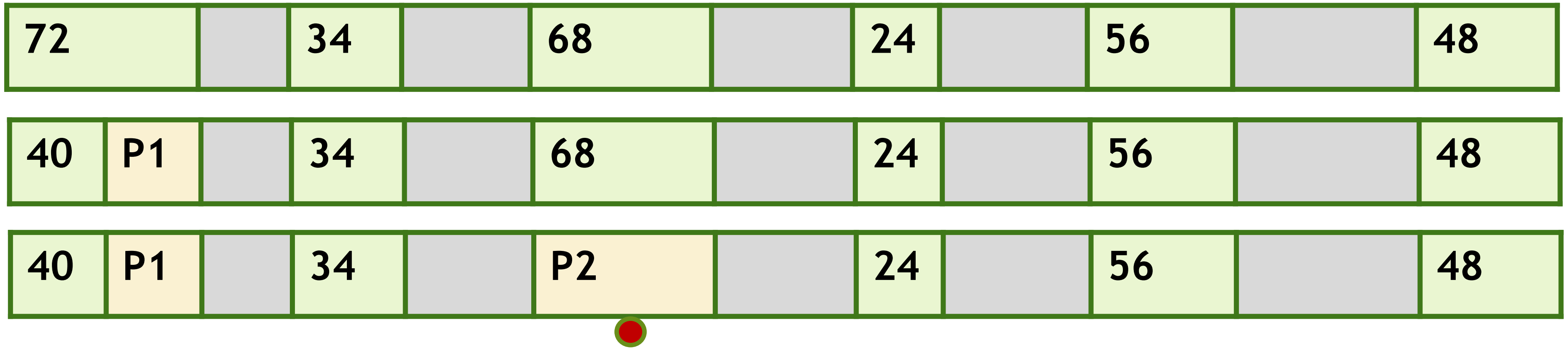
Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



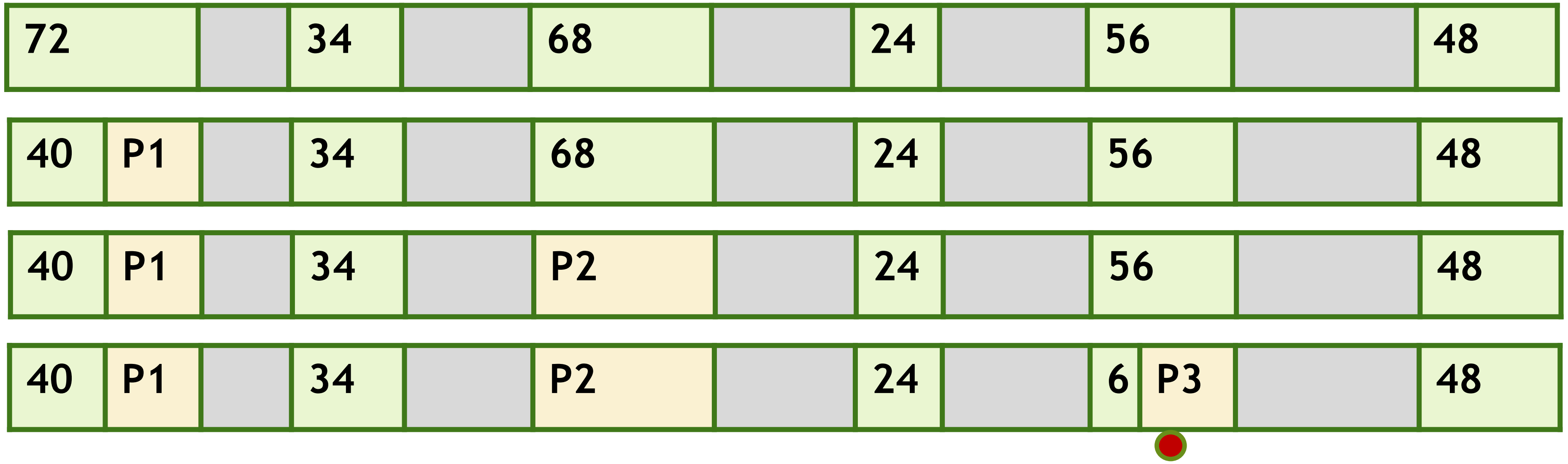
Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



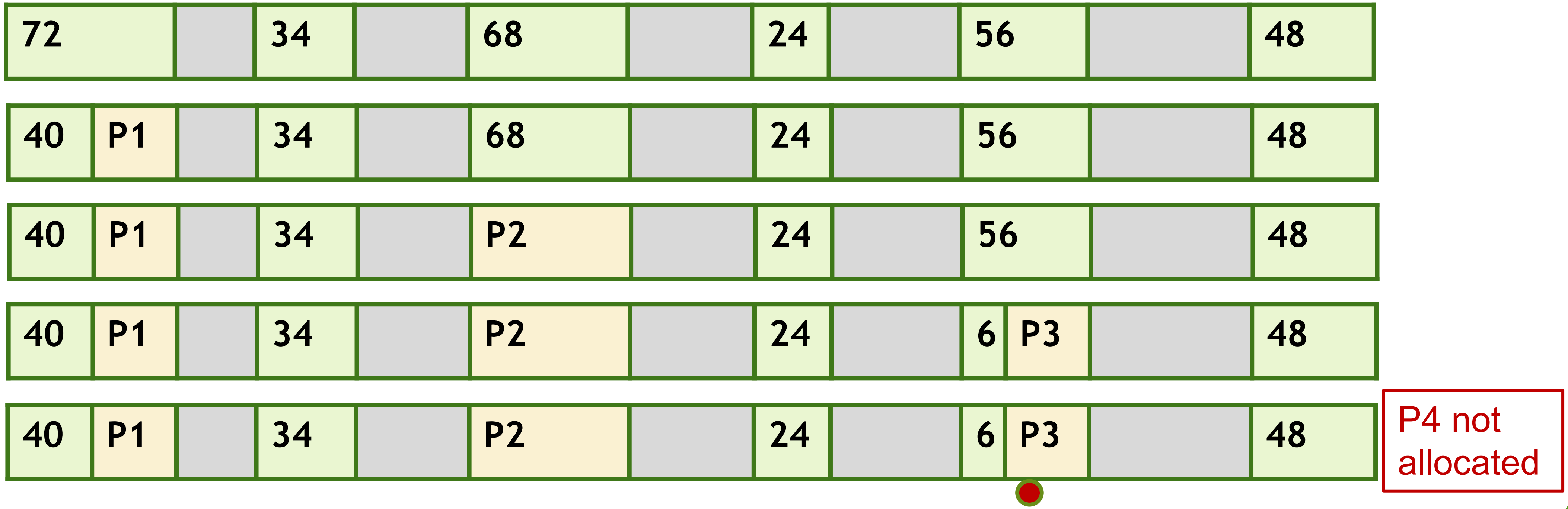
Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



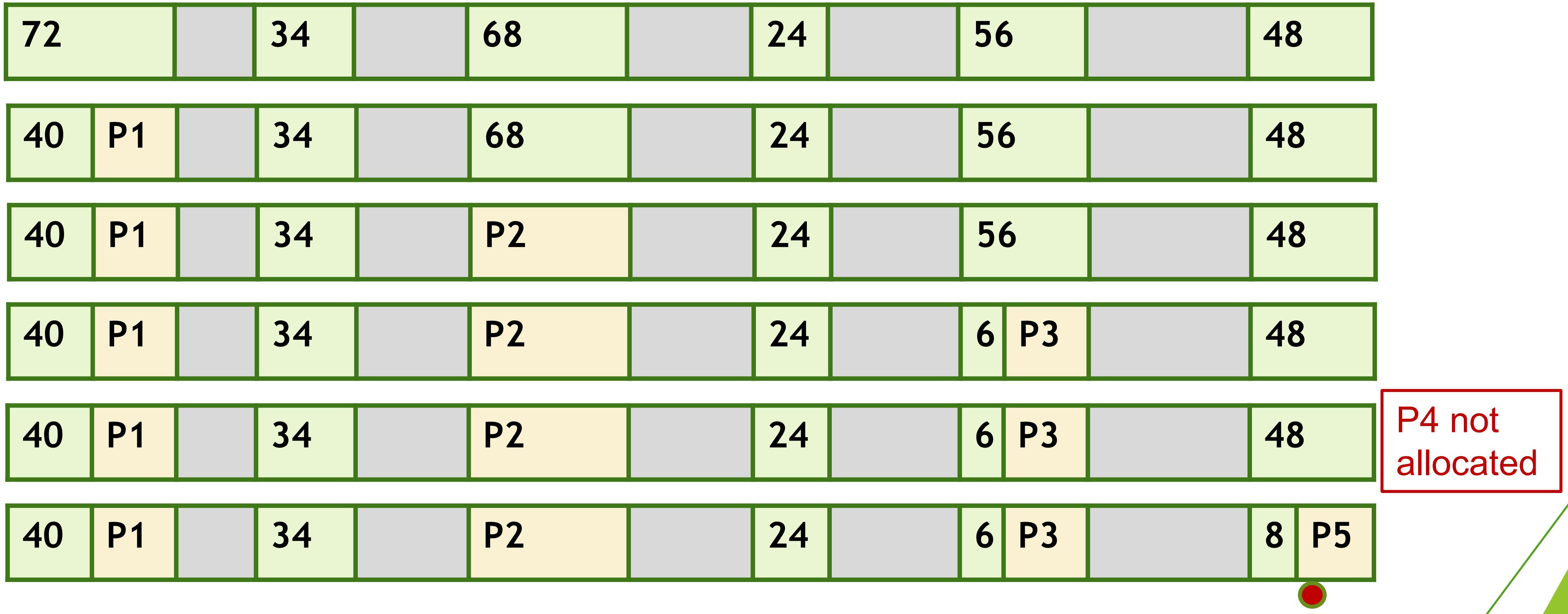
Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

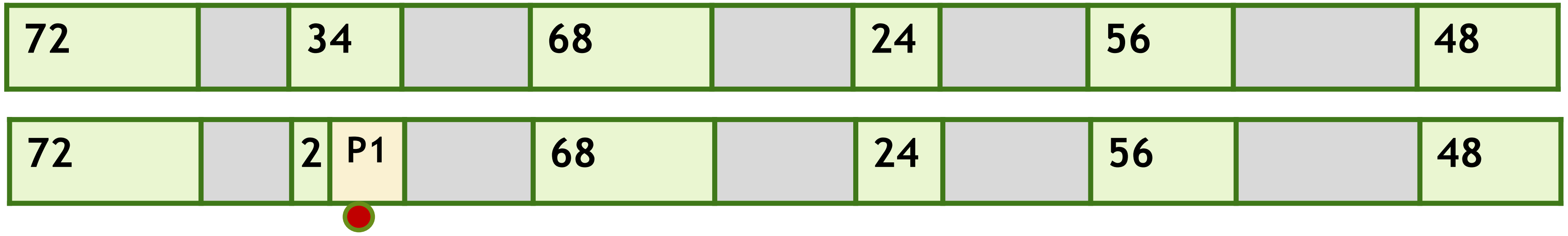
72		34		68		24		56		48				
40	P1		34		68		24		56		48			
40	P1		34		P2		24		56		48			
40	P1		34		P2		24		6	P3		48		
40	P1		34		P2		24		6	P3		48		P4 not allocated
40	P1		34		P2		24		6	P3		8	P5	

Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

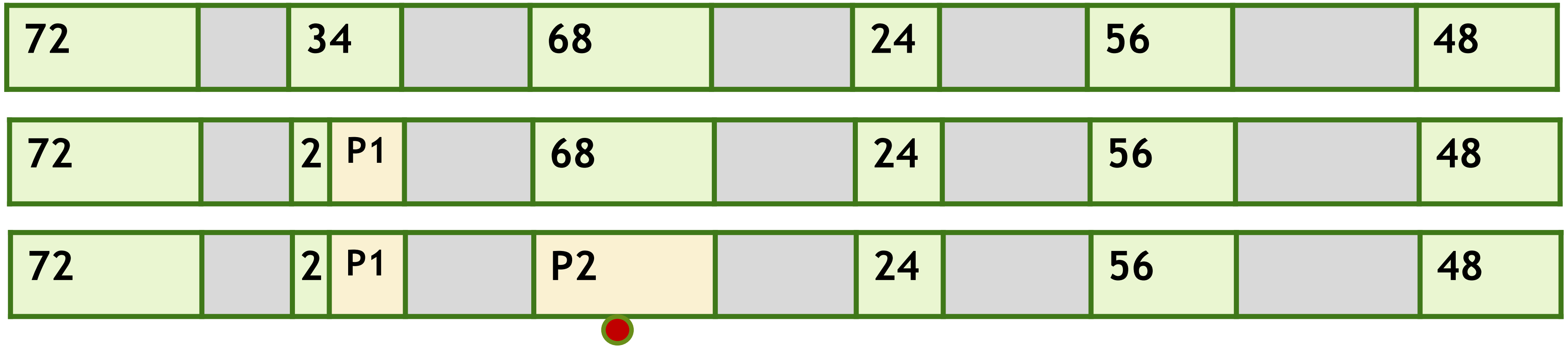
Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



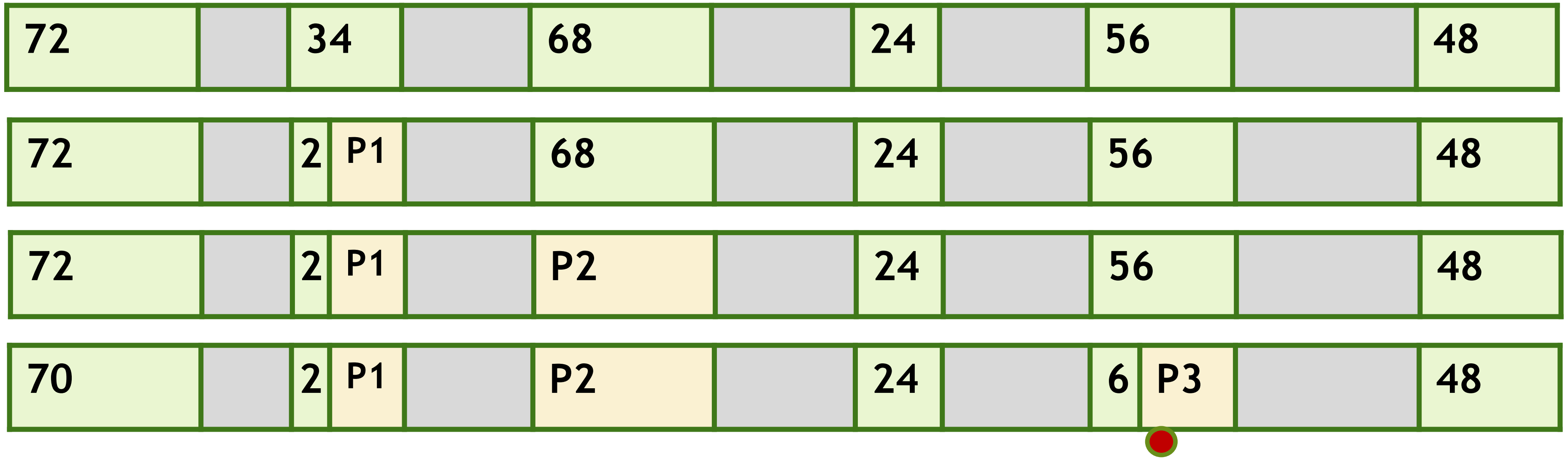
Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



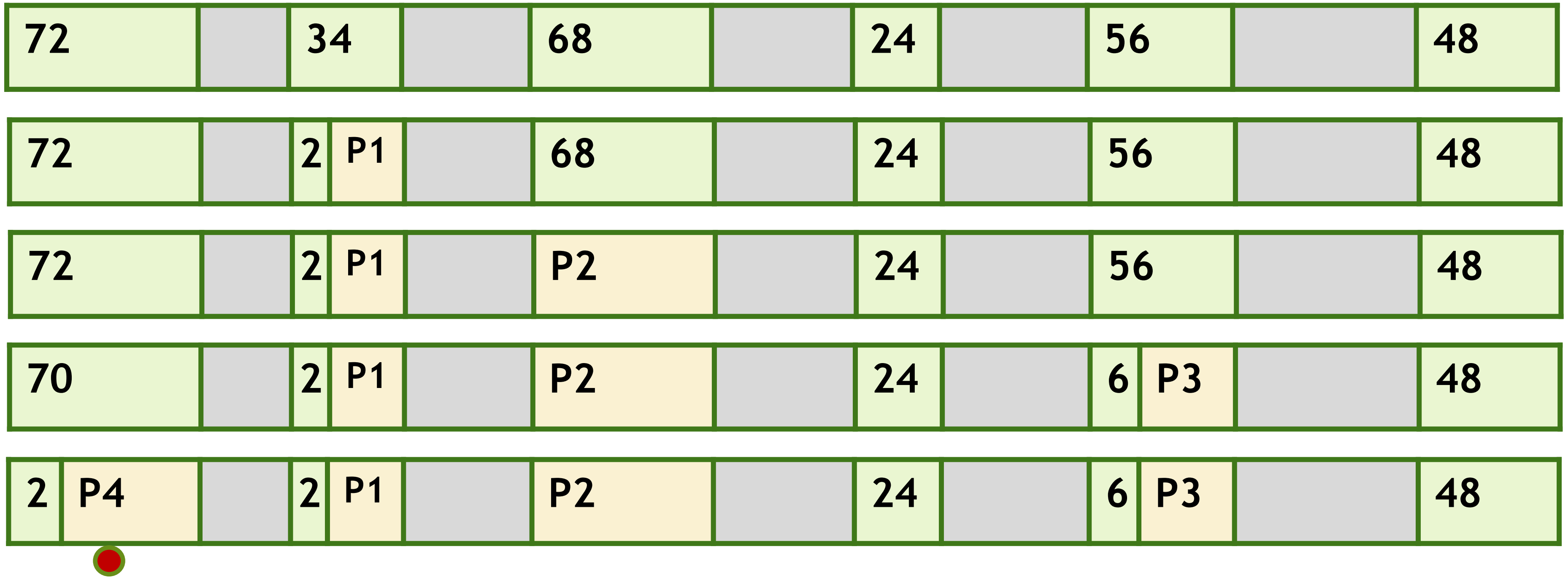
Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



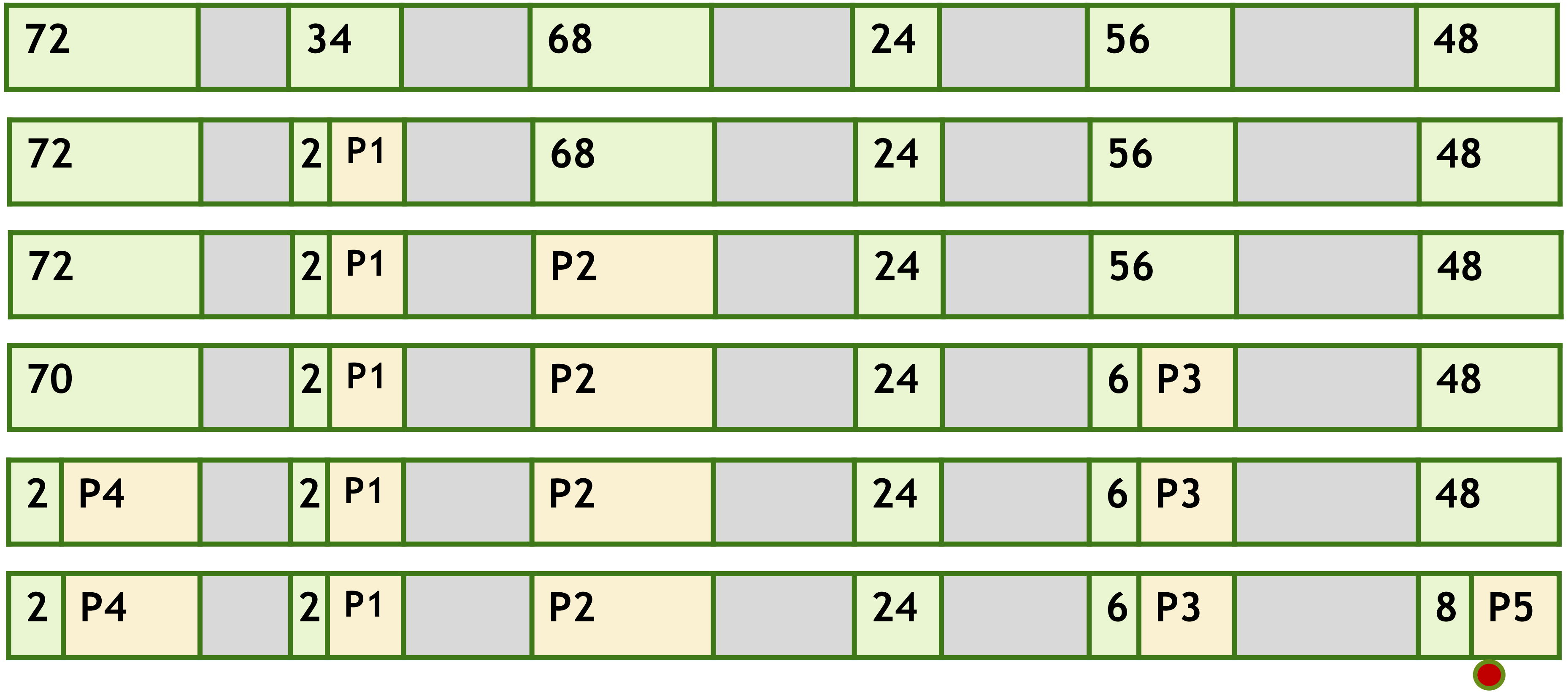
Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



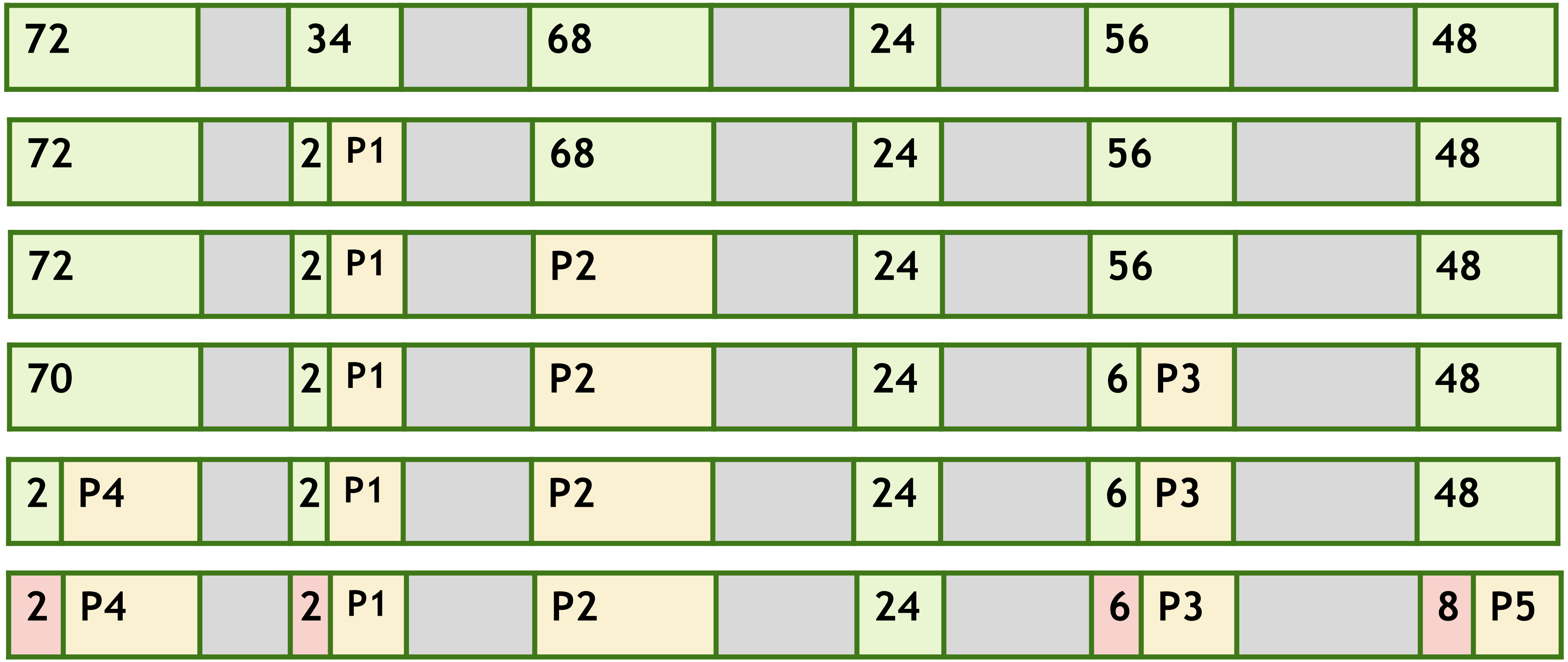
Next Fit

- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40

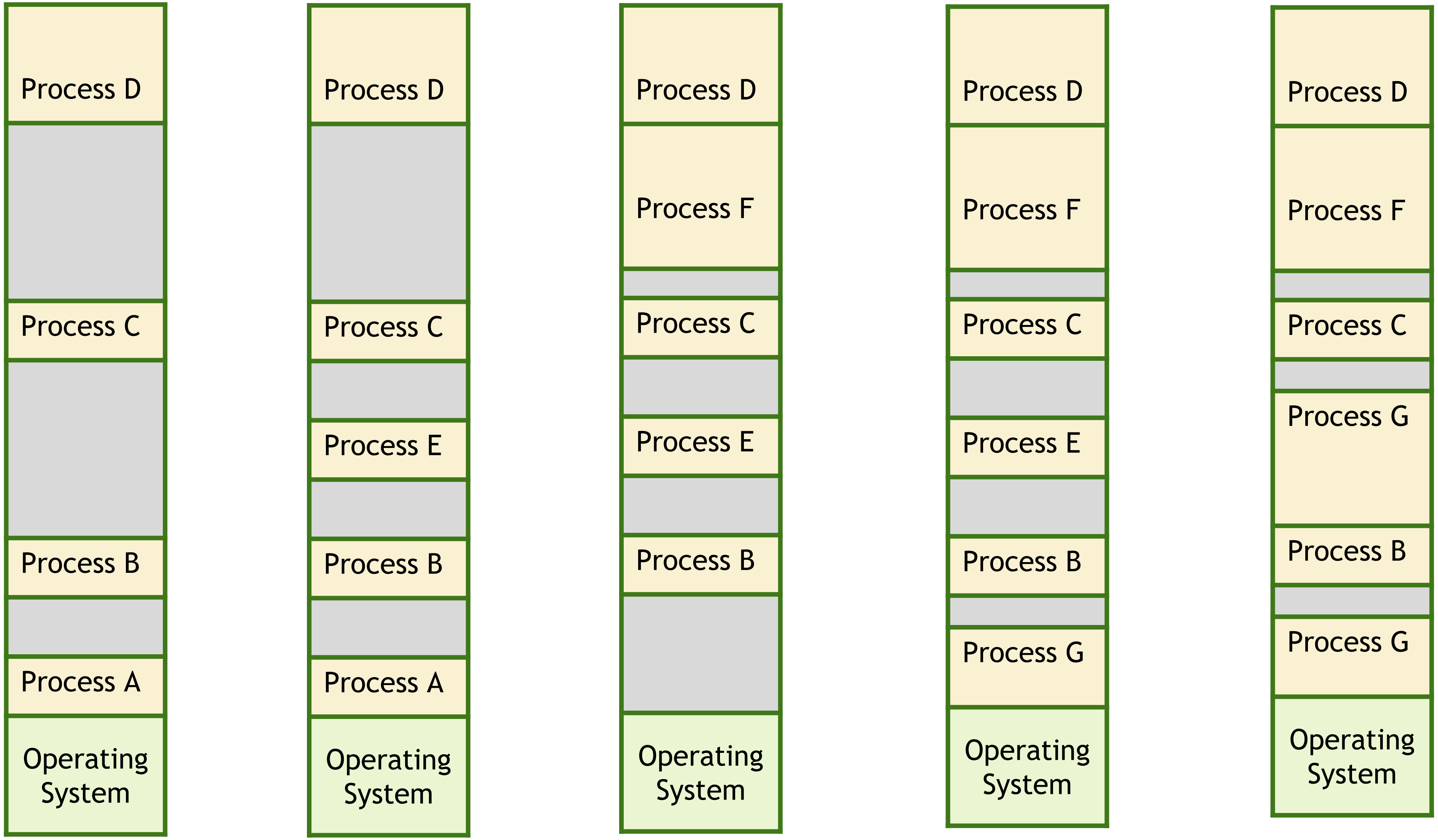


Next Fit

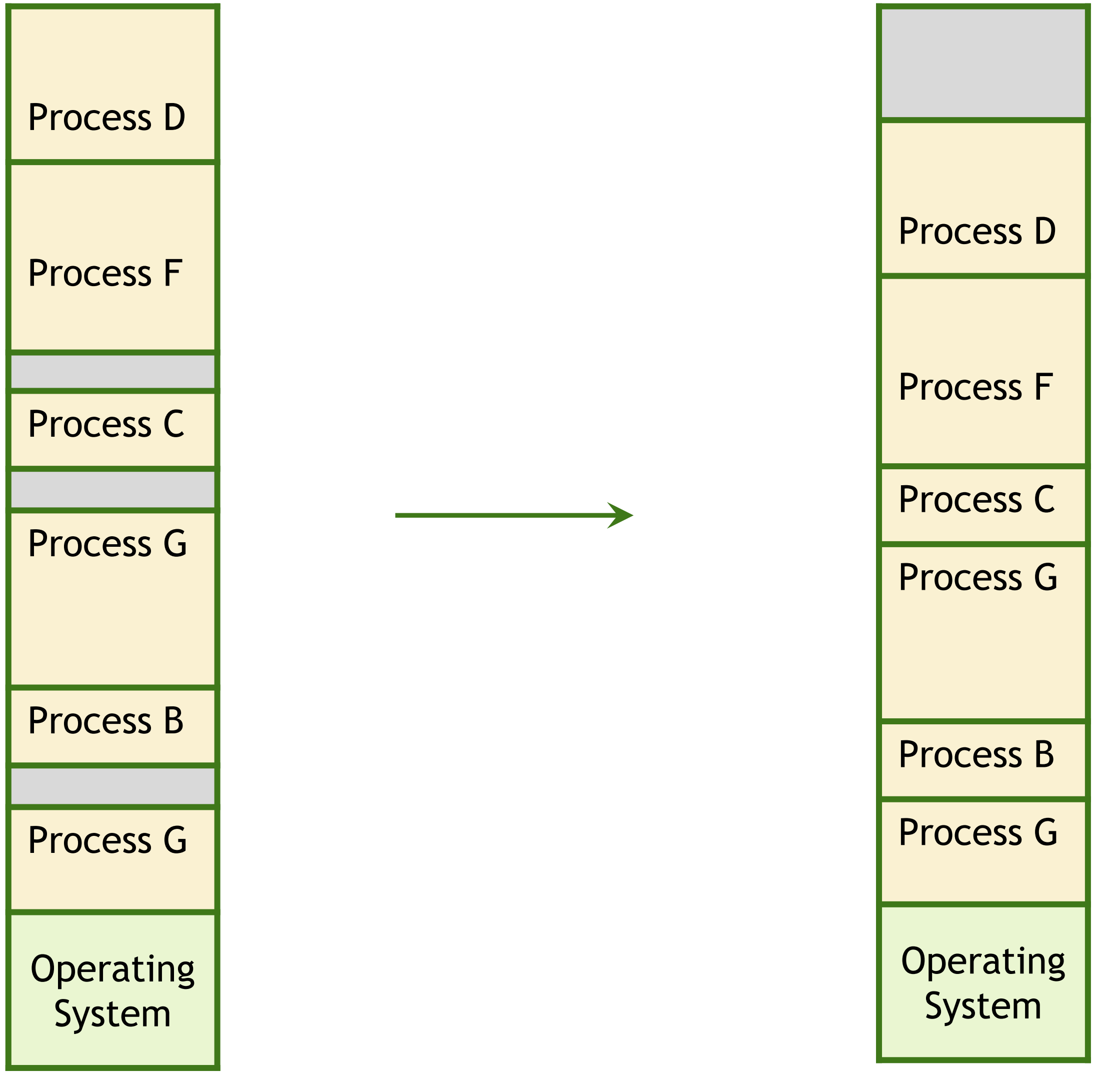
- Holes (KB): H_0: 72, H_1: 34, H_2: 68, H_3: 24, H_4: 56, H_5: 48
- Process size (KB): P_1: 32, P_2: 68, P_3: 50, P_4: 70, P_5: 40



Memory Compaction



Memory Compaction



Assignment: Memory Compaction

- Memory copy (location M1 to M2) requires following machine code execution

  MOV AX, M1

  MOV M2, AX

  If each machine instruction requires 20 ns, what shall be time required to copy 32 MB of data from one location to another?
- If CPU requires 16 ns to copy 32 bytes from one location to another, what would be rime required to compact a 64 GB memory?

Thank you.