ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LUYỆN TẬP 5

| CHỦ ĐỀ |

BỘ NHỚ VÀ BỘ NHỚ ẢO

| GIẢNG VIÊN HƯỚNG DẪN |

TS. Vũ Thị Mỹ Hằng

| SINH VIÊN |

21120302 – Huỳnh Trí Nhân

Thành phố Hồ Chí Minh – 2023

MỤC LỤC

1 NỘI DUNG BÀI LÀM

1.1 Câu 1

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

What are the physical addresses of the following logical addresses?
a) 0, 701
b) 1, 8
c) 2, 100
d) 3, 429
e) 4, 78

Bài làm :

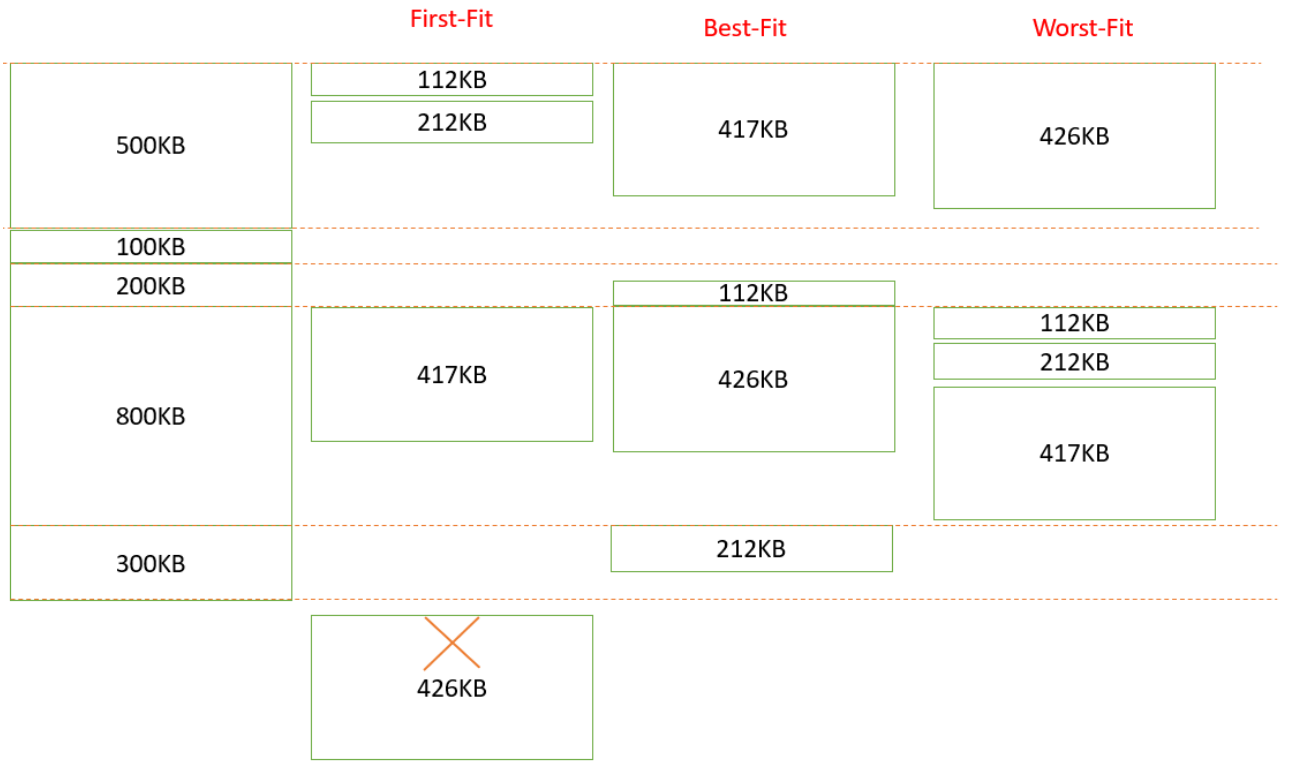
a) Không hợp lệ do cần phải tìm đến offset 700 trong khi segment 0 chỉ có limit là 600
b) Hợp lệ, địa chỉ physical là segment 1 địa chỉ 2308
c) Không hợp lệ, bởi vì offset 10 là giới hạn của segment 2, offset luôn nhỏ hơn limit
d) Hợp lệ, địa chỉ physical là segemtn 3 địa chỉ 1756
e) Họp lệ, địa chỉ physical là segemt 4, địa chỉ 2030

1.2 Câu 2

- **First-Fit:**
  - Chiến lược first-fit nhanh chóng tìm thấy không gian trống đầu tiên đủ lớn để đáp ứng yêu cầu, giảm thiểu thời gian tìm kiếm.
  - Đơn Giản, Dễ Hiểu: Thuật toán này đơn giản và dễ hiểu, làm cho việc triển khai và bảo trì trở nên dễ dàng hơn.
  - Phù Hợp cho Hệ Thống Có Thời Gian Thực: Do tính nhanh chóng của nó, first-fit có thể phù hợp với các hệ thống yêu cầu thời gian phản hồi nhanh.
  - ⇨ **Nhược điểm**: External Fragmentation: First-fit có thể dẫn đến đoạn vỡ ngoại bộ, do việc sử dụng không gian trống đầu tiên mà không cần xem xét kích thước tối ưu. Với

thời gian, các khe trống nhỏ có thể phân tán khắp bộ nhớ, làm cho việc tìm kiếm không gian trống phù hợp trở nên khó khăn và chậm chạp hơn.

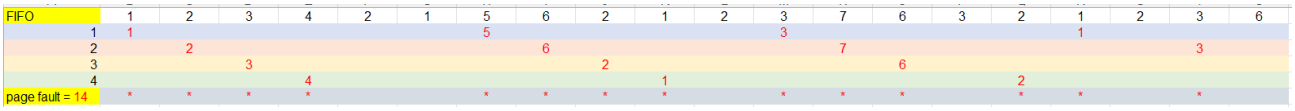
- **Best-Fit**
    - Giảm Đoạn Vỡ Nội Bộ: Best-fit cố gắng tối ưu hóa việc sử dụng bộ nhớ bằng cách tìm không gian trống phù hợp nhất với yêu cầu, giảm thiểu không gian lãng phí.
    - Hiệu Quả Bộ Nhớ Cao: Phù hợp với các hệ thống có nguồn bộ nhớ hạn chế, nơi mà việc tối ưu hóa không gian bộ nhớ là quan trọng.
    - Phù Hợp cho Ứng Dụng Cần Chính Xác: Trong các ứng dụng cần độ chính xác cao trong việc sử dụng bộ nhớ, best-fit là lựa chọn tốt.
    - ⇨ **Nhược điểm** : Internal Fragmentation: Best-fit có thể tạo ra nhiều đoạn vỡ nội bộ nhỏ, vì nó thường chọn phân đoạn chính xác với kích thước yêu cầu. Việc tìm kiếm không gian trống tối ưu nhất có thể mất nhiều thời gian, đặc biệt trong hệ thống có nhiều yêu cầu bộ nhớ và phân đoạn.
- **Worst-Fit:**
    - Tối Ưu Hóa Không Gian Lớn: Worst-fit giữ lại các không gian lớn trống, có thể hữu ích cho các yêu cầu bộ nhớ lớn sau này.
    - Phân Tán Đoạn Vỡ: Giúp phân tán đoạn vỡ nội bộ qua nhiều phần của bộ nhớ, có thể giảm thiểu tác động của đoạn vỡ.
    - ⇨ **Nhược điểm** :
    - Internal Fragmentation: Worst-fit thường để lại các đoạn vỡ nội bộ lớn, vì nó chọn không gian trống lớn nhất mỗi lần phân bổ.
    - Không Hiệu Quả cho Yêu Cầu Bộ Nhớ Nhỏ: Khi phân bổ không gian cho các yêu cầu bộ nhớ nhỏ, worst-fit có thể lãng phí một lượng lớn không gian bởi vì nó luôn chọn phân đoạn lớn nhất.
    - Hiệu Suất Không Ổn Định: Worst-fit có thể dẫn đến hiệu suất không ổn định, vì các kích thước của phân đoạn trống thay đổi đáng kể sau mỗi lần phân bổ.



⇨ Đối với trường hợp này theo em best-fit là lựa chọn tốt nhất vì số lượng partition nhỏ và số lượng process cũng nhỏ.

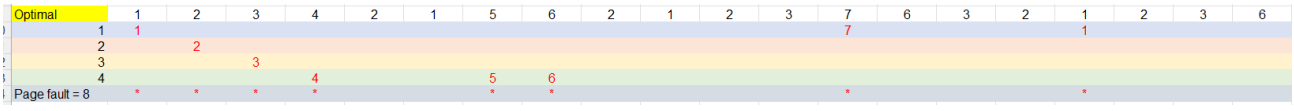
1.3 Câu 3

Ta có t_m =102ns , t_p = 192ns



Tỉ lệ lỗi trang là: p = 14/20 = 0.7

EAT = (1-p)* t_m + p* t_p = 0.3* 102 + 0.7*192 = 165ns



Tỉ lệ lỗi trang là: p = 8/20 = 0.4

EAT = (1-p)* t_m + p* t_p = 0.6* 102 + 0.4*192 = 138ns

RLU	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1					1				1				6						6
2		2			2				2		2					2		2		
3			3				5					3			3				3	
4				4				6					7				1			
Page fault = 10	*	*	*	*			*	*				*	*	*			*			

Tỉ lệ lỗi trang là: p = 10/20 = 0.5

EAT = (1-p)* t_m + p* t_p = 0.5* 102 + 0.5*192 = 147ns

Second Change	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1						5					3				2				
2		2						6					7						3	
3			3						2					6						
4				4						1										
FIFO	1	12	123	1234	1234	1234	2345	3456	4562	5621	5621	6213	2137	1376	1376	7612	7612	7612	6123	6123
Reference bit	1	12	123	1234	134	34	345	3456	2	21	1	13	137	1376	176	76	761	7612	3	36
Page fault = 14	*	*	*	*			*	*	*	*		*	*	*		*			*	

Tỉ lệ lỗi trang là: p = 14/20 = 0.7

EAT = (1-p)* t_m + p* t_p = 0.3* 102 + 0.7*192 = 165ns

1.4 Câu 4

Tóm tắt : page size = 1KB = 2^{10} bytes, p = 3, f = 5

a) Page size = frame size do đó số lượng bit để lưu trữ kích thước đều là 10 bít. Số lượng bit của physical address là: f + d = 15bit
b) SỐ lượng bit của virtual memory để lưu trữ program P là : p + d = 13 bit => size of virtual memory = 2^{13}
c) Size of physical memory là : 2^{15} bit

1.5 Câu 5

Tóm tắt : page size = 8kB ~ 2^{13} bytes, 48 bit logical address, byte addressable memory: 320MB

a) Ta có page size =frame size = 8KB => number of frame = 320MB/ 8KB **= 40* 2^{10} frames**

b) Ta có 48- 13 = 25 bit để lưu trữ page => maximum number of page = 2^{25} pages

c) Ta có $\frac{logical\ address}{page\ size} = p\ \frac{d}{page\ size}$

Từ đó ta có 1892 / 2^{13} **= <0, 1892>**

15296 / 2^{13} **= <1, 7104>**

20300 / 2^{13} **= <2, 3916>**

1.6 Câu 6

Tóm tắt: two-level page, 32-bit addressable: 9-bit top-level, 11-bit second-level, word-addressable (an addressable point to a word-memory of 4 bytes) of 10GB

a) Số bit offset để lưu page size là d = 32 – 9-11 = 12 bit => page size = 2^{12} ~ 4KB
b) Page size = frame size = 2^{12} => number of frame = 10GB / 2^{12} = 10 * 2^{30} / 2^{12} = 10 * 2^{18} frames
c) Maximum size of process page = 2^{32}
d) 2.8 GB / page-size = 2.8GB / 2^{12} = 716.8 pages =>internal fragmentation
e) Có 2 loại fragmentation issue
   - Internal fragmentation: Xảy ra khi bộ nhớ được chia thành các khối cố định, và một tiến trình được cấp phát một khối lớn hơn một chút so với nhu cầu của nó. Phần không gian dư thừa trong khối được cấp phát, quá nhỏ để gán cho tiến trình khác, dẫn đến đoạn vỡ nội bộ. Điều này thường gặp trong hệ thống phân trang vì mỗi trang có kích thước cố định. Nếu trang cuối cùng của một tiến trình không cần sử dụng hết kích thước của trang, phần còn lại của trang đó sẽ bị lãng phí.
   - External fragmentation: Đoạn vỡ ngoại bộ ít gặp hơn trong hệ thống phân trang thuần túy nhưng có thể xảy ra trong các hệ thống sử dụng sự kết hợp của phân trang và phân đoạn (segmentation). Loại đoạn vỡ này xảy ra khi có đủ tổng không gian bộ nhớ để đáp ứng yêu cầu bộ nhớ, nhưng các không gian khả dụng không liên tục, do đó không thể sử dụng cho yêu cầu, dẫn đến việc sử dụng bộ nhớ không hiệu quả.

1.7 Câu 7

Ta có công thức EAT = h($t_c + t_m$) + (1-h) ($t_c + 2*t_m$) áp dụng với level of paging = 1

Tóm tắt : t_m: 132ns , t_c : 25ns, h : 75% , level of paging : 3 => EAT = ?

=> EAT = 0.75* (25 + 132) + 0.25* (25 + 4*132) = 256ns

1.8 Câu 8:

Tóm tắt : t_m : 150ns , t_c : 25ns, level of paging :1 , EAT = 225ns => h = ?

Ta có : 225 = h(25+150) + (1-h)(25 + 2*15)

⇨ **h = 2/3 ~ 66,667%**