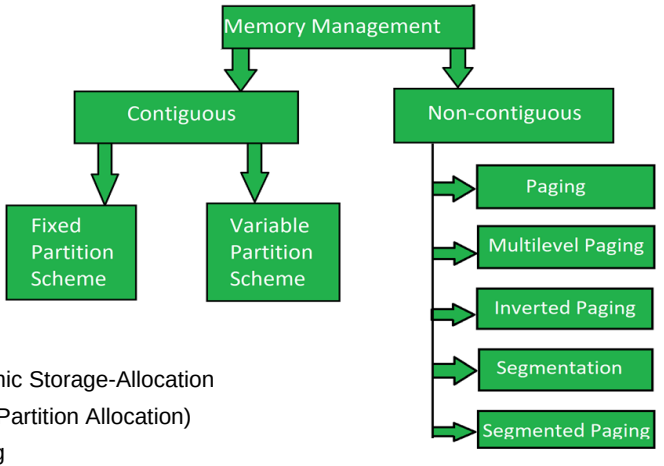
Chapter 3: Memory Management

3.1. Main Memory - Excercise

GV: Nguyễn Thị Thanh Vân

Memory management



- Dynamic Storage-Allocation

(Variable Partition Allocation)

- Paging
- Segmentation
- Segmented Paging

Logical Address & Physical Address

- Consider a logical address space of 64 pages of 1,024 words each, mapped onto a physical memory of 32 frames.
  - a. How many bits are there in the logical address?
  - b. How many bits are there in the physical address?
  - c. How many entries are there in the page table?
  - d. what is size of the page table, given each entry need 4B ?

- **Expl**: Page: LoAddr. Frame: PhyAddr. *(size của page= size của frame)*
- a) logical address
  - 64pg => 2^6 => p=6bit
  - 1024B = 2^{10}B = > d=10bit
  - => 16b
- b) physical address
  - 32frames= 2^5 = > f=5bit
  - 1024B = 2^{10}B = > d=10bit
  - => 15bit

c) 64pgs => 64entries
d) Size of the table: 64*4B

Logical Address & Physical Address

- Given segment table:
- Give the physical address corresponding to the following logical addresses:
  - a. 0,430
  - b. 1,10
  - c. 2,500
  - d. 3,400
  - e. 4,112

seg	Limit	Base
0	600	219
1	14	2300
2	100	90
3	580	1327
4	96	1952

- Lưu ý cách viết trên dùng dấu "," để tách phần chỉ số đoạn và vị trí của dữ liệu cần tìm trong đoạn.
  - Ex, 0,430 tức dữ liệu cần tìm trong bộ nhớ ảo đang ở đoạn (segment) thứ 0 và ở vị trí thứ 430 trong đoạn 0.
- **Expl**: (s,d). Check: d>base, d<limit

  430>219, 430<600 => dchi hợp lý => PhAdrr=219+430=649

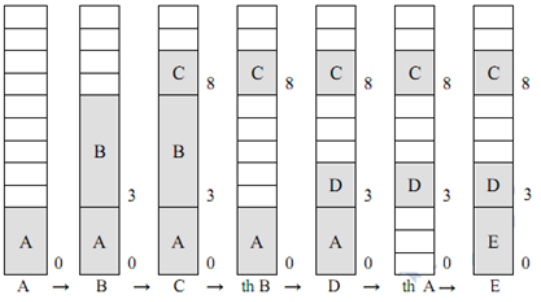
Variable Partition Allocation

- Trong cấp phát bộ nhớ với kỹ thuật **phân vùng động**. Cho các tiến trình

Tiến trình	Số đơn vị bộ nhớ yêu cầu
A	3
B	5
C	2
D	2
E	3

- Vẽ hình minh họa chuỗi cấp phát sau :

A→B→C→thu hồi B→D→thu hồi A→E



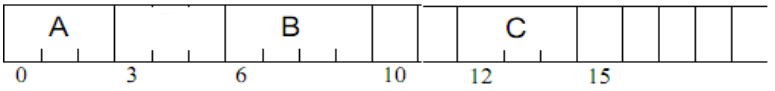
Variable Partition Allocation

- Given six memory partitions of 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB,and 375 KB (in order)?

	first-fit	best-fit	worst-fit
300	P1: 115. Free: 185		P5?
600	P2: 500. Free: 100	P2: 500. Free: 100	P3:358 .Free: 242
350	P4: 200. Free: 150		P4: 200. Free: 150
200		P4: 200. Free: 0	
750	P3:358 .Free: 392 P5: 375. Free: 17	P3:358 .Free: 392 P5: 375. Free: 17	P1: 115. Free: 635 P2: 500. Free: 135
125		P1: 115. Free: 10	

Variable Partition Allocation

- Cho hiện trạng của bộ nhớ như sau (mỗi ô là 1 đơn vị cấp phát) : A (3) (3 đơn vị cấp phát), B(4),C (3)



a. Xây dựng danh sách (liên kết) quản lý bộ nhớ.

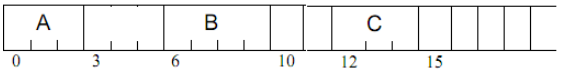
b. Sử dụng giải thuật Best-Fit. Cho biết danh sách quản lý bộ nhớ sau khi cấp phát cho E(2).

c. Sử dụng giải thuật First-Fit / Worst-Fit. Cho biết danh sách quản lý bộ nhớ sau khi cấp phát cho D(3).

d. Sử dụng giải thuật Next-Fit. Cho biết danh sách quản lý bộ nhớ sau khi cấp phát cho D(3).

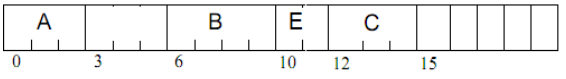
Variable Partition Allocation

- A.



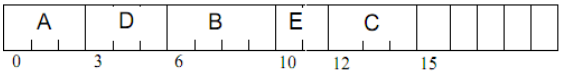
A	0	3
H	3	3
B	6	4
H	10	2
C	12	3
H	15	5

- B. Using Best-Fit, E(2)



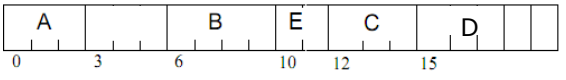
A	0	3
H	3	3
B	6	4
E	10	2
C	12	3
H	15	5

- C. Using First-Fit, D(3)

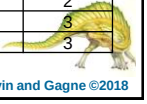


A	0	3
D	3	3
B	6	4
E	10	2
C	12	3
H	15	5

- Using Worst-Fit, D(3)



A	0	3
H	3	3
B	6	4
E	10	2
C	12	3
D	15	3



Segmentation

- Cho các tiến trình :
  - P1 có các phân đoạn S0 (200K), S1(300K), S2 (400K).
  - P2 có các phân đoạn S0 (100K), S1(400K), S2 (200K), S3 (300K).
- Xây dựng các bảng quản lý cấp phát khi hệ thống cấp phát bộ nhớ đủ theo yêu cầu cho P1 và P2 theo kỹ thuật **phân đoạn**. Biết rằng hệ thống bắt đầu cấp phát tại địa chỉ 0K.

P1		
	s0	200K
	s1	300K
	s2	400K

P2		
	s0	100K
	s1	400K
	s2	200K
	s3	300K

Segmentation

P1	900K	37000K (đchỉ của SM table P1)
P2	1000K	42000K (đchỉ của SM table P2)

P1		
	s0	200K
	s1	300K
	s2	400K

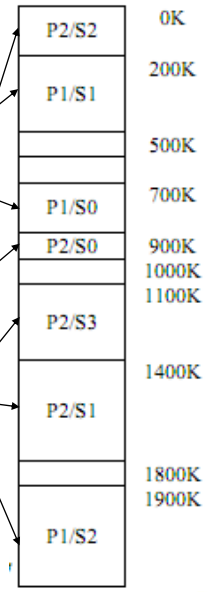
P2		
	s0	100K
	s1	400K
	s2	200K
	s3	300K

SM Table P1

0	200K	700K
1	300K	200K
2	400K	1900K

SM Table P2

0	100K	900K
1	400K	1400K
2	200K	0K
3	300K	1100K



Segmentation

- Vẽ sơ đồ biến đổi địa chỉ logic thành địa chỉ vật lý trong kỹ thuật **phân đoạn**. Cho địa chỉ bắt đầu cấp phát trong bộ nhớ là 200K. Cho bảng phân đoạn (SMT) của P như sau

S	Kích thước	Địa chỉ
0	300K	200K
1	200K	1300K
2	500K	700K
3	400K	1500K

- Tính địa chỉ vật lý tương ứng với các địa chỉ logic sau

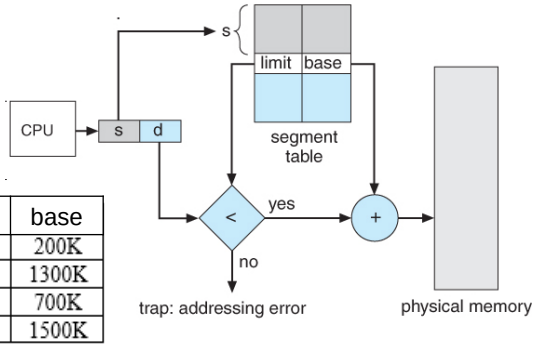
s0	130K

s2	270K

s3	125K

Segmentation

- s in table, d< limit
- đ/c vật lý: base+d



S	limit	base
0	300K	200K
1	200K	1300K
2	500K	700K
3	400K	1500K

S=0	130K

S=0: (d=130) < (limit=300) → đ/c vật lý = 200K + 130K = 330K

S=2	270K

S=2: (d=270) < (limit=500) → đ/c vật lý = 700K + 270K = 970K

S=3	425K

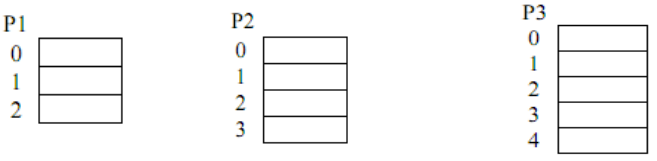
S=3: (d=425) > (limit=400) → addressing error

S=4	25K

S=4 not in table → addressing error

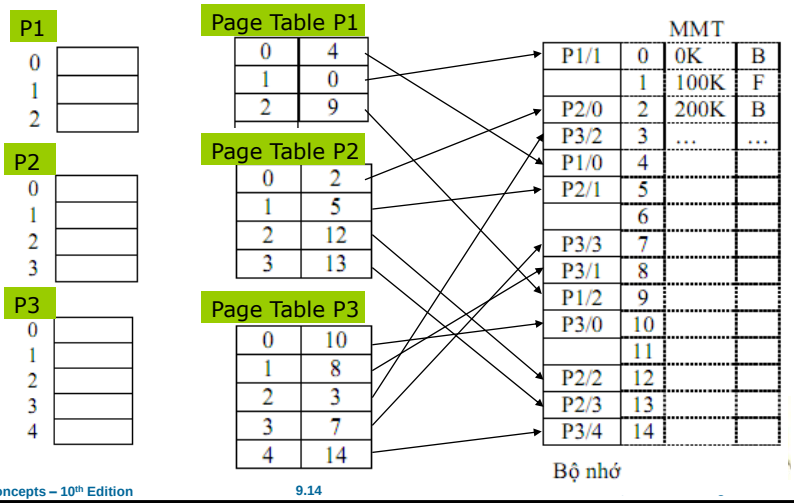
Paging

- Cho kích thước trang và kích thước khung trang là 100K và địa chỉ bắt đầu cấp phát là 0K. Tiến trình P1 có 3 trang, P2 có 4 trang, P3 có 5 trang. Xây dựng các bảng quản lý cấp phát. Biết rằng hệ thống cấp đủ theo yêu cầu của tiến trình



Paging

P1	300K	37000 (đchỉ của Page Table P1)
P2	400K	42000 (đchỉ của Page Table P2)
P3	500K	45000 (đchỉ của Page Table P2)



Paging

- Vẽ sơ đồ biến đổi địa chỉ logic thành địa chỉ vật lý trong kỹ thuật **phân trang**. Cho kích thước trang và kích thước khung trang là 100K, địa chỉ bắt đầu cấp phát trong bộ nhớ là 0K. Cho bảng trang (PMT) của tiến trình P như sau :

p	f
0	7
1	2
2	5
3	4

- Tính địa chỉ vật lý tương ứng với các địa chỉ logic sau :

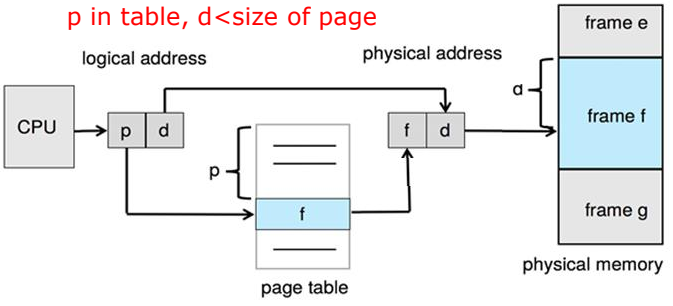
0	30K

2	70K

3	25K

Paging

p in table, d<size of page



p	f
0	7
1	2
2	5
3	4

0	30K

p = 0 → f = 7 , d = 30K → đ/c vật lý = 7*100K + 30K = 730K

2	70K

p = 2 → f = 5 , d = 70K → đ/c vật lý = 5*100K + 70K = 570K

3	25K

p = 3 → f = 4 , d = 25K → đ/c vật lý = 4*100K + 25K = 425K

4	25K

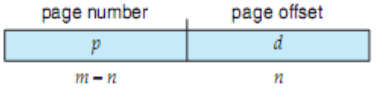
p = 4 → not in table → address error

1	125K

p = 1 → f = 2 , d = 125K>page size → address error

Structure of the Page Table

- Modern computer systems support a large 32-bit logical address space
  - 2^{32} physical page frames (m=32)
  - Page size of 4 KB (2^{12}Byte), d=12



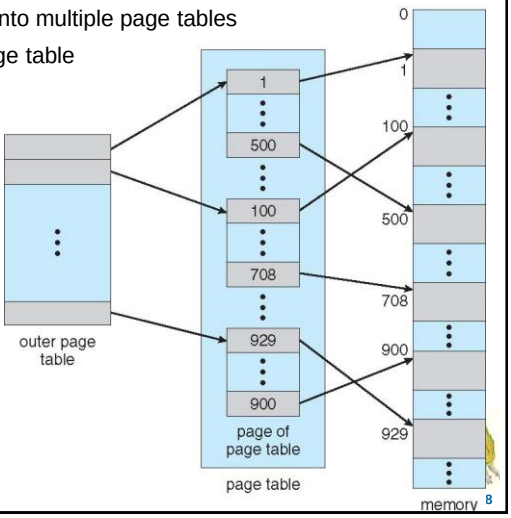
=> Page table would have 1 million entries: $2^{20} = 2^{32} / 2^{12}$ (p=32-12=20)

- If each entry is 4 bytes ➔ each process requires: 2^{20} * 4bytes= 4MB of physical address space for the page table alone – **high cost**
  - Ex, If a process size is 1GB (2^{30}B). %: 4MB/2^{10}MB = 1/2^{8}= 1/256
  - Do not want to allocate that contiguously in main memory
- One simple solution is to divide the page table into smaller units.
  - We can accomplish this division in several ways

- The most common techniques for structuring the page table
  - Hierarchical Paging
  - Hashed Page Tables
  - Inverted Page Tables

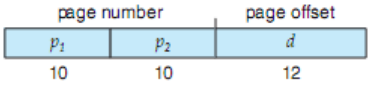
Hierarchical Page Tables

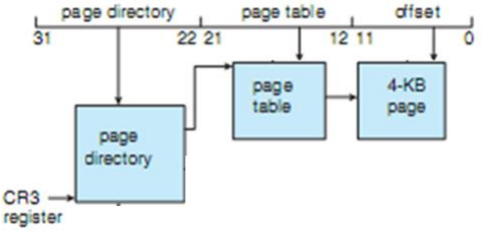
- Most modern computer systems support a large logical address space (2^{32} -> 2^{64}).
  - =>the page table itself becomes excessively large
- Noncontiguous => divide the page table into **smaller pieces**
- Break up the logical address space into multiple page tables
- A simple technique is a two-level page table
- We then page the page table



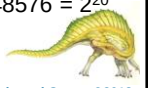
Hierarchical Page Tables

- Shows how to organize a 2-level page table in a 32-bit Windows operating system. What is the meaning of this work?
- Expl: Trong HĐH windows 32 bit. Địa chỉ logic 32 bit được tổ chức như sau
  - p1: Thư mục trang (=1024 mục)
  - p2: Bảng trang (=1024 bảng)
  - d: Kích thước trang (=4096 Byte=4 KB)





- Cách tổ chức này giúp tiết kiệm thời gian tìm kiếm 1 trang trong 1048576 = 2^{20} chỉ mất 1024 + 1024 lần tìm kiếm.



Hierarchical Page Tables

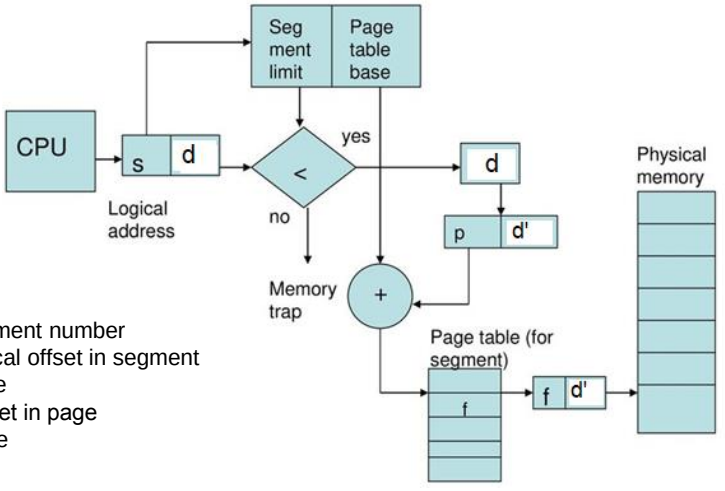
- A computer system has a 36-bit virtual address space with a page size of 8K, and 4 bytes per page table entry.
  - How many pages are in the virtual address space?
  - What is the maximum size of addressable physical memory in this system?
  - If the average process size is 8GB, would you use a one-level, two-level, or three-level page table? Why?
  - Compute the average size of a page table in question 3 above

Inverted Page Tables

- The BTV operating system has a 21-bit virtual address, yet on certain embedded devices, it has only a 16-bit physical address. It also has a 2-KB page size. How many entries are there in each of the following?
- a. A conventional, single-level page table
- b. An inverted page table
- What is the maximum amount of physical memory in the BTV operating system?

- Expl:
- a. single-level page table: entry tính theo số trang
  - $2^{21}/ 2^{11} = 2^{10}$
- b. An inverted page table: entry tính theo bộ nhớ vật lý:
  - 16bit physical / frame size: $2^{16}/ 2^{11} = 2^{5}$
- c. the maximum amount of physical memory:
  - $2^{5} * 2KB = 64KB$

Segmented Paging



s: segment number
d: logical offset in segment
p: page
d': offset in page
f: frame

Segmented Paging

- Cho các tiến trình :
  - P1 có các phân đoạn S0 (250K), S1(370K), S2 (420K).
  - P2 có các phân đoạn S0 (180K), S1(470K).
- Xây dựng các bảng quản lý cấp phát khi hệ thống cấp phát bộ nhớ đủ theo yêu cầu cho P1 và P2 với kỹ thuật **phân đoạn kết hợp**. Biết rằng kích thước trang, khung trang là 100K và địa chỉ bắt đầu cấp phát là 0K

P1

S0	0
	1
	2
S1	0
	1
	2
	3
S2	0
	1
	2
	3
	4

P2

S0	0
	1
S1	0
	1
	2
	3
	4

Segmented Paging

P1

S0	0
	1
	2
S1	0
	1
	2
	3
S2	0
	1
	2
	3
	4

SMT P1

0	300K	(0)
1	400K	(1)
2	500K	(2)

(0) PT S0

0	0
1	3
2	8

(1) PT S1

0	7
1	6
2	13
3	17

(2) PT S2

0	10
1	2
2	20
3	15
4	22

P2

S0	0
	1
S1	0
	1
	2
	3
	4

SMT P2

0	200K	(0)
1	500K	(1)

(0) PMT S0

0	4
1	12

(1) PMT S1

0	11
1	16
2	18
3	19
4	9

P1/s0/0	0	0K	B
	1	100K	F
P1/s2/1	2	200K	B
P1/s0/1	3	...	...
P2/s0/0	4		
	5		
P1/s1/1	6		
P1/s1/0	7		
P1/s0/2	8		
P2/s1/4	9		
P1/s2/0	10		
P2/s1/0	11		
P2/s0/1	12		
P1/s1/2	13		
	14		
P1/s2/3	15		
P2/s1/1	16		
P1/s1/3	17		
P2/s1/2	18		
P2/s1/3	19		
P1/s2/2	20		
	21		
P1/s2/4	22		

Bộ nhớ

Segmented Paging

- Vẽ sơ đồ biến đổi địa chỉ logic thành địa chỉ vật lý trong kỹ thuật **phân đoạn kết hợp**. Cho địa chỉ bắt đầu cấp phát trong bộ nhớ là 0K, kích thước trang và khung trang là 100K. Cho bảng phân đoạn (SMT) và các bảng trang (PMT) của P như sau :

SMT

S	Kích thước	Địa chỉ
0	300K	(0)
1	400K	(1)
2	500K	(2)

(0)PMT của S0

0	5
1	4
2	1

(1)PMT của S1

0	3
1	7
2	10
3	9

(2)PMT của S2

0	8
1	2
2	6
3	15
4	12

- Tính địa chỉ vật lý tương ứng với các địa chỉ logic sau

s1	330K

s2	230K

Từ 1 seg sẽ phân bổ trong 1 trang.
Size của seg (d) /size của trang (100KB) = số trang
=> p= 330K div 100K = 3
Phần offset d' trong trang: 330 mod 100=30
Từ PMT của s1, p=3 => f=9
Địa chỉ vật lý = f * size của frame + offset d'
=> 9*100K + 30K = 930

s2 → (2) PMT s2; kt = 500K > d = 230K;
d(230K) mod 100K = 30K = d'
d(230K) div 100K = 2 = p
từ (2) PMT s2 và p = 2 → f = 6
địa chỉ vật lý = 6*100K + d'(30K) = 630K

Effective Access Time

- Xét một hệ thống sử dụng kỹ thuật phân trang, với bảng trang được lưu trữ trong bộ nhớ chính. Thời gian cho một lần truy xuất bộ nhớ bình thường là 200 nanoseconds
- a) Nếu không dùng TLB, tổng cộng mất bao nhiêu thời gian cho một thao tác tìm dữ liệu trong hệ thống này?
- b) Nếu sử dụng TLB với hit-ratio (tỉ lệ tìm thấy) là 75%, thời gian để tìm trong TLB xem như bằng 0, tính thời gian truy xuất bộ nhớ trong hệ thống (effective memory reference time)

Effective Access Time

- a) Không dùng TLB: Để tìm một dữ liệu khi biết địa chỉ luận lý của nó, cần 2 thao tác truy xuất bộ nhớ:
  - Thao tác 1: Truy xuất vào bảng phân trang để tìm được địa chỉ frame vật lý
  - Thao tác 2: Khi có được frame, biết địa chỉ vật lý, truy xuất vào bộ nhớ vật lý lấy dữ liệu
  - Đề bài cho một lần truy xuất bộ nhớ tốn 200 ns → 2 thao tác truy xuất bộ nhớ sẽ tốn 400 ns. Thời gian cần = 400 ns
- b) Dùng TLB: Thời gian truy xuất TLB = 0 = ε
  - Thời gian tìm thấy dữ liệu trong bộ nhớ vật lý khi TLB hit (tức tìm thấy chỉ số frame trong TLB) = ε + thời gian truy xuất vào bộ nhớ vật lý

    = 0 + 200 = 200 ns
  - Thời gian tìm thấy dữ liệu trong bộ nhớ vật lý khi TLB miss (tức không tìm thấy chỉ số frame trong TLB) = ε + thời gian truy xuất vào bảng phân trang + thời gian truy xuất bộ nhớ vật lý = 0 + 200 + 200 = 400 ns
  - Tổng thời gian truy xuất bộ nhớ trong hệ thống (effective memory reference time) = 200*0.75 + 400*0.25 = 250 ns

End of Chapter 3.1