Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Part 4: Memory management

Seminar 11
Virtual memory exercises

fSO

DISCA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Contenido

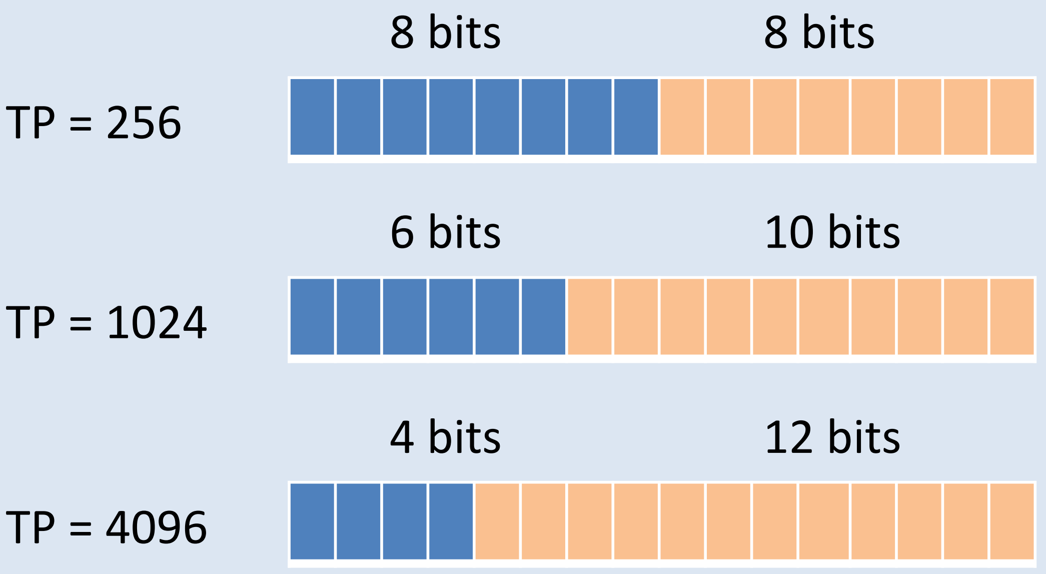
Exercise 1. Paging without virtual memory

A processor has **16-bit logical addresses** managed by paging. It is made in three versions with page sizes **of 256, 1024 y 4096 bytes**, respectively.
A given executable file contains 2800 bytes of instructions from address 0x1000, 1198 bytes of data from address 0x3000 and reserves initially 2048 bytes for the stack from address 0x9000.

a) Compute the number of page table entries and the initial number of pages in use for every processor model

Page size (bytes)	Number of page table entries	Initial number of pages
256		
1024		
4096		

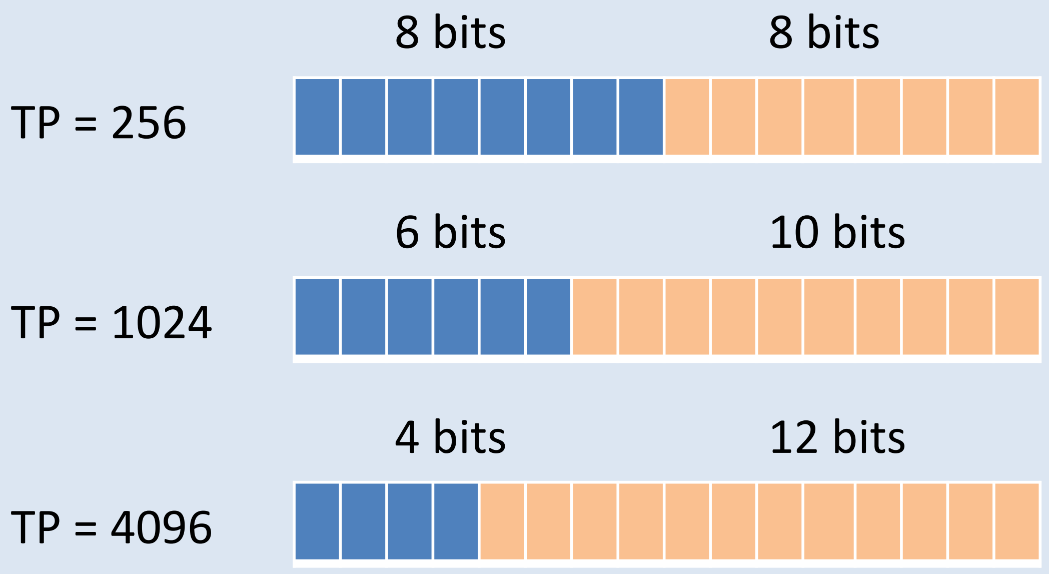
Exercises 1: Paging without virtual memory



region	size (bytes)	Base address
code	2800	0x1000
data	1198	0x3000
stack	2048	0x9000

Page size (bytes)	Number of page table entries	Initial number of pages
256	$2^8 = 256$	
1024	$2^6 = 64$	
4096	$2^4 = 16$	

Exercises 1: Paging without virtual memory



region	size (bytes)	Base address
code	2800	0x1000
data	1198	0x3000
stack	2048	0x9000

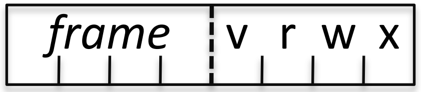
Page size (bytes)	Number of page table entries	Initial number of pages
256	$2^8 = 256$	ceil(2800/256) + ceil(1198/256) +ceil(2048/256) = **24**
1024	$2^6 = 64$	ceil(2800/1024) + ceil(1198/1024) +ceil(2048/1024) = **7**
4096	$2^4 = 16$	ceil(2800/4096) + ceil(1198/4096) +ceil(2048/4096) = **3**

Exercise 1 (cont.)

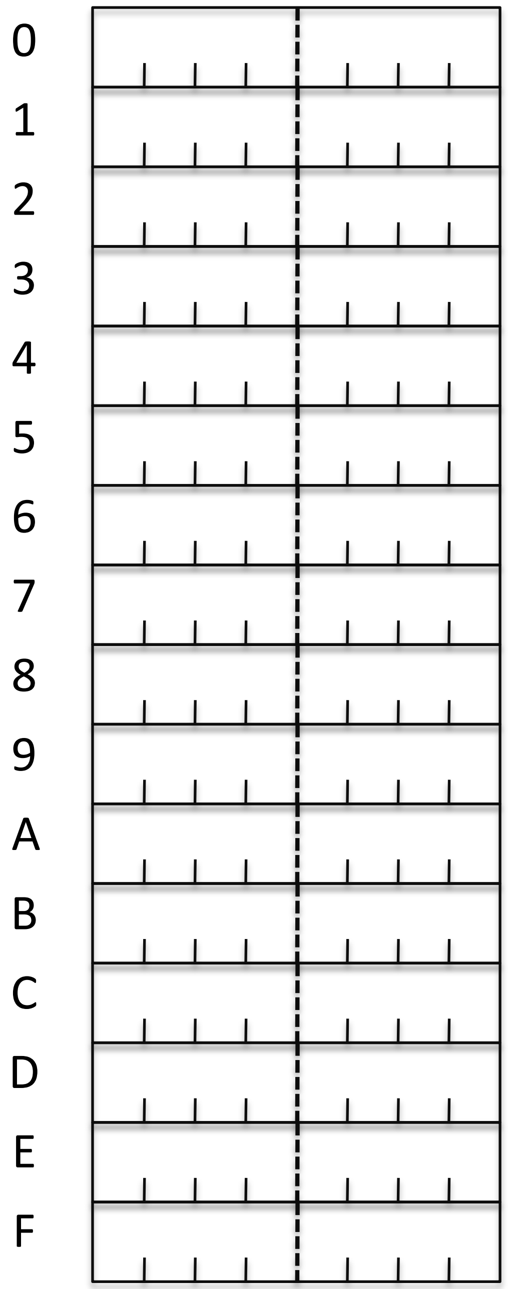
fSO

Region	Size (bytes)	Base (hex)
Code	2800	1000
Variables	1198	3000
Stack	2048	9000

b) Build *in binary* the initial process page table when the program is executed with a **page size of 4096 bytes**.
Consider that **physical memory size is 64 KBytes** and **when the process is loaded only the upper frames are allocated**. The OS allocates frames in ascending order of physical addresses. **It allocates first the code, then variables and finally the stack**.
Page descriptors have the following content:

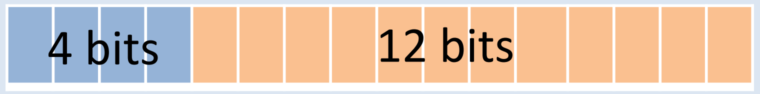


If a page is not in use it has to be indicated by r=w=x=0

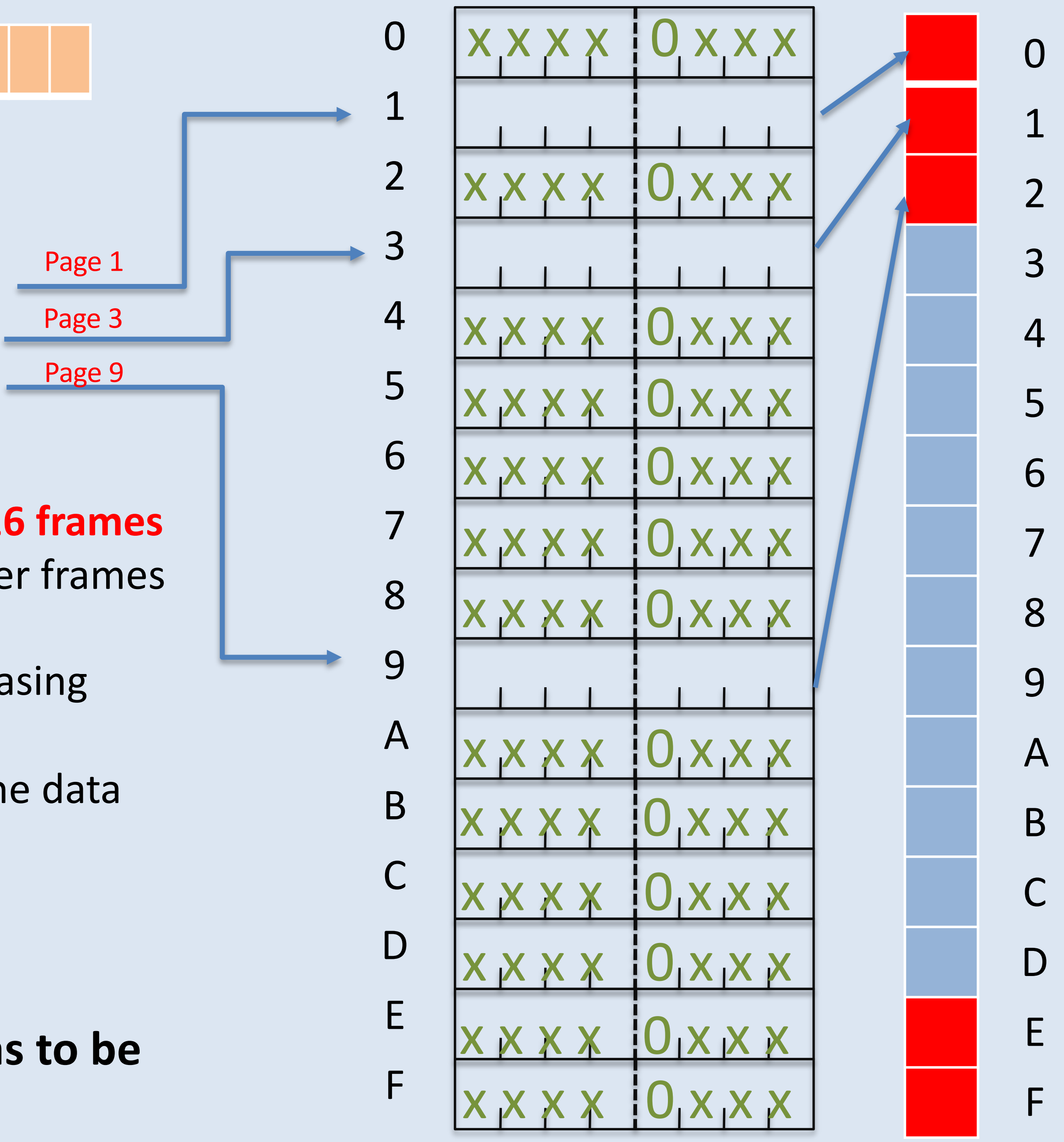


Exercise 1 (cont.)

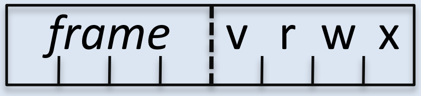
TP = 4096



region	size (bytes)	base address
code	2800	0x1000
data	1198	0x3000
stack	2048	0x9000



Physical memory => **64KB** => **16 frames**
At the beginning, the two upper frames are occupied (OS).
The OS assigns frames in increasing order of physical addresses.
First it places the code, then the data and finally the stack.



If a page is not in use it has to be indicated by r=w=x=0

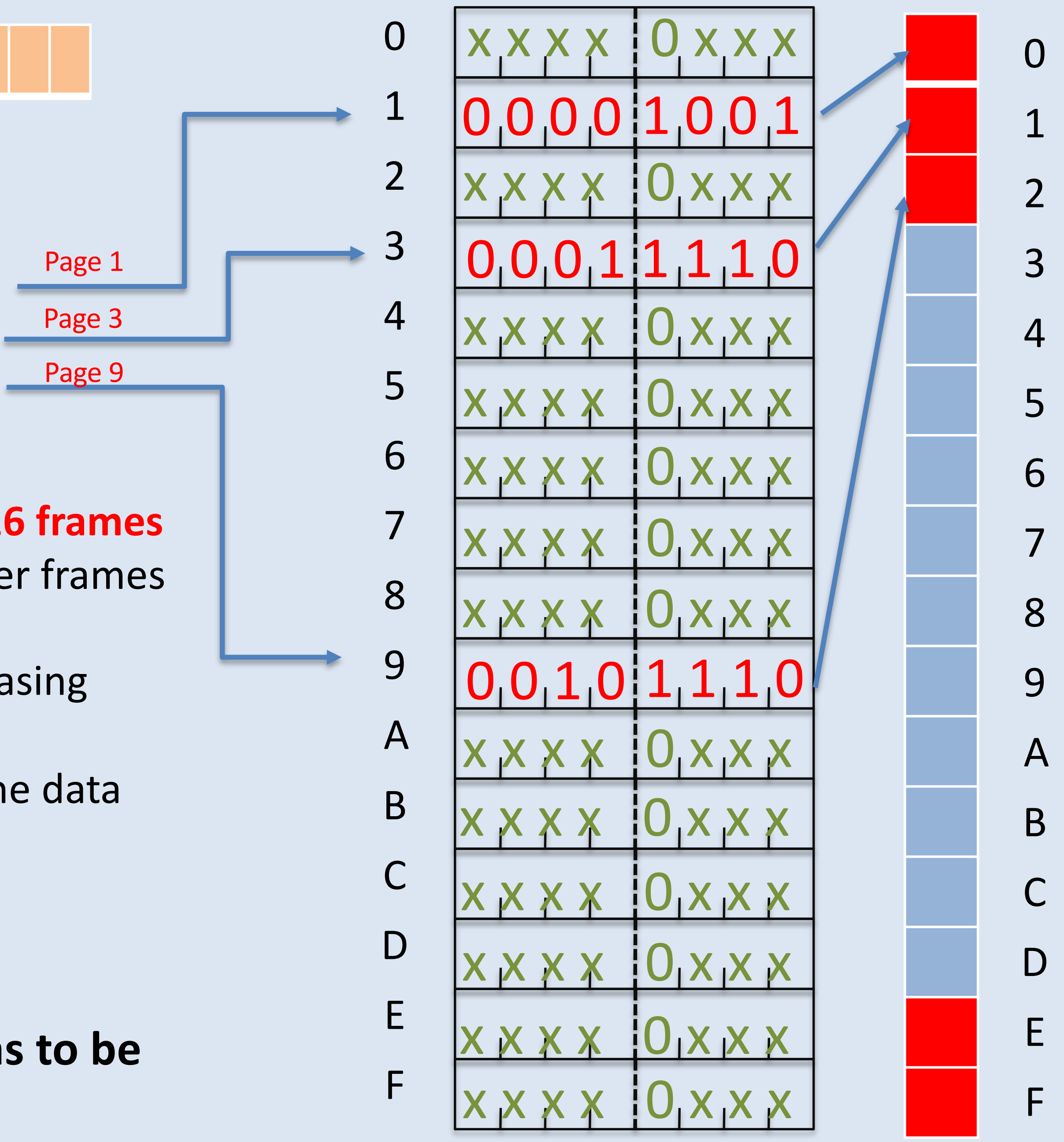
Exercise 1 (cont.)

fSO

TP = 4096

4 bits	12 bits

region	size (bytes)	base address
code	2800	0x1000
data	1198	0x3000
stack	2048	0x9000



Physical memory => **64KB** => **16 frames**
At the beginning, the two upper frames are occupied (OS).
The OS assigns frames in increasing order of physical addresses.
First it places the code, then the data and finally the stack.

frame	v	r	w	x

If a page is not in use it has to be indicated by r=w=x=0

Exercise 1 (cont.)

c) Compute, if possible, the physical addresses that the MMU generates for every of the following accesses. If the translation is not possible, tell why.

Region	Size (bytes)	Base (hex)
Code	2800	1000
Variables	1198	3000
Stack	2048	9000

Access type	Logical address	Physical address	Legal access? (yes/no)
Instruction read	1000		
Instruction read	1004		
Instruction read	2000		
Instruction read	3000		
Variable read	3010		
Variable read	9010		
Variable write	1000		
Variable write	5000		

Exercise 1 (cont.)

c) Compute, if possible, the physical addresses that the MMU generates for every of the following accesses. If the translation is not possible, tell why.

Region	Size (bytes)	Base (hex)
Code	2800	1000
Variables	1198	3000
Stack	2048	9000

Access type	Logical address	Physical address	Legal access? (yes/no)
Instruction read	0x1000	0x0000	Legal
Instruction read	0x1004	0x0004	Legal
Instruction read	0x2000		Page 2 does not exist
Instruction read	0x3000		Page 3. Data not exec
Data read	0x3010	0x1010	Legal
Data read	0x9010	0x2010	Legal
Data write	0x1000		Ilegal. Code. Write not allowed
Data write	0x5000		Page 5 does not exist

Exercise 2. Paging with virtual memory

Consider the processor from exercise 1 with **4-KByte pages**, and an OS with **virtual memory**. The process starts without any frame allocated and the OS allocates free frames in the following order: frame 0, frame 1, frame 2, etc.

Region	Size (bytes)	Base (hex)
Code	2800	1000
Data	1198	3000
Stack	2048	9000

a) Complete the next table considering that logical addresses are emitted following the "Logical address" column order.

Access type	Logical address (hex)	Physical address (hex)	Page fault?	Legal access?
Instruction read	0x1000			
Instruction read	0x1004			
Variable read	0x97FC			
Instruction read	0x1008			
Variable write	0x97F8			
Instruction read	0x5000			

¿How many frames has the process allocated after the fifth access?
¿Can the process continue after the last access?

Exercise 2. Paging with virtual memory

Consider the processor from exercise 1 with **4-KByte pages**, and an OS with **virtual memory**. The process starts without any frame allocated and the OS allocates free frames in the following order: frame 0, frame 1, frame 2, etc.

Region	Size (bytes)	Base (hex)
Code	2800	1000
Data	1198	3000
Stack	2048	9000

a) Complete the next table considering that logical addresses are emitted following the "Logical address" column order.

Access type	Logical address (hex)	Physical address (hex)	Page fault?	Legal access?
Instruction read	0x1000	0x0000	Yes. Frame 0	Yes
Instruction read	0x1004			
Variable read	0x97FC			
Instruction read	0x1008			
Variable write	0x97F8			
Instruction read	0x5000			

¿How many frames has the process allocated after the fifth access?

¿Can the process continue after the last access?

Exercise 2. Paging with virtual memory

Consider the processor from exercise 1 with **4-KByte pages**, and an OS with **virtual memory**. The process starts without any frame allocated and the OS allocates free frames in the following order: frame 0, frame 1, frame 2, etc.

Region	Size (bytes)	Base (hex)
Code	2800	1000
Data	1198	3000
Stack	2048	9000

a) Complete the next table considering that logical addresses are emitted following the "Logical address" column order.

Access type	Logical address (hex)	Physical address (hex)	Page fault?	Legal access?
Instruction read	0x1000	0x0000	Yes. Frame 0	Yes
Instruction read	0x1004	0x0004	No	Yes
Variable read	0x97FC	0x17FC	Yes. Frame 1	Yes
Instruction read	0x1008	0x0008	No	Yes
Variable write	0x97F8	0x17F8	No	Yes
Instruction read	0x5000	---	YES	NO

¿How many frames has the process allocated after the fifth access? **2 frames**

¿Can the process continue after the last access? **No. It produces an exception and finishes**

Exercise 2 (cont.)

b) Describe *in hexadecimal* the evolution of the process page table.
Consider that physical memory size is 64 KBytes and with the process is loaded only the upper frames are allocated. The OS allocates frames in ascending order of physical addresses.
Page descriptors are 16-bit and have the following content:

0	0	0	0	*frame*	v	r	w	x	0	0	0	m

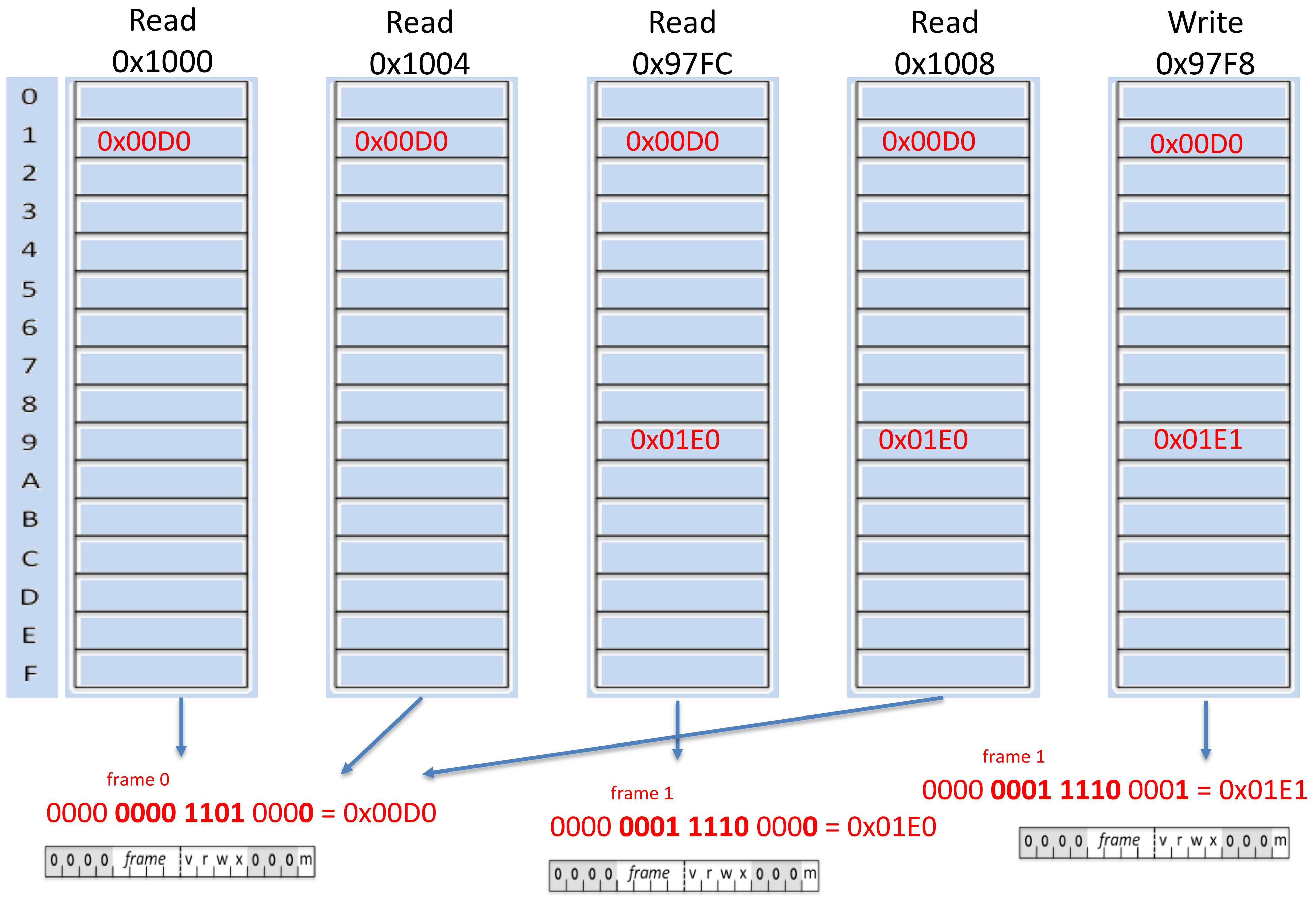
If a page is not in use it has to be indicated by r=w=x=0

	Initial state
0	0 ? 0 0
1	0 ? 5 0
2	0 ? 0 0
3	0 ? 6 0
4	0 ? 0 0
5	0 ? 0 0
6	0 ? 0 0
7	0 ? 0 0
8	0 ? 0 0
9	0 ? 6 0
A	0 ? 0 0
B	0 ? 0 0
C	0 ? 0 0
D	0 ? 0 0
E	0 ? 0 0
F	0 ? 0 0

Exercise 2 (cont.)

	Read 0x1000	Read 0x1004	Read 0x97FC	Read 0x1008	Write 0x97F8
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
A					
B					
C					
D					
E					
F					

Ejercicio 2. Apartado 2


Read 0x1000
Read 0x1004
Read 0x97FC
Read 0x1008
Write 0x97F8
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
0x00D0
0x00D0
0x00D0
0x01E0
0x00D0
0x01E0
0x00D0
0x01E1
frame 0
0000 **0000 1101** 0000 = 0x00D0
0 0 0 0 frame v r w x 0 0 0 m
frame 1
0000 **0001 1110** 0000 = 0x01E0
0 0 0 0 frame v r w x 0 0 0 m
frame 1
0000 **0001 1110** 0001 = 0x01E1
0 0 0 0 frame v r w x 0 0 0 m

Exercise 3. Replacement algorithms

In a computer with **32 MB of main memory, with a memory** management policy of **paging with 4KB page size, the OS** assigns to process A **4 main memory frames (from 0 to 3)**, that are initially empty.

Answer the following items:

a) Describe the **physical memory address format.**

b) If process A generates the following logical address sequence (shown in hexadecimal):

0x02D4B8, 0x02D4B9, 0x02D4EB, 0x02D4EB, 0x02D86F, 0xF0B621, 0xF0B815, 0xF05963, 0xF0B832, 0xF0BA23, 0xD946C3, 0xD9B1A7, 0xD9B1A1, 0xF0BA25, 0x02D4C7, 0x628A31, 0xF0B328, 0xD9B325, 0xD73425

Obtain, for **FIFO and LRU algorithms, how many page faults are** generated and the final main memory state, telling the page allocated in every frame assigned to the process.

Exercise 3. Replacement algorithms

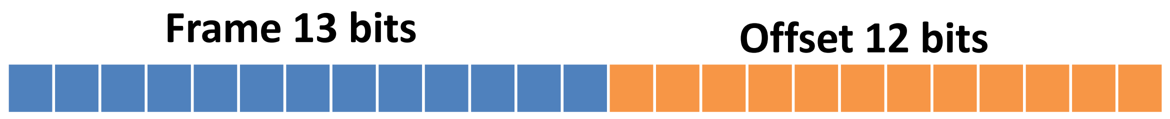
In a computer with **32 MB of main memory, with a memory** management policy of **paging with 4KB page size, the OS** assigns to process A **4 main memory frames (from 0 to 3)**, that are initially empty.

Answer the following items:

a) Describe the **physical memory address format.**

Memory = 32MB => 25 bits

Frame size = 4K => 12 bits

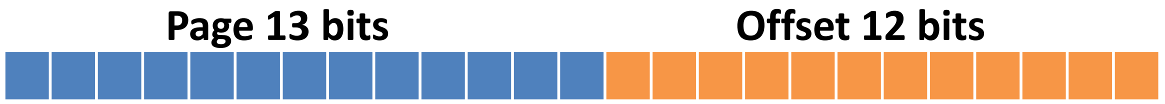


b) If process A generates the following logical address sequence:

L. Address	Page	Offset	L. Address	Page	Offset
0x02D4B8	**0x02D**	0x4B8	0xD946C3	**0xD94**	0x6C3
0x02D4B9	**0x02D**	0x4B9	0xD9B1A7	**0xD9B**	0x1A7
0x02D4EB	**0x02D**	0x4EB	0xD9B1A1	**0xD9B**	0x1A1
0x02D4EB	**0x02D**	0x4EB	0xF0BA25	**0xF0B**	0xA25
0x02D86F	**0x02D**	0x86F	0x02D4C7	**0x02D**	0x4C7
0xF0B621	**0xF0B**	0x621	0x628A31	**0x628**	0xA31
0xF0B815	**0xF0B**	0x815	0xF0B328	**0xF0B**	0x328
0xF05963	**0xF05**	0x963	0xD9B325	**0xD9B**	0x325
0xF0B832	**0xF0B**	0x832	0xD73425	**0xD73**	0x425
0xF0BA23	**0xF0B**	0xA23			

02D, F0B, F05, F0B, D94, D9B, F0B, 02D, 628, F0B, D9B, D73

Exercise 3. Replacement algorithms



Page size = 4K => 12 bits

Secuencia de referencias a páginas: **02D, F0B, F05, F0B, D94, D9B, F0B, 02D, 628, F0B, D9B, D73**

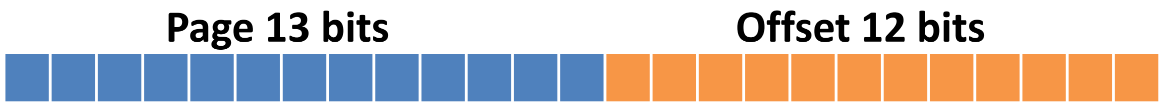
FIFO

Frame	02D	F0B	F05	F0B	D94	D9B	F0B	02D	628	F0B	D9B	D73
0												
1												
2												
3												

LRU

Frame	02D	F0B	F05	F0B	D94	D9B	F0B	02D	628	F0B	D9B	D73
0												
1												
2												
3												

Exercise 3. Replacement algorithms



Page size = 4K => 12 bits

Secuencia de referencias a páginas: **02D, F0B, F05, F0B, D94, D9B, F0B, 02D, 628, F0B, D9B, D73**

FIFO

Frame	02D	F0B	F05	F0B	D94	D9B	F0B	02D	628	F0B	D9B	D73
0	**02D**	02D	02D	02D	02D	**D9B**	D9B	D9B	D9B	D9B	D9B	**D73**
1		**F0B**	F0B	F0B	F0B	F0B	F0B	**02D**	02D	02D	02D	02D
2			**F05**	F05	F05	F05	F05	F05	**628**	628	628	628
3					**D94**	D94	D94	D94	D94	**F0B**	F0B	F0B

LRU

Frame	02D	F0B	F05	F0B	D94	D9B	F0B	02D	628	F0B	D9B	D73
0	**02D**	02D	02D	02D	02D	**D9B**	D9B	D9B	D9B	D9B	D9B	D9B
1		**F0B**	F0B	F0B	F0B	F0B	F0B	F0B	F0B	F0B	F0B	F0B
2			**F05**	F05	F05	F05	F05	**02D**	02D	02D	02D	**D73**
3					**D94**	D94	D94	D94	**628**	628	628	628

head	02D	F0B	F05	F0B	D94	D9B	F0B	02D	628	F0B	D9B	D73
stack		02D	F0B	F05	F0B	D94	D9B	F0B	02D	628	F0B	F0B
			02D	02D	F05	F0B	D94	D9B	F0B	02D	628	628
tail					02D	F05	F05	D94	D9B	D9B	02D	

Exercise 4. Replacement scope (intro)

- There are two possible page replacement scopes:
  - **Local replacement:**
    - A process selects the victim between its own pages allocated into main memory frames, it can not take frames from another process.
    - The number of process allocated frames doesn’t change
  - **Global replacement:**
    - A process selects the victim between whole set of main memory frames
    - The victim can belong to another process different from the one that produces the page fault

Exercise 4. Replacement scope

On a virtual memory system, with 1024 byte page size, the OS has allocated 6 frames (from 0 to 5) to two processes A y B. At time t = 10, A and B page tables have the following content:

Process A page table

	Frame	Valid bit	Counter
0		i	
1		i	
2	2	v	10
3	5	v	3
4		i	
5	4	v	5
6		i	
7		i	

Process B page table

	Frame	Valid bit	Counter
0		i	
1		i	
2		i	
3	1	v	2
4		i	
5		i	
6		i	
7		i	

Then the processes emit the following logical address sequence. Consider that all the addresses are legal:

A,100; A,4000; B,100; A,7000; B,2100; B,1028; A,5800; A,100

Obtain what pages are allocated on every frame and the physical address translation of the first and the last access in the following situations:

a) The replacement algorithm is **LRU** with **global scope**

b) The replacement algorithm is **LRU** with **local scope**. Process A has 4 frames and process B has 2 frames

Exercise 4. Replacement scope

On a virtual memory system, with 1024 byte page size, the OS has allocated 6 frames (from 0 to 5) to two processes A y B. At time t = 10, A and B page tables have the following content:

Process A page table

	Frame	Valid bit	Counter
0		i	
1		i	
2	2	v	10
3	5	v	3
4		i	
5	4	v	5
6		i	
7		i	

Process B page table

	Frame	Valid bit	Counter
0		i	
1		i	
2		i	
3	1	v	2
4		i	
5		i	
6		i	
7		i	

a) The replacement algorithm is **LRU** with **global scope**

	A	A	B	A	B	B	A	A
L.Add	100	4000	100	7000	2100	1028	5800	100
Page	0	3	0	6	2	1	5	0
offset	100	928	100	856	52	4	680	100

frame	t = 10	11	12	13	14	15	16	17	18
0	**free**	**A0, 11**	**A0, 11**	**A0, 11**	**A0, 11**	**A0, 11**	**A0, 11**	**A5, 17**	**A5, 17**
1	**B3, 2**	**B3, 2**	**B3, 2**	**B3, 2**	**A6, 14**	**A6, 14**	**A6, 14**	**A6, 14**	**A6, 14**
2	**A2, 10**	**A2, 10**	**A2, 10**	**A2, 10**	**A2, 10**	**A2, 10**	**B1, 16**	**B1, 16**	**B1, 16**
3	**free**	**free**	**free**	**B0, 13**	**B0, 13**	**B0, 13**	**B0, 13**	**B0, 13**	**B0, 13**
4	**A5, 5**	**A5, 5**	**A5, 5**	**A5, 5**	**A5, 5**	**B2, 15**	**B2, 15**	**B2, 15**	**B2, 15**
5	**A3, 3**	**A3, 3**	**A3, 12**	**A3, 12**	**A3, 12**	**A3, 12**	**A3, 12**	**A3, 12**	**A0, 18**

Exercise 4. Replacement scope

On a virtual memory system, with 1024 byte page size, the OS has allocated 6 frames (from 0 to 5) to two processes A y B. At time t = 10, A and B page tables have the following content:

Process A page table

	Frame	Valid bit	Counter
0		i	
1		i	
2	2	v	10
3	5	v	3
4		i	
5	4	v	5
6		i	
7		i	

Process B page table

	Frame	Valid bit	Counter
0		i	
1		i	
2		i	
3	1	v	2
4		i	
5		i	
6		i	
7		i	

a) The replacement algorithm is **LRU** with **global local**

	A	A	B	A	B	B	A	A
L.Add	100	4000	100	7000	2100	1028	5800	100
Page	0	3	0	6	2	1	5	0
offset	100	928	100	856	52	4	680	100

frame	t = 10	11	12	13	14	15	16	17	18
0	**free**	**A0, 11**	**A0, 11**	**A0, 11**	**A0, 11**	**A0, 11**	**A0, 11**	**A0, 11**	**A0, 18**
1	**B3, 2**	**B3, 2**	**B3, 2**	**B3, 2**	**B3, 2**	**B2, 15**	**B2, 15**	**B2, 15**	**B2, 15**
2	**A2, 10**	**A2, 10**	**A2, 10**	**A2, 10**	**A2, 10**	**A2, 10**	**A2, 10**	**A5, 17**	**A5, 17**
3	**free**	**free**	**free**	**B0, 13**	**B0, 13**	**B0, 13**	**B1, 16**	**B1, 16**	**B1, 16**
4	**A5, 5**	**A5, 5**	**A5, 5**	**A5, 5**	**A6, 14**	**A6, 14**	**A6, 14**	**A6, 14**	**A6, 14**
5	**A3, 3**	**A3, 3**	**A3, 12**	**A3, 12**	**A3, 12**	**A3, 12**	**A3, 12**	**A3, 12**	**A3, 12**