TD N°5: Management memory (Segmentation) and Page-replacement algorithms

Exercise n°1

We consider the following table of segments for a process P_1:

Segment	Base	Limite
0	540	234
1	1254	128
2	54	328
3	2048	1024
4	976	200

1) Calculate the physical addresses corresponding to the following virtual addresses (you may indicate any addressing errors) where the addresses are given in the form (n° segment:offset):
   (0:128), (1:100), (2:465), (3:888), (4:344)

- (0:128): valid offset (128<234). Physical_addr = base + limit------------------- = 540 + 128 = 668.
- (1:100): valid offset (100<128). Physical_addr = base + limit = 1254+ 100 = 1354.
- (2:465): invalid offset (465>328).
- (3:888): valid offset (888<1024). Physical_addr = base + limit = 2048 + 888 = 2936.
- (4:344): invalid offset because (344>200).

2) Is the virtual address (4,200) valid?

   No. In a segment of length 200, the valid displacements are in the interval [0-199].

Exercise n°2

We consider a system with a paged segmented virtual memory where the size of a page is 4KB and a physical memory of 64KB. The address space of a process P is made up of three segments S0, S1 and S2 of sizes 16KB, 8KB and 4KB respectively. At a given moment, for process P, pages 1 and 2 of segment S0, page 1 of segment S1 and page 0 of segment S2 are loaded into physical memory, in slots 2, 0, 9 and 12 respectively. For data located in the address space of process P at decimal address 8212, indicate:

Solution:

Segment	Page	Frame
S0	1	2
	2	0
S1	1	9
S2	0	12

8212 = 2*4096 + 20

1) The segment number

   segment = S0

2) The page number in the segment

   Page = 2

3) The offset within the page

   Offset = 20

4) The frame number

   Frame = 0

5) The offset within the frame

   Offset = 20

6) The physical address (in decimal and binary)

   The physical address is therefore 20 in decimal: 20 = 0*4096 + 20

   The physical address is expressed in 16 bits (64KB = 2^{16}), including 4 bits for the frame number and 12 bits for the offset within the frame (4KB = 2^{12}), which gives us 0000 0000 0001 0100 in binary.

   0000 : no. frame 0000 0001 0100 : offset

Exercise n°3

We consider a system with 64 KB of physical memory managed in a segmented and paged way. Each process can use 16 segments of 1 KB and the system supports up to 256 processes. The page frames are 512 Bytes in size.

1) What is the size of the physical address?

   16 bits (2^{16} = 64K).

2) What is the size of the logical address?

   16 segments of 1KB = 16KB, so 14 bits (2^{14} = 16K): 4 bits for the segment number, 10 for the offset.

3) What is the size of the virtual memory?

   256 processes of 16 KB = 4 MB of virtual memory.

4) How many pages can a process use at most? Justify your answer.

$$\underline{\boldsymbol{Number\ of\ pages} = \frac{\boldsymbol{Process\ size}}{\boldsymbol{Page\ size}} = \frac{\boldsymbol{16\ KB}}{\boldsymbol{512\ Bytes}} = \boldsymbol{32}}$$

   Each process can have a maximum of 32 pages of 512 bytes (32 × 512 = 16 KB max per process).

Exercise n°4

We consider a memory managed using the "pagination on demand" technique with 3 frames and a sequence of page references in this order: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2. The frames are initially empty.

Calculate the number of page faults generated by the FIFO, OPTIMAL, LRU, LFU and SECOND CHANCE replacement algorithms.

Solution:

FIFO algorithm

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all the pages in memory in a queue, with the oldest page at the head of the queue. When a page requires to be replaced, the page at the head of the queue is selected to be removed.

	7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**
F0	**7**	7	7	**2**	2	2	2	**4**	4	4	**0**	0	0
F1		**0**	0	0	0	**3**	3	3	**2**	2	2	2	2
F2			**1**	1	1	1	**0**	0	0	**3**	3	3	3
Page default	x	x	x	x		x	x	x	x	x	x		

$$NumberNumber\ of\ page\ faults = 10$$

$$Page\ fault\ rate = \frac{NumberNumber\ of\ page\ faults}{Number\ of\ pages} = \frac{10}{13} * 100 = \mathbf{76.92}\%$$

LRU (least recently used) algorithm

In this algorithm, the page will be replaced by the one that has been used least recently.

	7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**
F0	**7**	7	7	**2**	2	2	2	**4**	4	4	**0**	0	0
F1		**0**	0	0	0	0	0	0	0	**3**	3	3	3
F2			**1**	1	1	**3**	3	3	**2**	2	2	2	2
Page default	x	x	x	x		x		x	x	x	x		

$$NumberNumber\ of\ page\ faults = 9$$

$$Page\ fault\ rate = \frac{NumberNumber\ of\ page\ faults}{Number\ of\ pages} = \frac{9}{13} * 100 = \mathbf{69.23}\ \%$$

Optimal algorithm

In this algorithm, pages that are not used for the longest time in the future are replaced. This is not a realistic algorithm.

	7	0	1	2	0	3	0	4	2	3	0	3	2
F0	7	7	7	2	2	2	2	2	2	2	2	2	2
F1		0	0	0	0	0	0	4	4	4	0	0	0
F2			1	1	1	3	3	3	3	3	3	3	3
Page default	x	x	x	x		x		x			x		

$$NumberNumber\ of\ page\ faults = 7$$

$$Page\ fault\ rate = \frac{NumberNumber\ of\ page\ faults}{Number\ of\ pages} = \frac{7}{13} * 100 = \mathbf{53.84}\ \%$$

LFU (least frequency used) algorithm

In the LFU page replacement algorithm, the page with the minimum number of frequencies is selected to be replaced by the page that is to enter the system.

	7	0	1	2	0	3	0	4	2	3	0	3	2
F0	7	7	7	2	2	2	2	4	4	3	3	3	3
F1		0	0	0	0	0	0	0	0	0	0	0	0
F2			1	1	1	3	3	3	2	2	2	2	2
Page default	x	x	x	x		x		x	x	x			

$$NumberNumber\ of\ page\ faults = 8$$

$$Page\ fault\ rate = \frac{NumberNumber\ of\ page\ faults}{Number\ of\ pages} = \frac{8}{13} * 100 = 61.53\ \%$$

Second chance

	7	0	1	2	0	3	0	4	2	3	0	3	2	7
F0	7(0)	7(0)	7(0)	2(0)	2(0)	2(0)	2(0)	4(0)	4(0)	4(0)	0(0)	0(0)	0(0)	7(0)
F1		0(0)	0(0)	0(0)	0(1)	0(0)	0(1)	0(0)	2(0)	2(0)	2(0)	2(0)	2(1)	2(0)
F2			1(0)	1(0)	1(0)	3(0)	3(0)	3(0)	3(0)	3(1)	3(0)	3(1)	3(1)	3(0)
Page default	x	x	x	x		x		x	x		x			x

$$NumberNumber\ of\ page\ faults = 8$$

$$Page\ fault\ rate = \frac{NumberNumber\ of\ page\ faults}{Number\ of\ pages} = \frac{9}{13} * 100 = 69.23\ \%$$

Second chance algorithm with FIFO

A modified form of the FIFO page replacement algorithm, known as the second chance page replacement algorithm, performs relatively better than FIFO.

It works by looking at the start of the queue as FIFO does, but instead of immediately replacing that page, it checks to see if its referenced bit is 1. If it is not 1, the page is replaced, otherwise, the referenced bit is reset to 0 and the page will only be replaced.

	2	**3**	**2**	**1**	**5**	**2**	**4**	**5**	**3**	**2**	**3**	**5**	**5**
F0	**2(0)**	2(0)	2(1)	2(1)	2(0)	2(1)	2(0)	2(0)	**3(0)**	3(0)	3(1)	3(1)	3(1)
F1		**3(0)**	3(0)	3(0)	**5(0)**	5(0)	5(0)	5(1)	5(1)	5(0)	5(0)	5(1)	5(1)
F2				**1(0)**	1(0)	1(0)	**4(0)**	4(0)	4(0)	**2(0)**	2(0)	2(0)	2(0)
Page default	x	x		x	x		x		x	x			

$$NumberNumber\ of\ page\ faults = 7$$

$$Page\ fault\ rate = \frac{NumberNumber\ of\ page\ faults}{Number\ of\ pages} = \frac{7}{13} * 100 = 53.84\ \%$$