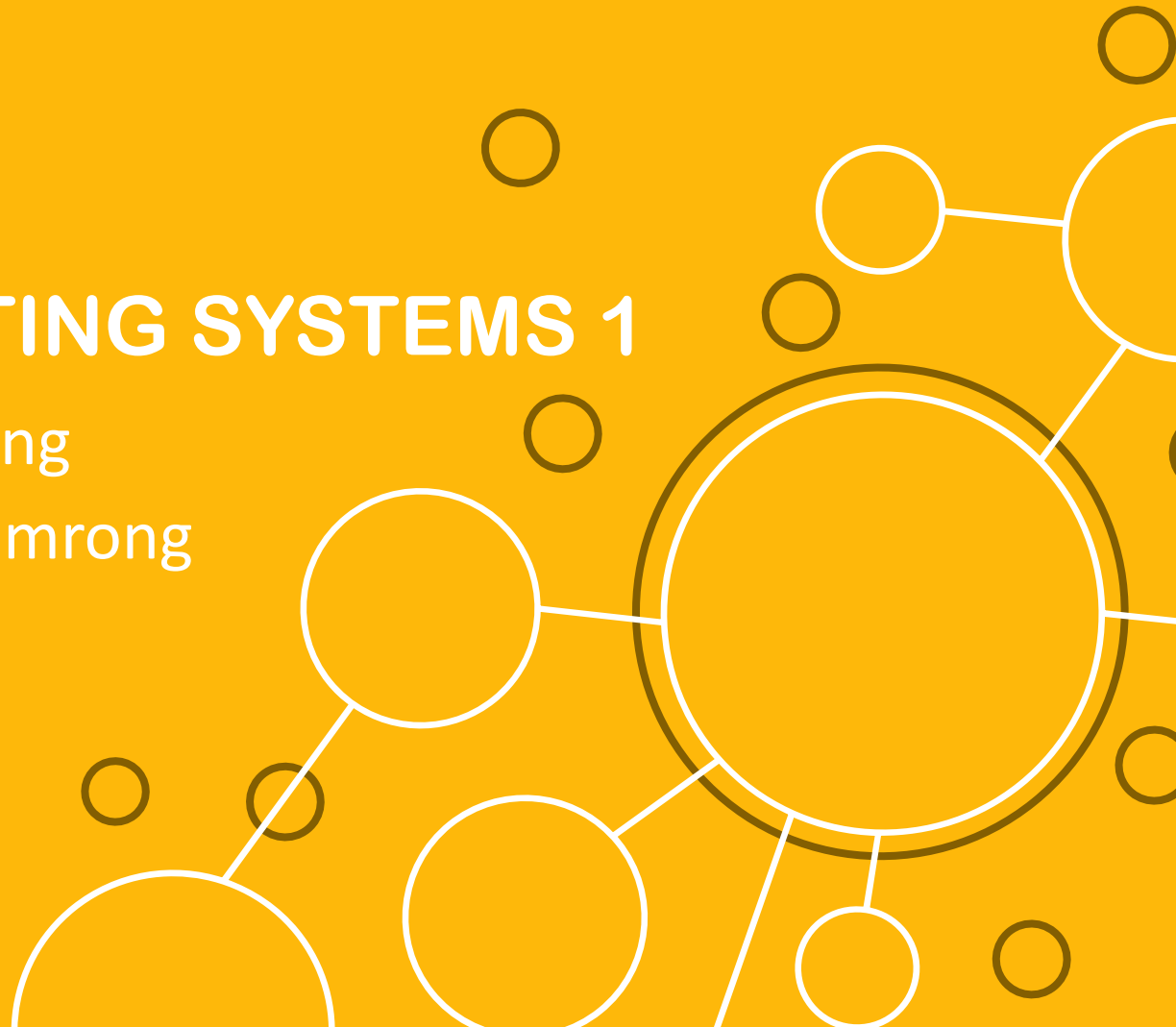
MEMORY MANAGEMENT (1)

CS341 OPERATING SYSTEMS 1

Wilawan Rakpakawong
Prapaporn Rattanatamrong

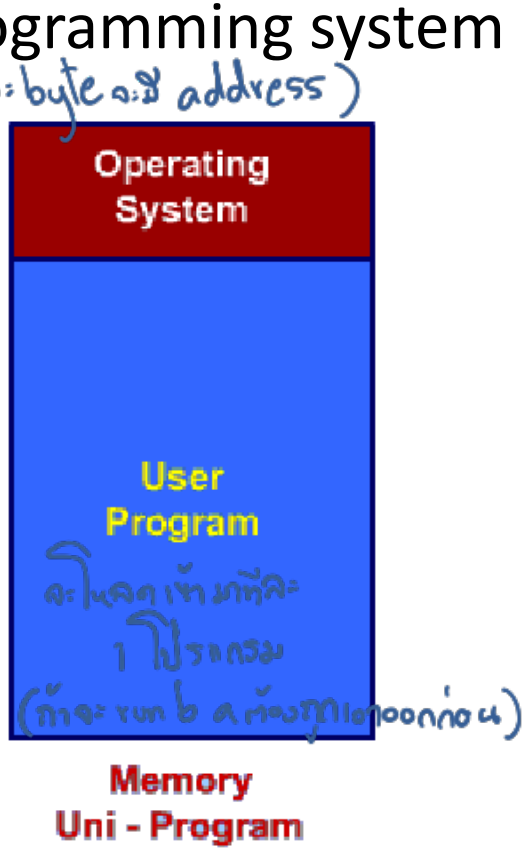


OUTLINE (หน้าที่ของ OS ในการจัดการ memory ทั้งหมด)

- Functions of memory management
- Single contiguous memory model (ของแต่ละโปรแกรมสำเป็นก้อนเดียวกันต่อเนื่องกัน)
  - Memory partitioning
  - Process Relocation
  - Memory protection

Role of Memory Management

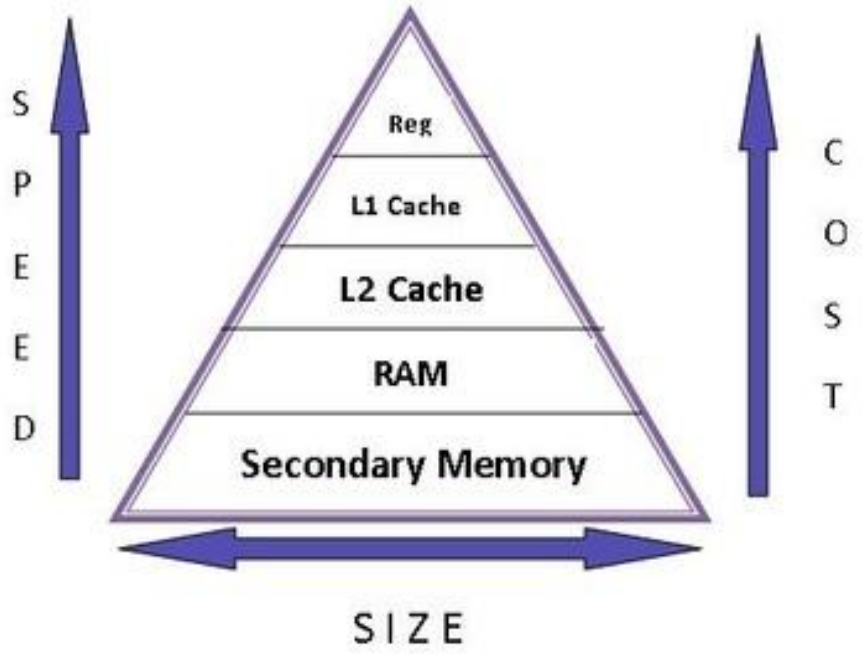
- Uniprogramming vs Multiprogramming
- Effective memory management is very important in a multiprogramming system



Functions of Memory Management

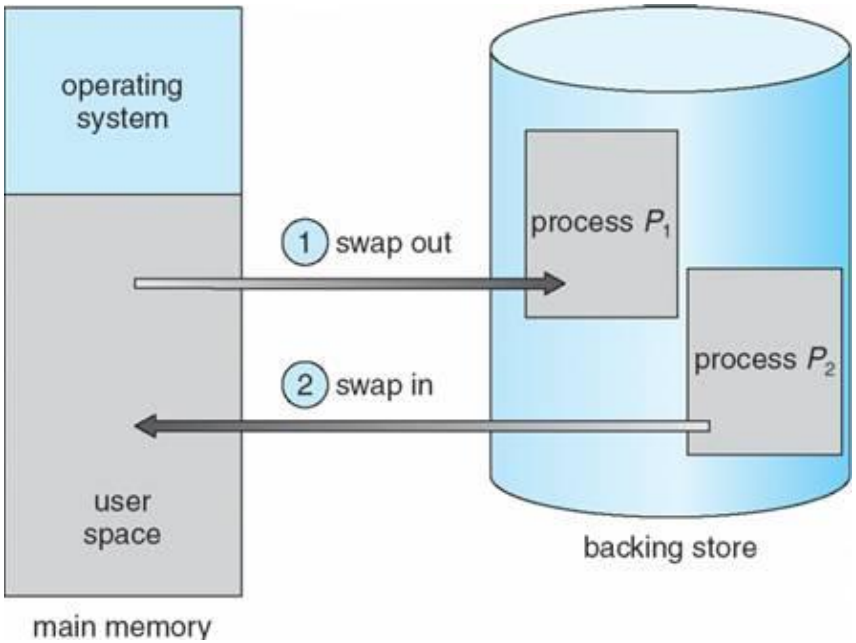
- Physical Organization
- Relocation
- Logical Organization
- Protection
- Sharing

Physical Organization



- Computer memory is organized into at least two levels:
  - Main memory
  - Secondary memory
- The task of moving information between the two levels of memory should be a system responsibility

RELOCATION



- Programmers typically do not know which other programs will co-locate in memory with their programs

Inflexible!
(ไม่ยืดหยุ่น)

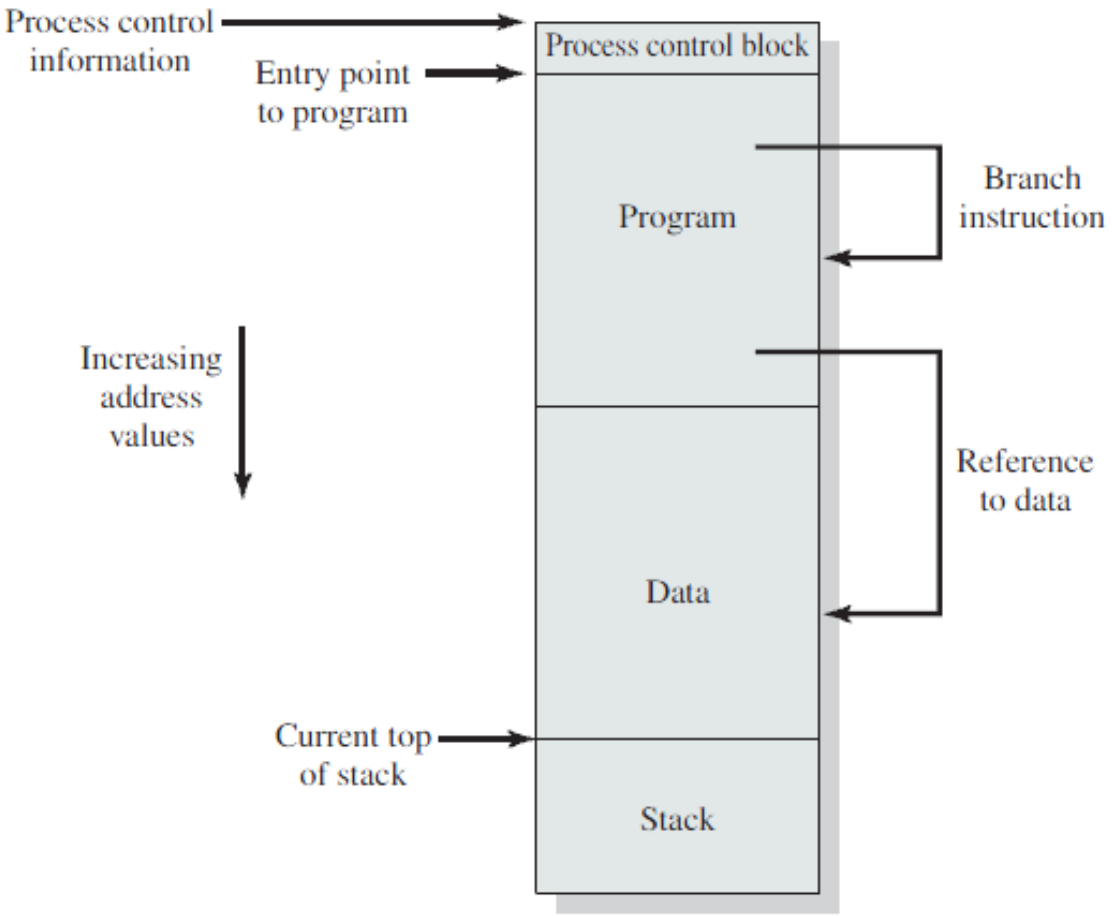
- Processes need to be swapped in and out to maximize processor utilization
  - Place in the same memory region as before
  - Relocate processes to different areas of memory

Logical Organization

- ช่วยในการ Relocation

- Main memory and second memory in a computer are organized as a linear or one-dimensional address space consisting of a sequence of bytes or words
- However, most programs are organized into modules
- Modules can be written and compiled independently
- Different degrees of protection may be needed by different modules
- Modules can be shared among processes

Addressing Issue

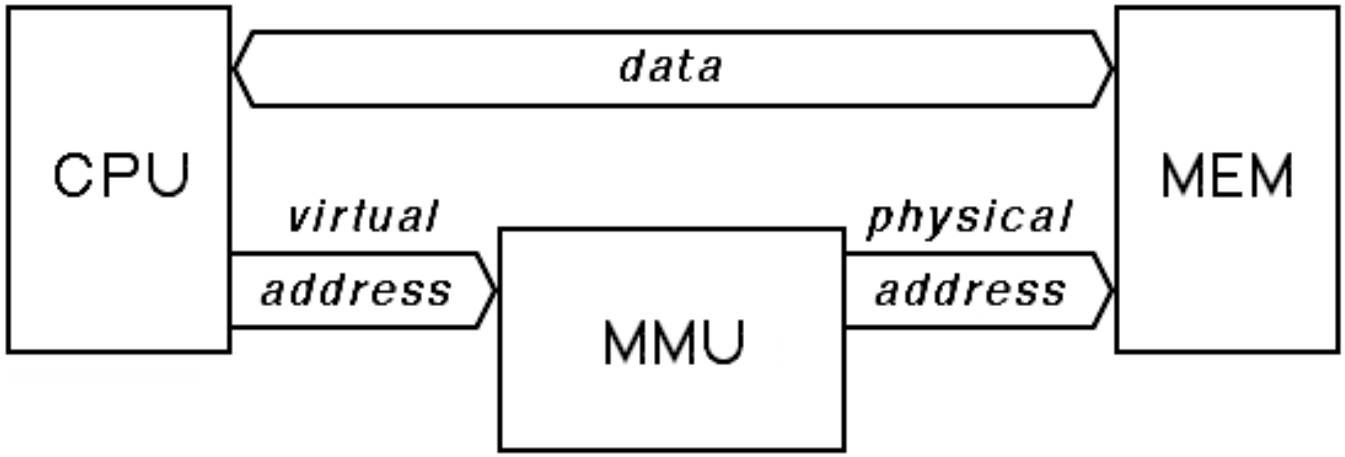


- The OS and processor must be able to translate the memory references found in the code into actual physical memory addresses

Memory-Management Unit (MMU)

หน้าที่: รับ visual address และข้อมูลเข้ามาแปลงเป็น physical address

- Hardware device that maps virtual to physical address at run time
- Many methods to perform the mapping are possible, covered in the rest of this chapter



PROTECTION

ป้องกันไม่ให้เข้าส่วนของ process อื่นที่ไม่ได้ใช้ร่วมกัน

- Each process should be protected against unwanted interference by other processes (accidental or intentional)
- All memory references generated by a process must be checked at run time to ensure protection
- Memory protection requirement must be satisfied by the processor (hardware) rather than the OS (software)

SHARING

process 2 process ต้องการเข้าถึงพื้นที่เดียวกัน OS ต้องอนุญาตให้เข้าถึง memory ที่ตำแหน่งเดียวกัน

- Any protection mechanism must have the flexibility to allow several processes to access the same portion of main memory
  - Running a number of processes executing the same program
  - Processes cooperate on some task accessing the same data structure
- We need controlled access to shared areas of memory without compromising essential protection

Memory Partitioning Techniques

process ไม่สามารถแยกส่วนกันได้ (ต้องเป็นก้อนเดียวต่อเนื่องกัน)

- The principal operation of memory management is to bring processes into main memory for execution by the processor (a.k.a. **memory partitioning**)
- Memory partitioning techniques:
  - Fixed partitioning
  - Dynamic partitioning
  - Buddy system
  - Paging
  - Segmentation
  - **Virtual memory paging**
  - **Virtual memory segmentation**

FIXED PARTITIONING

The Concept of Fixed Partitioning

- The assumptions:
  - The OS occupies some fixed portion of main memory
  - The rest of main memory is available for use by multiple processes
- The simplest scheme for managing this available memory is to partition it into regions with fixed boundaries
- Examples: Early IBM mainframe OS, OS/MFT (Multiprogramming with a Fixed number of Tasks)

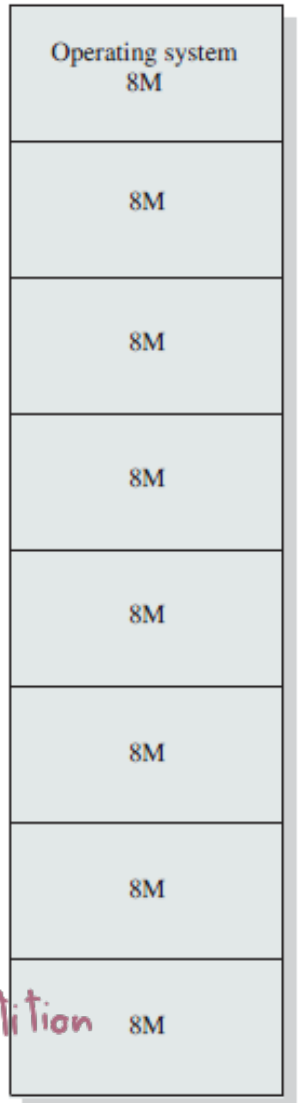
EQUAL-SIZE PARTITION (1)

ทุก partition มีขนาดเท่ากัน (หลายไม้ต้องใช้ทั้งก้อน

- Equal-size partitions
  - Any process whose size is less than or equal to the partition size can be loaded into any available partition
  - If all partitions are full and no process is in the ready or running state, the OS can swap a process out of the partitions and load in another process

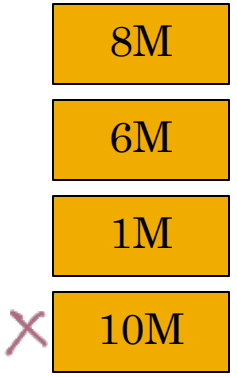
ข้อดี : ง่าย/เร็ว

ข้อเสีย : ไม่รองรับ process ที่ใหญ่กว่า partition



(a) Equal-size partitions

Can we load these processes in main memory?

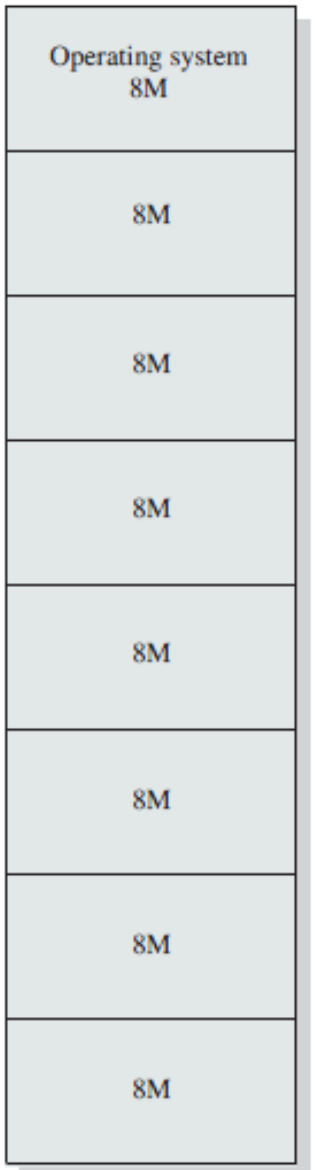


เพราะเกิน partition ที่มีทั้งหมด

EQUAL-SIZE PARTITION (2)

ถ้าเล็ก ขนาด partition

- Two difficulties with equal-size partitions
  - A program may be too big to fit into a partition, need overlay (more than one partition for one process)
  - Main memory utilization is extremely inefficient
    - The block of data loaded is smaller than the partition
      → internal fragmentation
      พื้นที่เหลือที่ไม่ถูกใช้งาน



(a) Equal-size partitions

8M

6M

1M

10M

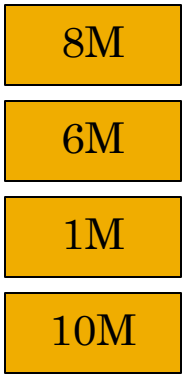
UNEQUAL-SIZE PARTITON (1)

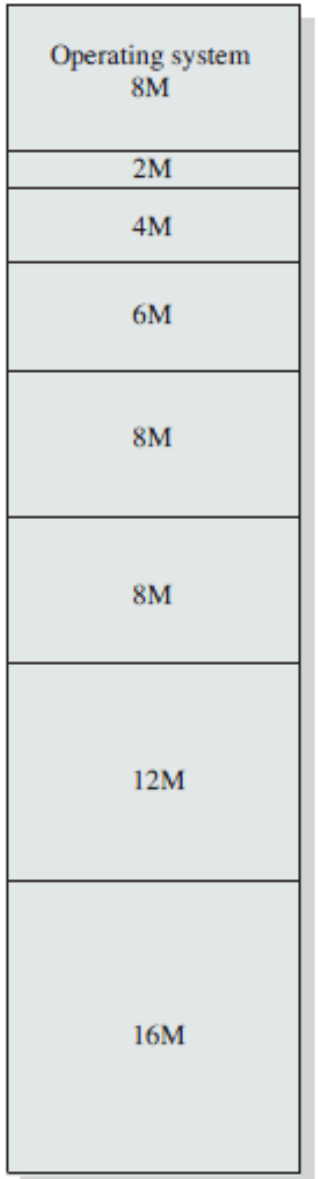
แต่ละ partition อาจมีไม่เท่ากัน แต่ไม่สามารถเปลี่ยน size ได้ (Fix)

- Unequal-size partitions
  - Memory is partitioned into chunks with different sizes ranging from small to large

เมื่อมีขนาดยืดหยุ่นมากขึ้น ก็จะมีความยากในการ manage มากขึ้น

Can we load these processes in main memory?



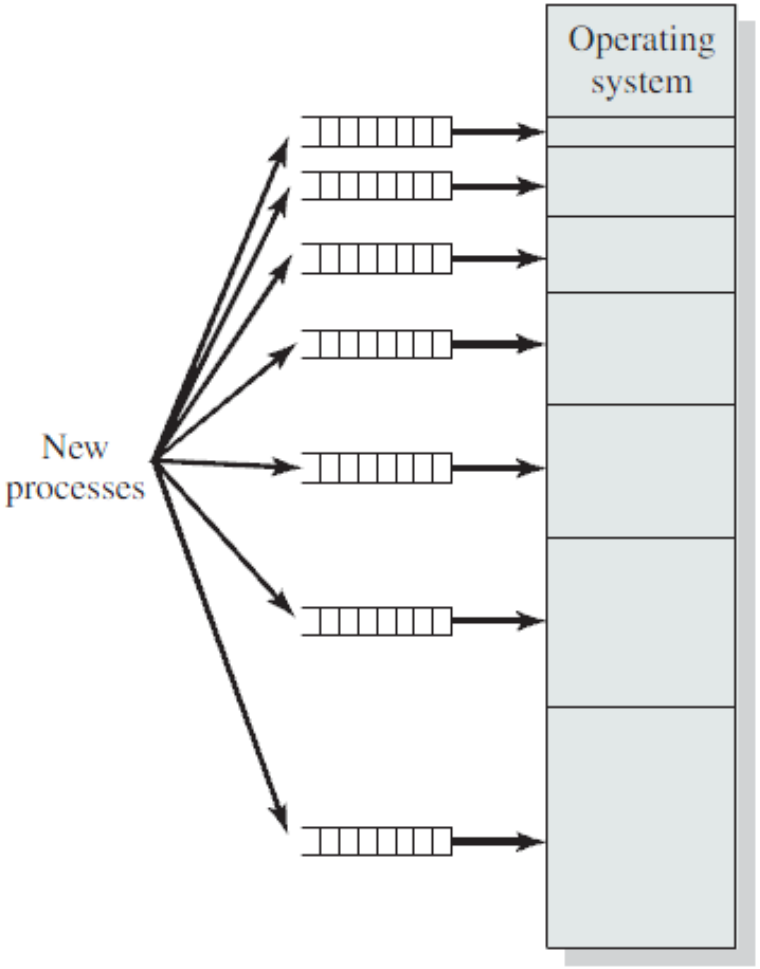


(b) Unequal-size partitions

UNEQUAL-SIZE PARTITION (2)

มองว่ามี queue สำหรับแต่ละขนาดไปเลย

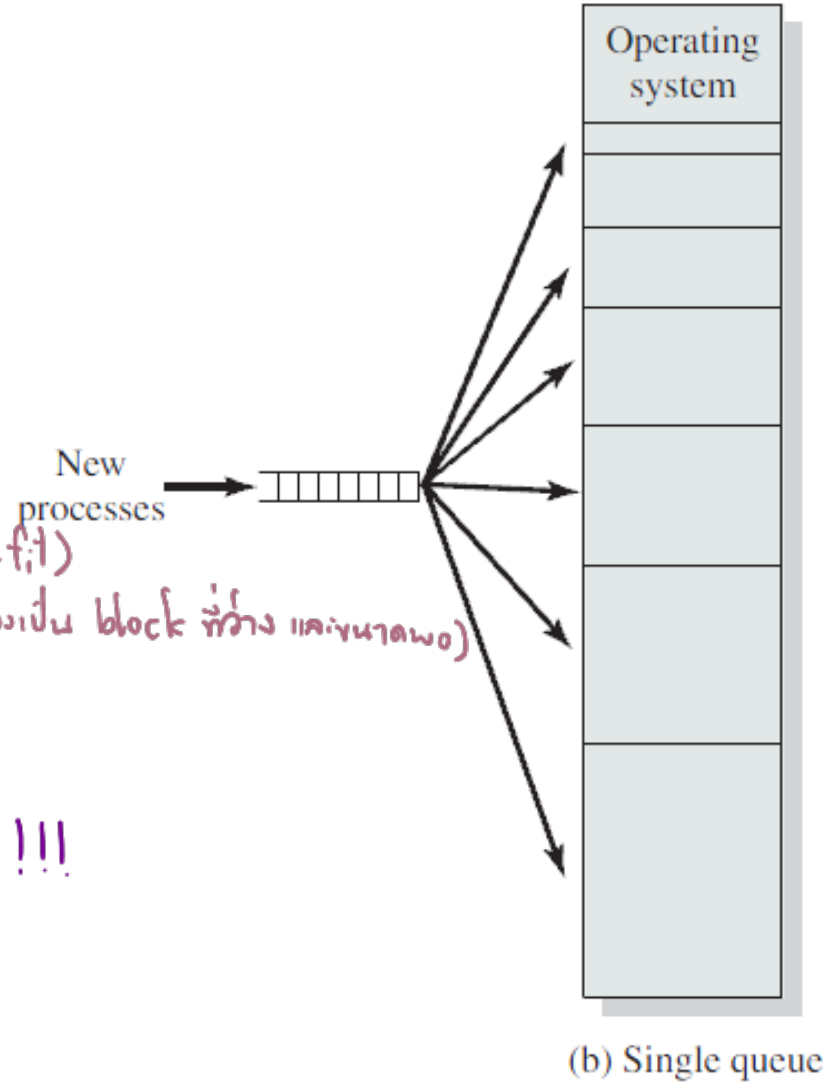
- One process queue per partition
  - Assign to the smallest partition within which it will fit
    - Pros: min internal fragmentation
    - Cons: some partitions may left unused even though some process could have been assigned to it



(a) One process queue per partition

PLACEMENT ALGORITHM

- Single queue for all partitions
  - Best-fit chooses available partition that is closest in size to the request
  - First-fit begins to scan memory from the beginning and chooses the first available partition that is large enough
  - Next-fit begins to scan memory from the location of the last placement, and chooses the next available partition that is large enough



(b) Single queue

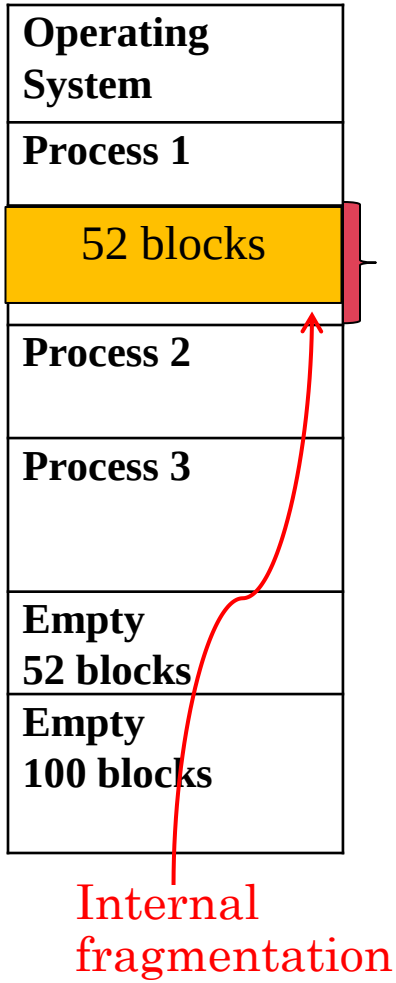
EXERCISE #1

Operating System
Process 1
Empty 60 blocks
Process 2
Process 3
Empty 52 blocks
Empty 100 blocks

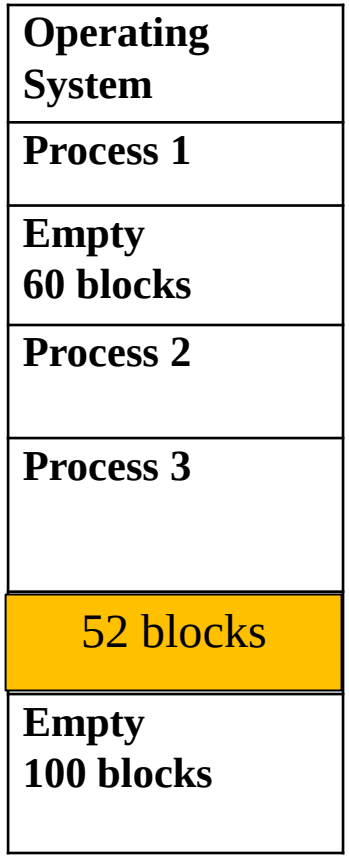
- If we use fixed partitioning and a new process requires 52 blocks of main memory space, show the memory configuration after using each of the following placement algorithms:
  - First available partition that will hold the process (first fit)
  - Smallest partition that will fit (best fit)

ANSWERS #1

- Fixed partition w/ first available partition that will hold the process (**First fit**)



- Fixed partition w/ smallest partition that will fit (**Best fit**)



Next-fit
ก็เข้าช่องนี้เลยครับ เป็นช่องว่างแรกต่อจาก process สุดท้ายที่เข้า

Fixed Partitioning

- Advantages
  - Provide a degree of flexibility via the use of unequal-size partitions
  - Relatively simple and require minimal OS and processing overhead
- Disadvantages
  - The number of partitions limits the number of active processes in the system
  - Small jobs will not utilize partition space efficiently
  - Not reasonable where the memory requirement of processes is not known beforehand

Dynamic Partitioning

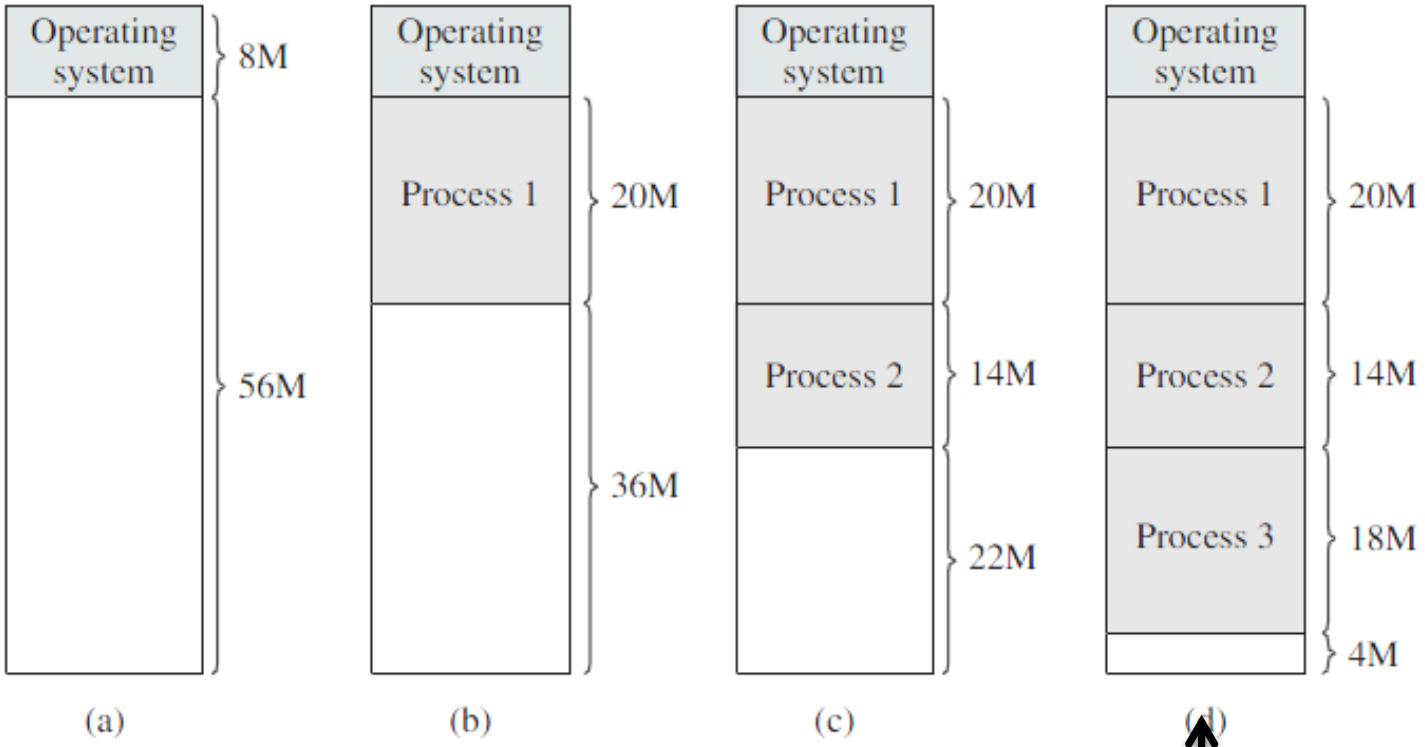
The Concept

หมายถึง process สามารถเข้าไปอยู่ partition ในหน่วยความจำได้ที่ขนาดใดขนาดหนึ่ง และ OS จะให้พื้นที่แค่เท่าที่ขอ (สามารถเปลี่ยนขนาด partition ได้)

- The partitions are of variable length and number
- When a process is brought into main memory, it is allocated exactly as much memory as it requires and no more
- Examples: Early IBM mainframe OS, OS/MVT (Multiprogramming with a Variable number of Tasks)

EXAMPLE

- Loading in 4 processes: 20M, 14M,18M, 8M

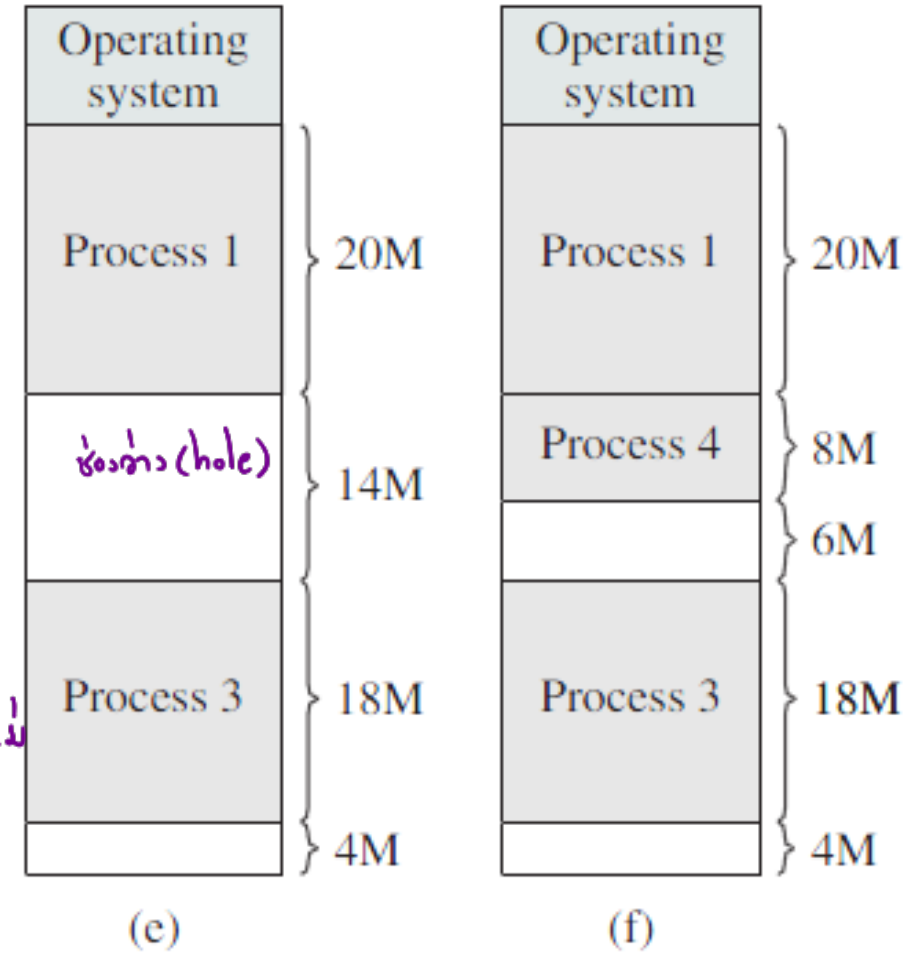


The remaining space is too small to fit in Process 4

EXAMPLE (CONT.)

- At some point, none of the processes in memory is ready
  - The OS swaps out Process 2
  - Now we have sufficient room to load in Process 4

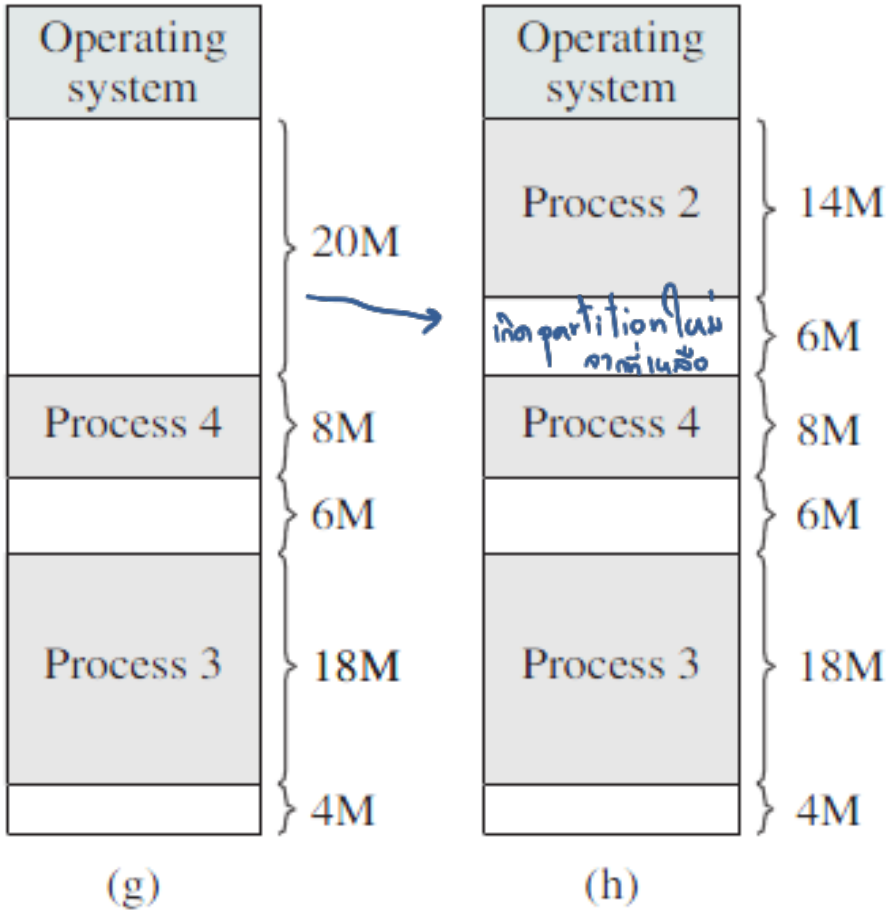
พอได้เท่าที่ขอแล้วที่เหลือจะกลายเป็น partition ใหม่



EXAMPLE (CONT.)

- Later, none of the processes in memory is ready, but Process 2 is in ready/suspend state
  - The OS swaps out Process 1
  - Then it swap in Process 2

รอจัด Relocation



worth-fit : ใช้อันที่เหลือพื้นที่มากที่สุด เพื่อใช้รองรับ process ในอนาคตได้มากกว่า

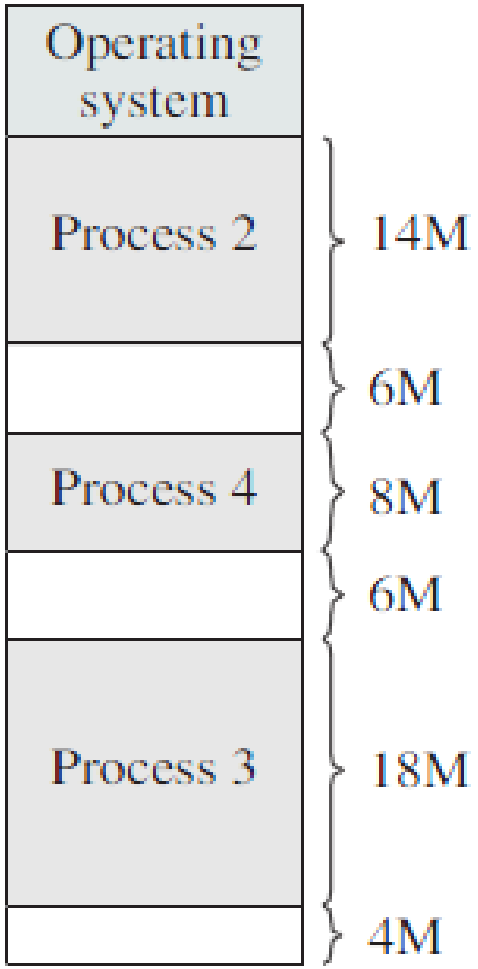
External Fragmentation

พื้นที่ที่ไม่ถูกใช้จากการใส่ partition ไม่พอกับขนาด process

- External fragmentation indicates that the memory that is external to all partitions becomes increasingly fragmented (a lot of small holes in memory)

อาจเกิดการสลับที่ process และ merge

- Overcoming external fragmentation using compaction
  - The OS shifts the processes so that they are contiguous and so that all of the free memory is together in one block
  - Time consuming and wasteful of processor time



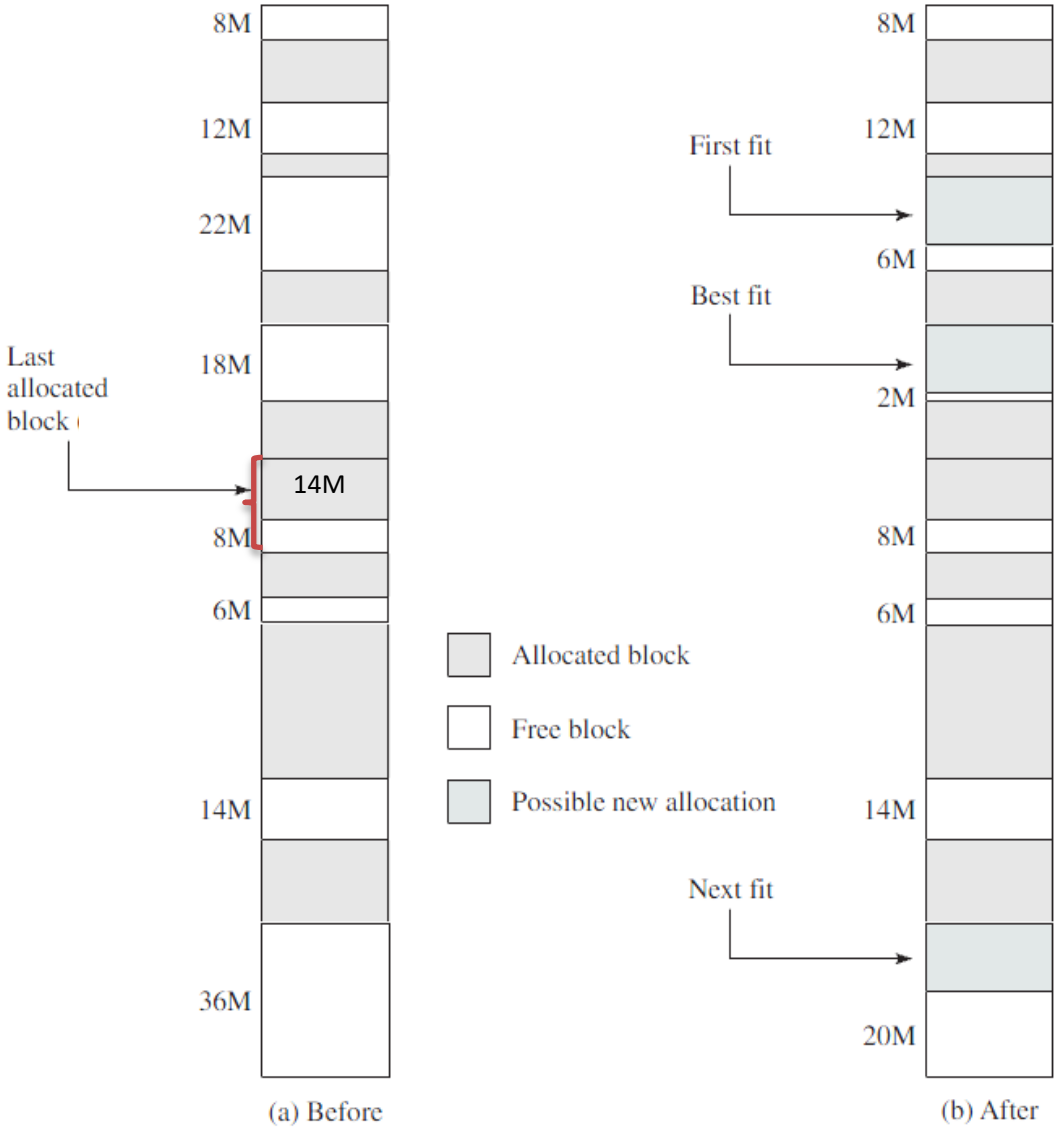
(h)

Placement Algorithms

- Similar to unequal-size partition of fixed partitioning
- Best-fit chooses the block that is closest in size to the request
- First-fit begins to scan memory from the beginning and chooses the first available block that is large enough
- Next-fit begins to scan memory from the location of the last placement, and chooses the next available block that is large enough
- All, of course, are limited to choosing among free blocks of main memory whose sizes are equal or larger than the process to be brought in

EXAMPLE

- The last block that was used was 22M block from which 14M partition was created
- The right figure shows the difference between best-, first- and next-fit in satisfying a 16M allocation request



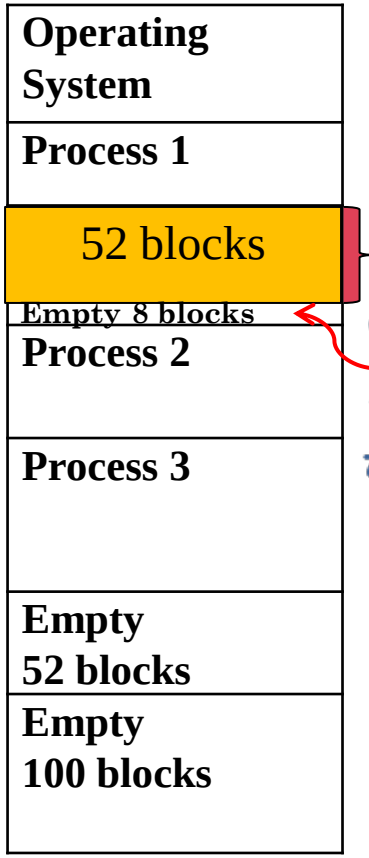
Exercise #2

Operating System
Process 1
Empty 60 blocks
Process 2
Process 3
Empty 52 blocks
Empty 100 blocks

- If we use dynamic partitioning and a new process requires 52 blocks of main memory space, show the memory configuration after using each of the following placement algorithms:
  - First fit
  - Best fit
  - Worst fit

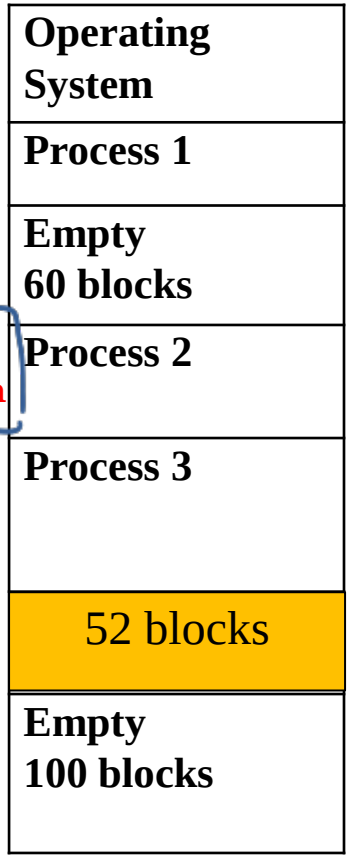
ANSWERS #2

- First fit

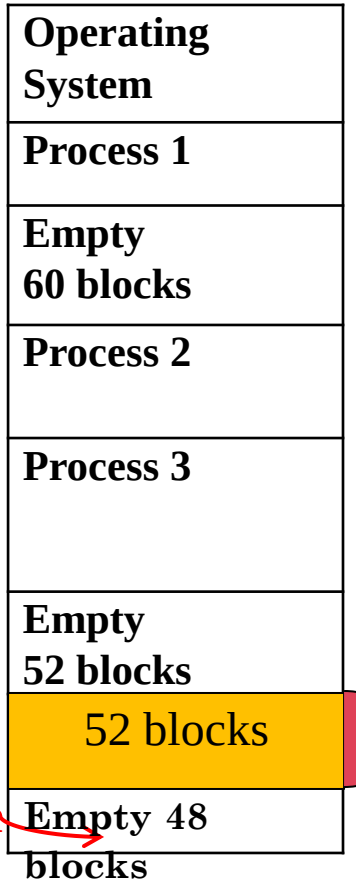


External fragmentation

- Best fit



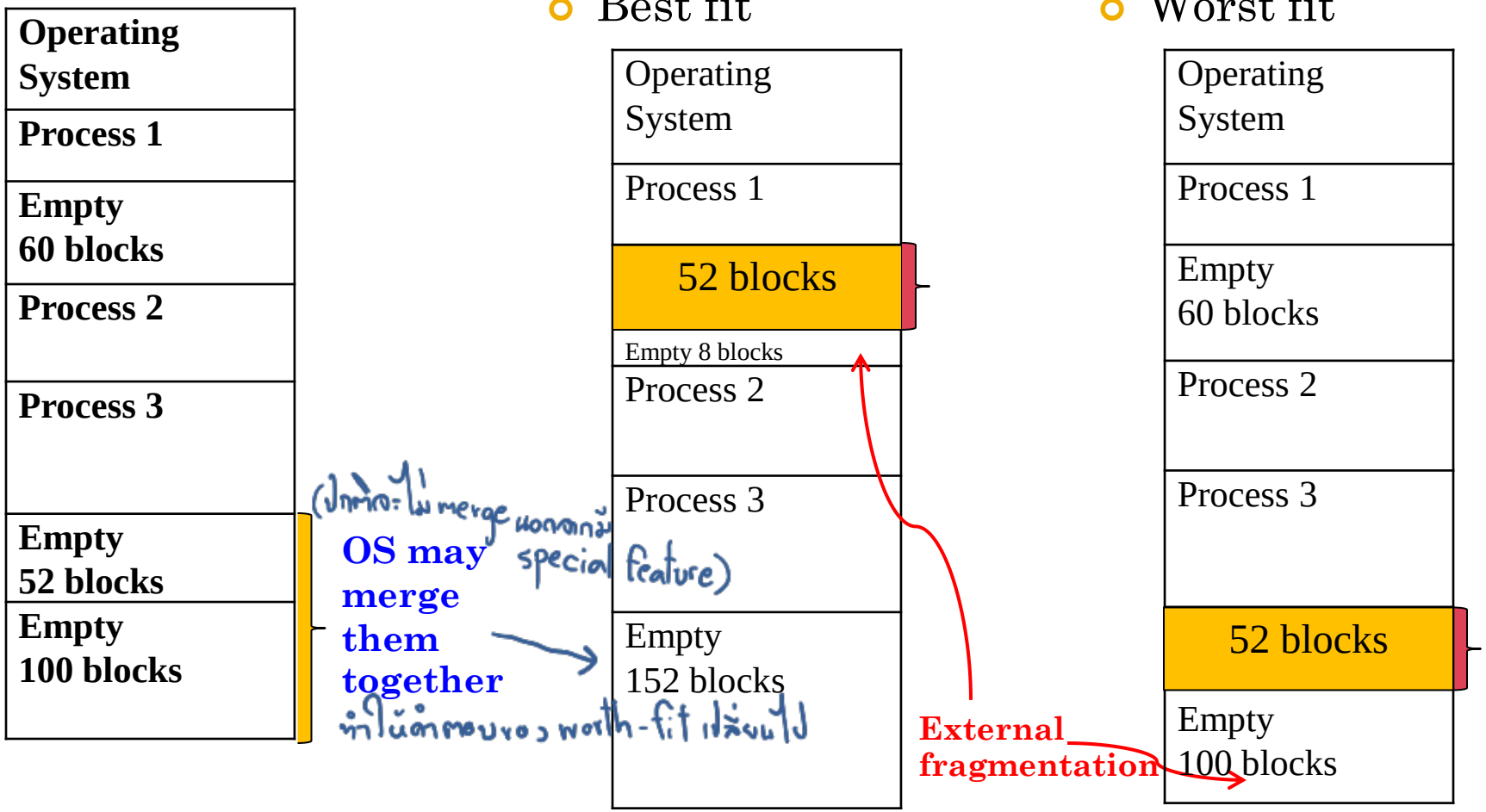
- Worst fit



External fragmentation

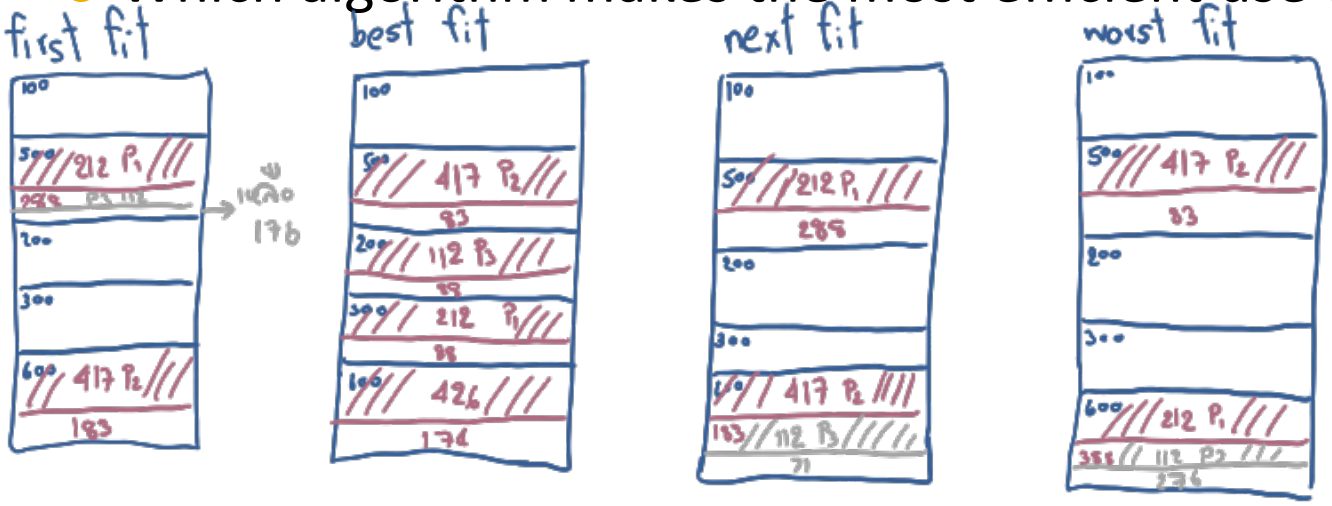
Interesting Issue of Exercise #2

OS จะไม่ merge อันในมีที่เยอะมากทำให้ต้องเสียเวลามากขึ้น



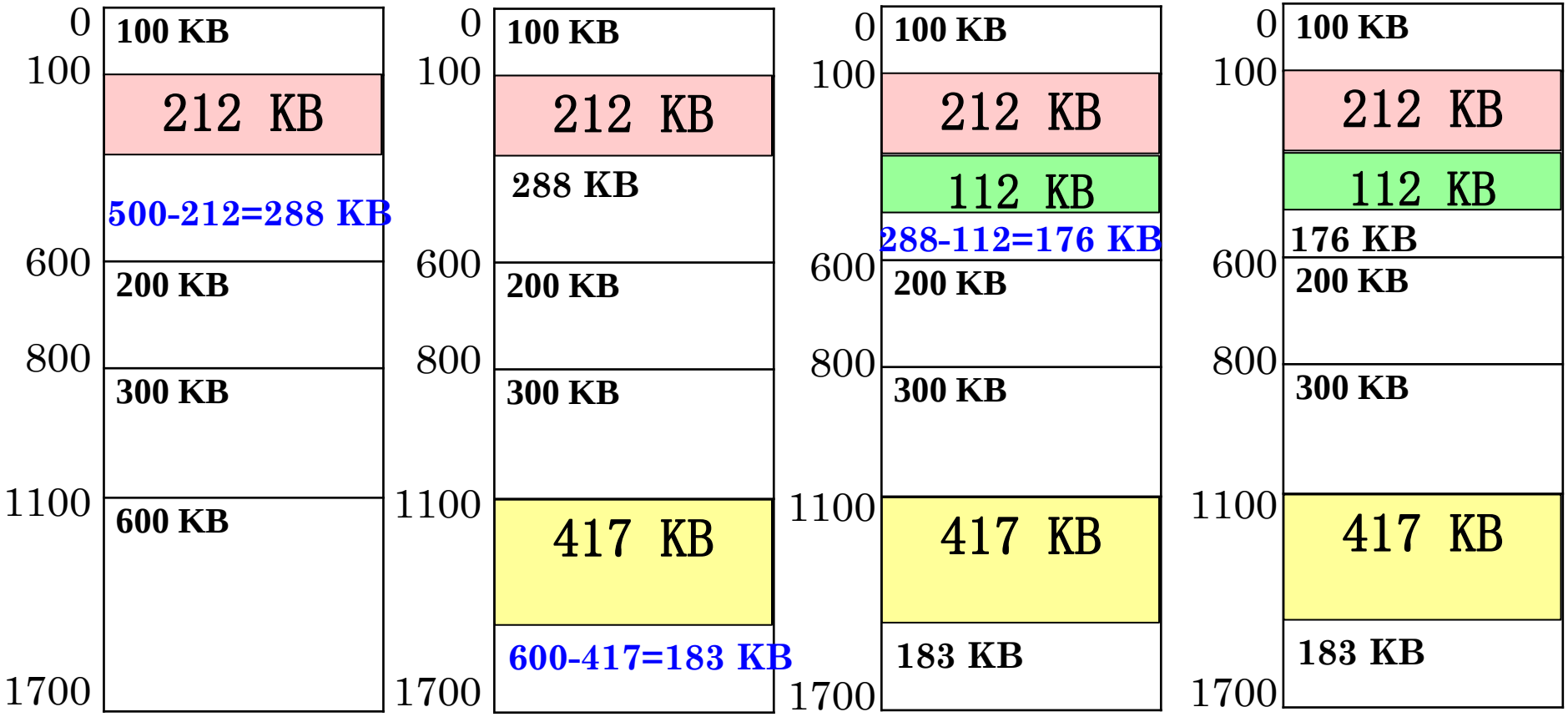
EXERCISE #3

- Assume that there are 5 memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order),
- How would each of the first-fit, best-fit, next-fit and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)?
- Which algorithm makes the most efficient use of memory?



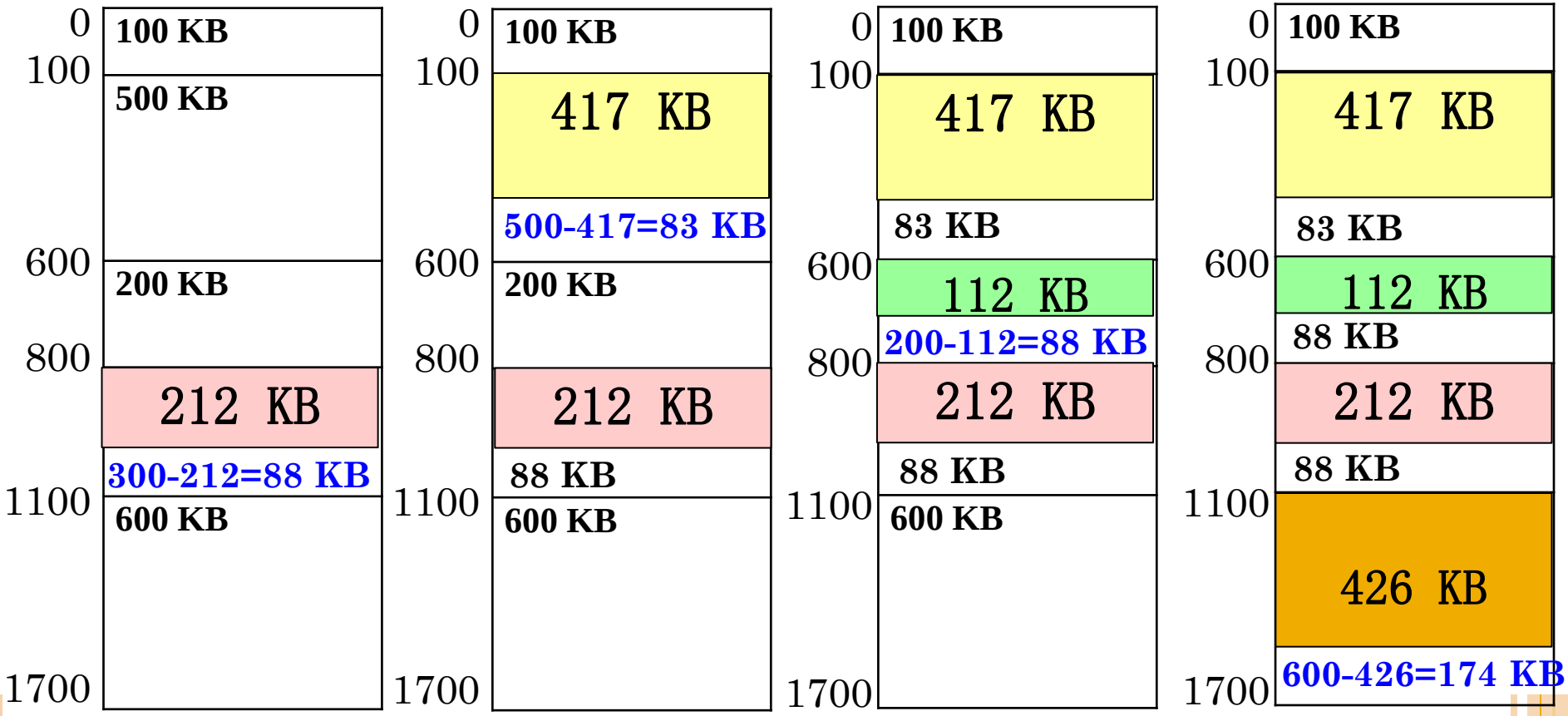
Answers #3: First Fit

- Requests for 212 KB, 417 KB, 112 KB and 426 KB



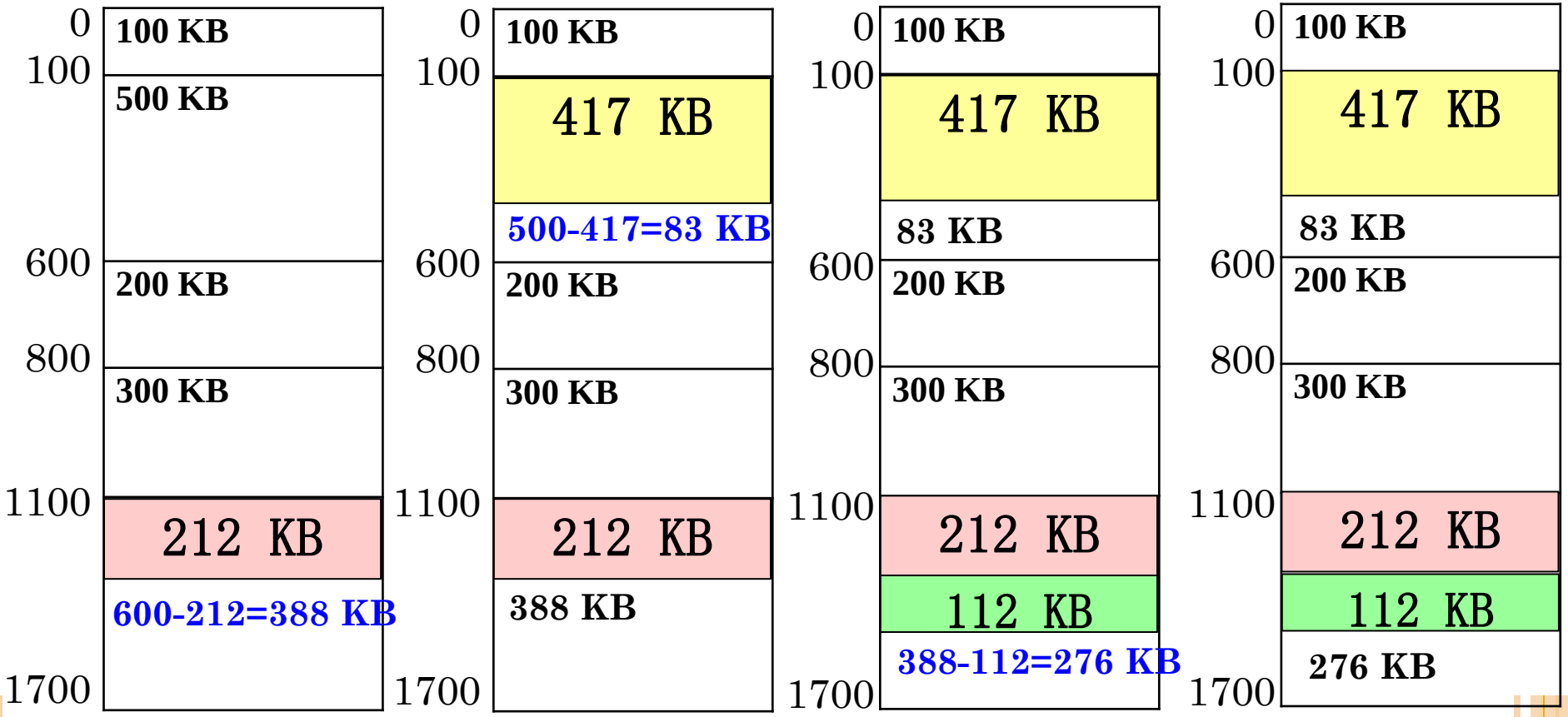
Answers #3: Best Fit

- Requests for 212 KB, 417 KB, 112 KB and 426 KB



Answers #3: Worst Fit

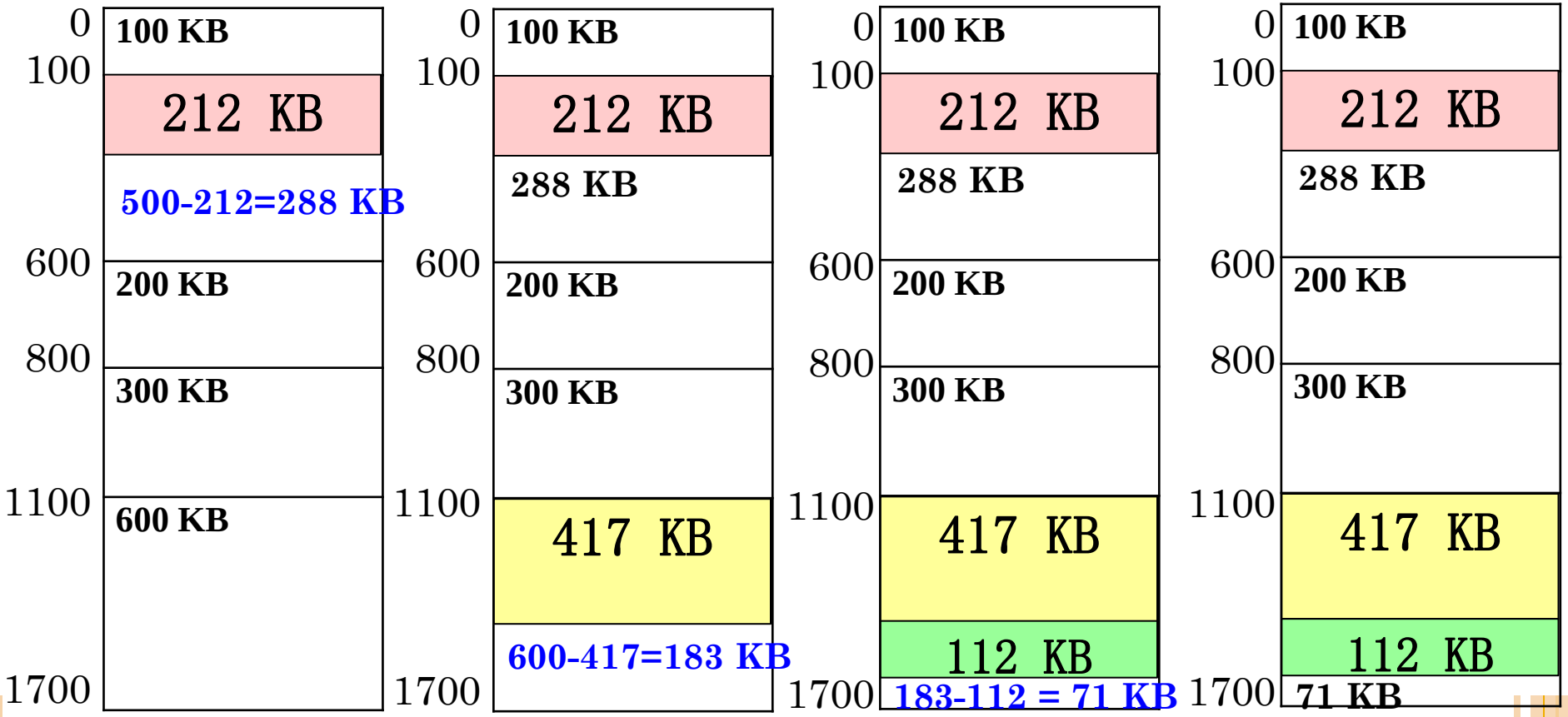
- Requests for 212 KB, 417 KB, 112 KB and 426 KB



Answers Exercise #3

Answers #3: Next Fit

- Requests for 212 KB, 417 KB, 112 KB and 426 KB



Comparison of Placement Algo.

- First-fit is not only the simplest, but usually the best and fastest as well เลือกอันที่แต่เร็ว
- The next-fit tends to produce slightly worse results than the first-fit
  - Frequently lead to an allocation from a free block at the end of memory
  - Compaction may be required more frequently
- The best-fit is usually the worst performer ใช้เวลามาก
  - May result in too small blocks to satisfy any memory allocation requests
  - Compaction must be done more frequently than other algorithms ต้องทำ compaction บ่อย

Process Relocation

↳ 9ish visual address

- Issue:
  - When a process image is first created, it is loaded into some partition in main memory
  - Later, the process may be swapped out
  - When it is subsequently swapped back in, it may be assigned to a different partition than the last time
  - Thus, the locations (of instructions and data) referenced by a process are not fixed, but change each time a process is swapped in or shifted

Process Relocation

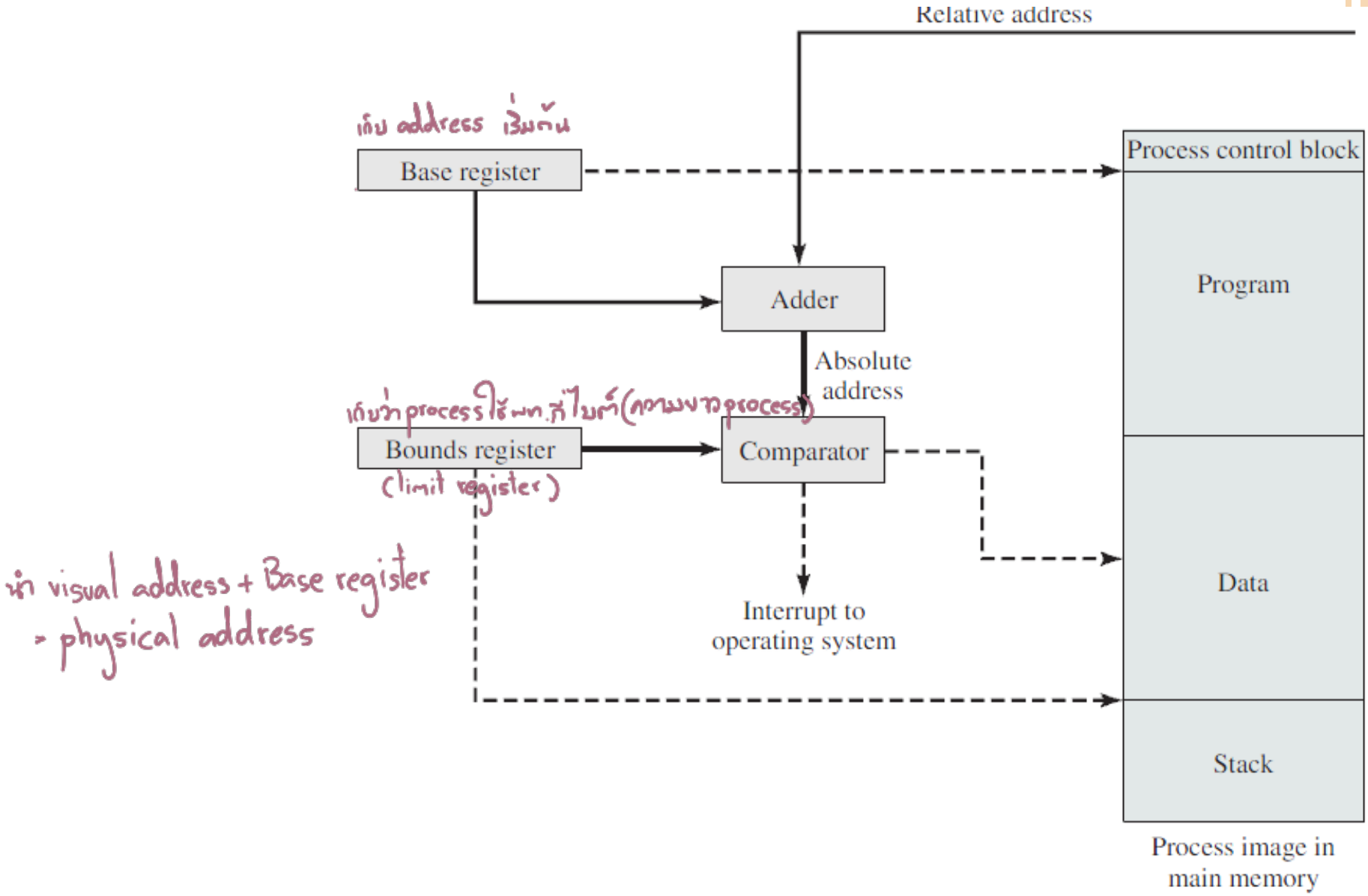
- To solve this relocation problem, we need to make a distinction between several types of addresses
  - Logical address

    A reference to a memory location independent of the current assignment of data to memory
  - Relative address

    A particular example of logical address, in which the address is expressed as a location relative to some known point (usually a value in a register)
  - Physical address or absolute address

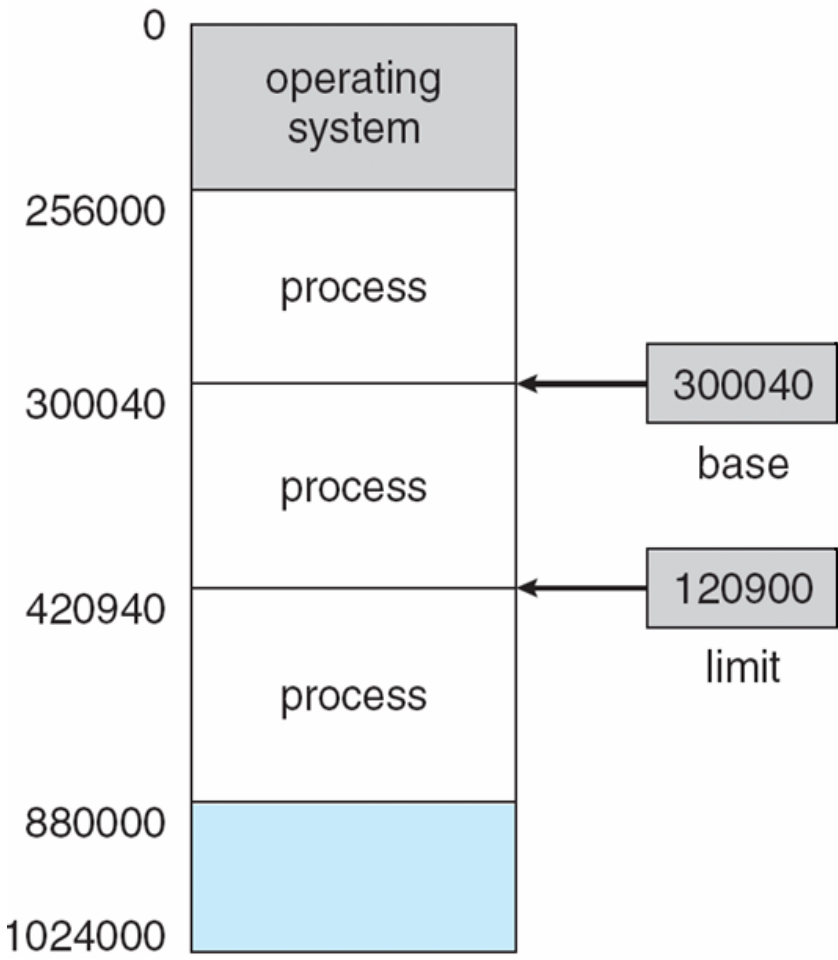
    An actual location in main memory

HARDWARE SUPPORT (1)



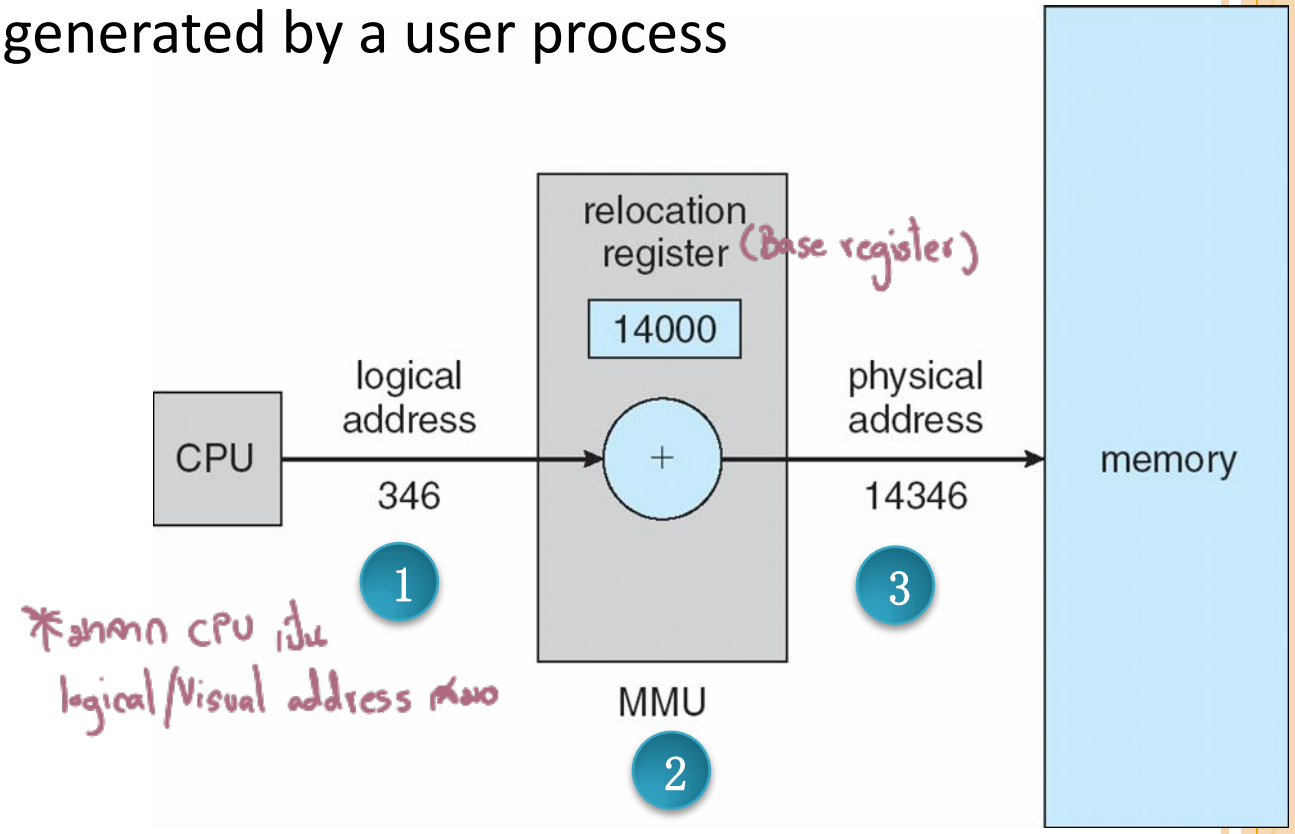
HARDWARE SUPPORT (2)

- A pair of base and limit registers define the logical address space
- The **base or relocation register** holds the smallest legal physical memory address
- The **bound or limit register** specifies the size of the range



How MMU performs process relocation?

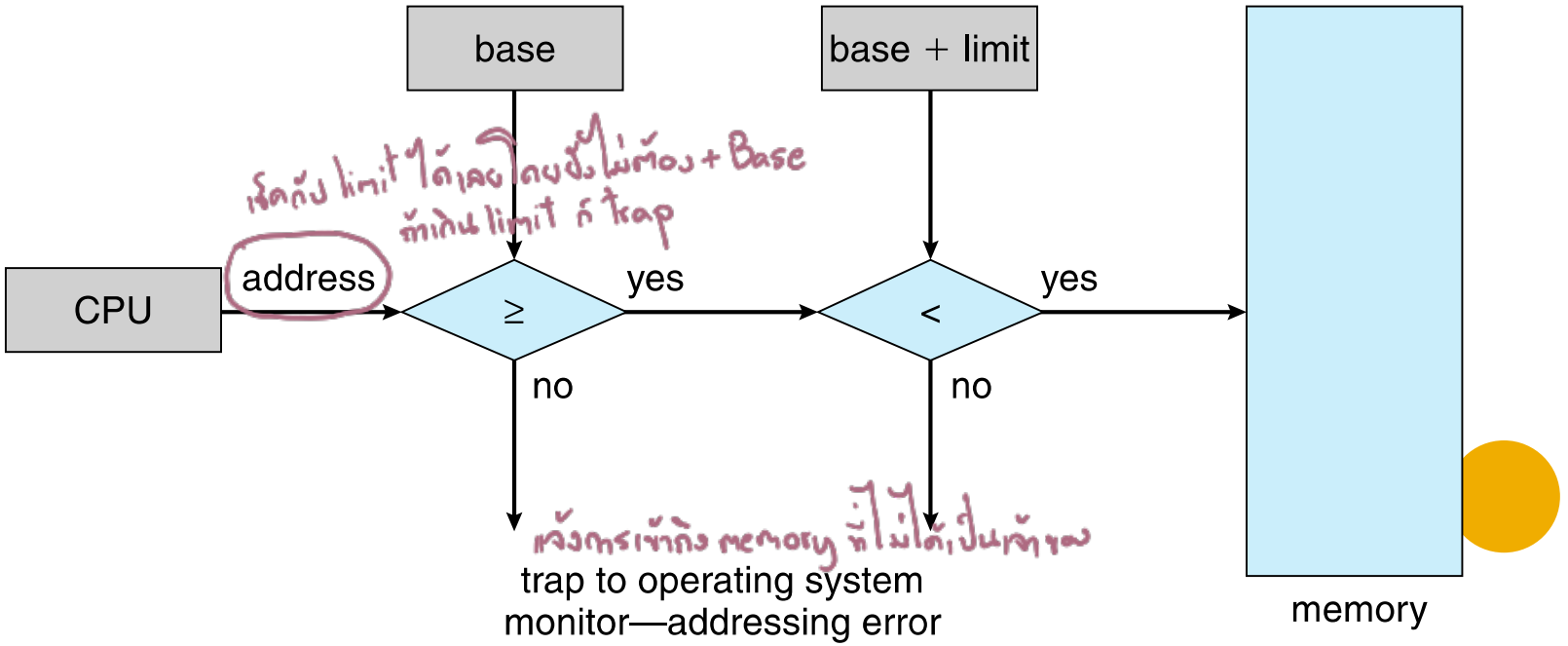
- When a process is sent to memory
  - The value of the relocation register is added to every address generated by a user process



Memory Address Protection

ป้องกันการที่ process จะไป access พื้นที่ที่ไม่ใช่ของตัวเอง

- The CPU hardware compare every memory address generated in user mode with the base and limit registers
- Any attempt by a user-mode program to access OS or other users' memory results in a trap to OS (a fatal error)



EXERCISE #4

If, in a single contiguous memory management system, the program is loaded at address 30215 (base), compute the physical addresses (in decimal) that correspond to the following logical addresses:

1) 9223
2) 2302
3) 7044

(1), (2) } logical

un physical

1.) 9223 + 30215 = 39438

2.) 2302 + 30215 = 32517

3.) 7044 + 30215 = 37259

ANSWERS #5

The base register is 42993 and the current value of the bounds register is 2031, compute the physical addresses that correspond to the following logical addresses:

1) 104 physical : 104 + 42993 = 43097
2) 1755 physical : 1755 + 42993 = 44748
3) 3041 × เกิน limit register

Buddy System

Review: Binary Number

RAM 32 MB ต้องการแบ่ง partition ละ 64 B จะได้กี่ partition

$32 = 2^5 \cdot 2^{20}$ $64 = 2^6$

$\therefore \frac{2^5 \cdot 2^{20}}{2^6} = 2^{19}$ = 512 K partition

Convert From Binary to Decimal

128	64	32	16	8	4	2	1	weight
0	0	1	0	0	1	1	0	

32 + 4 + 2 = 38 decimal

128	64	32	16	8	4	2	1	weight
1	1	0	0	1	1	0	1	

128 + 64 + 8 + 4 + 1 = 205 decimal

Powers of 2

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$

RAM 512 K แบ่งเป็น 32 partition จะได้ partition ขนาดเท่าไหร่

$512\,K = 2^9 \cdot 2^{10}$ $32 = 2^5$

$\therefore \frac{2^9 \cdot 2^{10}}{2^5} = 2^{14} = 16\ KB$

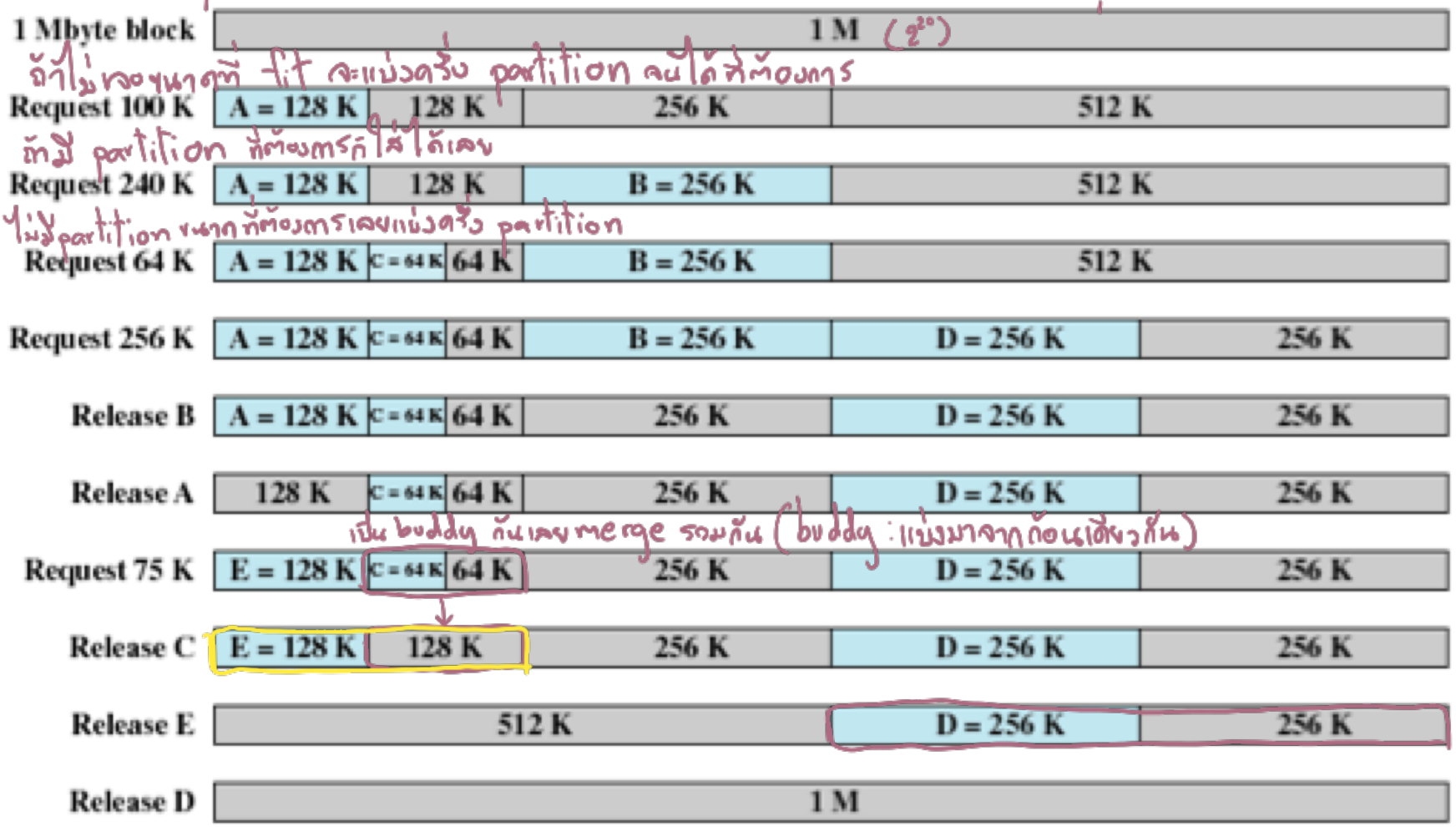
- $1K=2^{10}$, $1M=2^{20}$, $1G=2^{30}$
- Examples: $4K = 2^2 * 2^{10} = 2^{12}$, $1M/32K = 2^{20}/2^{15} = 2^5$

Buddy System

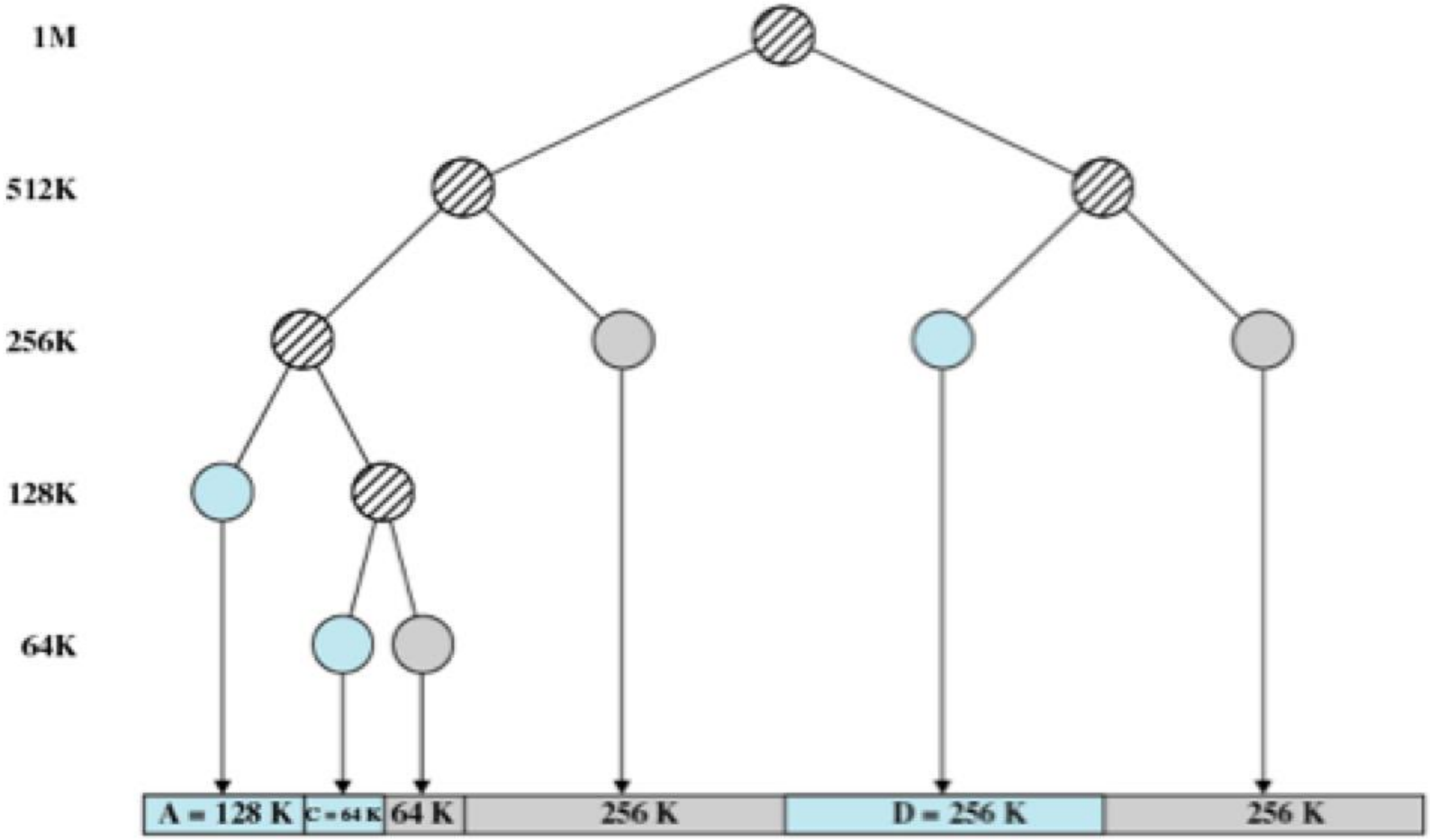
- Entire space available is treated as a single block of 2^U
- If a request of size s such that $2^{U-1} < s <= 2^U$, entire block is allocated
  - Otherwise block is split into two equal buddies
  - Process continues until smallest block greater than or equal to s is generated

EXAMPLE *ให้ดู partition แบบ fix

มองว่าเริ่มต้นมี partition เดียว ถ้าขนาดที่ขอเข้ามาต้องเป็น 2^n เท่าไหร่ก็จะ fit ที่สุด



EXAMPLE



Buddy System

- Advantages
  - Less external fragmentation
  - Cheaper search than a linked list
  - Merging adjacent blocks is easy (the buddy for any block of size 2^U is another block of the same size with the same address except the U^{th} bit is reversed)
- Disadvantage
  - Allows internal fragmentation

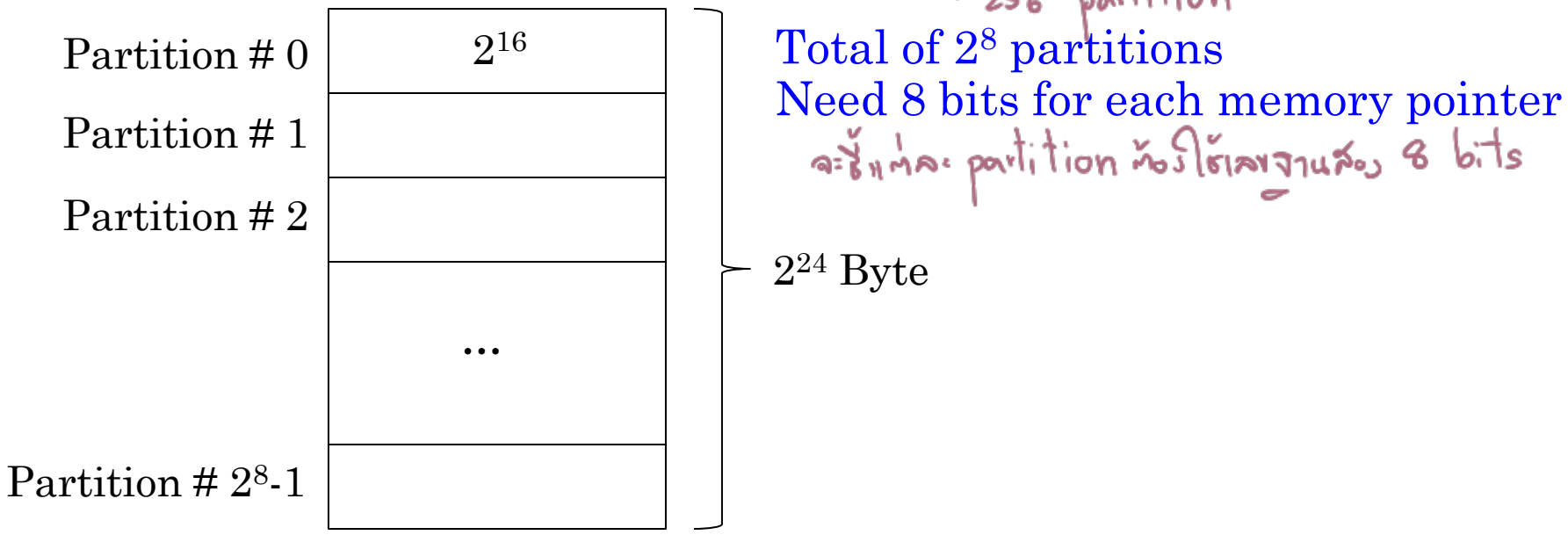
EXERCISE #6

16 MB

- สมมติให้หน่วยความจำมีขนาด 2^{24} ไบต์ และใช้วิธี fixed-partitioning แบบ equal-size partitions แต่ละ partition มีขนาด 2^{16} ไบต์ จงหาว่า Memory pointer ใน Process table จะต้องมีขนาดกี่บิต เพื่อสามารถชี้ไปยังแต่ละ partitionได้

$\frac{2^{24}}{2^{16}} = 2^8$ partition

$= 256$ partition



Total of 2^8 partitions

Need 8 bits for each memory pointer

จะชี้แต่ละ partition ต้องใช้เลขฐานสอง 8 bits

Summary

- Functions of memory management

  fix, dynamic, buddy

  Physical organization, relocation, logical organization, protection, sharing

- Single contiguous memory model
  - Memory partitioning
  - Process Relocation
  - Memory protection

  Approaches: Fixed partitioning, dynamic partitioning, buddy system