Fundamentos de los Sistemas Operativos (FSO)

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

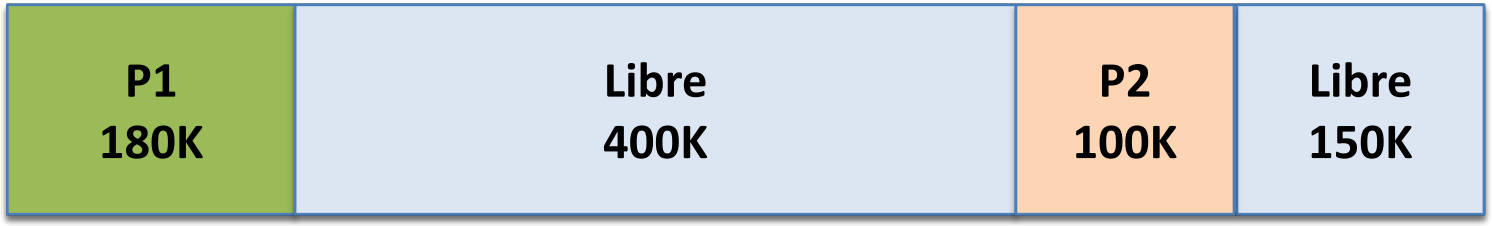
Part 4: Memory Management
Seminar Unit 10

SUT10:
Contiguous and Sparse Memory Allocation Problems

fSO

- **Exercise 1: Contiguous Allocation**
  - **Exercise 1.1: Variable Partitions**
  - **Exercise 1.2: Base and limit registration**
  - **Exercise 1.3: Degree of Multiprogramming**
- Exercise 2: Sparse Allocation
  - Exercise 2.1: Formatting logical and physical addresses.
  - Exercise 2.2: From logical to physical addresses
  - Exercise 2.3: From logical to physical addresses
  - Exercise 2.4: From physical to logical addresses
  - Exercise 2.5: From physical to logical addresses
- Exercise 3: Table Sizes and Fragmentation
  - Exercise 3.1: Table size
  - Exercise 3.2: Internal Fragmentation

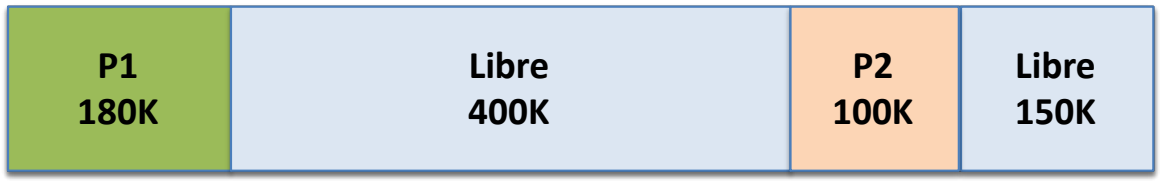
- A system is organised with a memory management with multiple partitions of variable size with compaction. At a given moment, the following memory allocation is produced:



- In the job queue we have in this order: P4(120K), P5(200K) and P6(80K), which must be attended in FIFO order. Supposing that no process is finished and after trying to load in memory all the processes that are in the queue. Specify next answer if the allocation technique is **First Fit, Best it and Worst Fit.**

1. Indicate how many partitions are free and their sizes
2. If compaction is applied in this situation, indicate which processes should be moved so that the number of Kbytes handled is as small as possible and a single gap remains.
3. If the base records of each process are, respectively, B1, B2, B3, B4, B5 and B6, indicate how the base records corresponding to the process or processes that have been moved due to compaction have changed

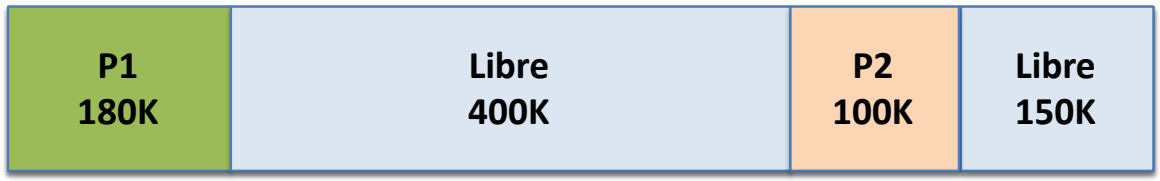
- **First Fit**



	Offset	Size	Limit
P1	0	180	180
EMPTY	180	400	580
P2	580	100	680
EMPTY	680	150	830

	Offset	Size	Limit
P1	0	180	180
P4	180	120	300
P5	300	200	500
P6	500	80	580
P2	580	100	680
EMPTY	680	150	830

- **Best Fit**



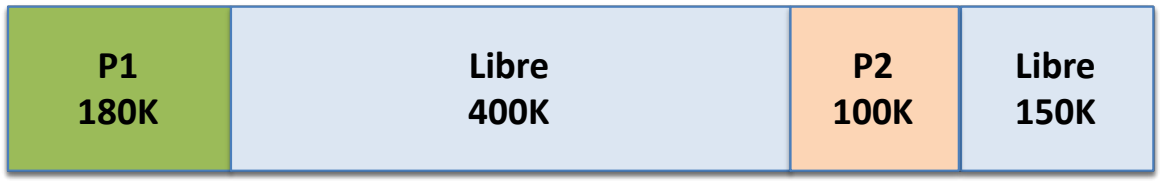
	Offset	Size	Limit
P1	0	180	180
EMPTY	180	400	580
P2	580	100	680
EMPTY	680	150	830

	Offset	Size	Limit
P1	0	180	180
P5	180	200	380
P6	380	80	460
EMPTY	460	120	580
P2	580	100	680
P4	680	120	800
EMPTY	800	30	830

modified

	Offset	Size	Limit
P1	0	180	180
P5	180	200	380
P6	380	80	460
P4	**460**	**120**	**580**
P2	580	100	680
EMPTY	**680**	**150**	**830**

- **Worst Fit**



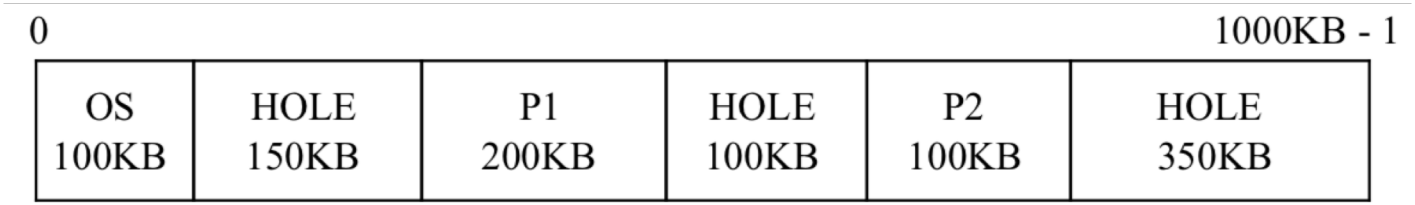
	Offset	Size	Limit
P1	0	180	180
EMPTY	180	400	580
P2	580	100	680
EMPTY	680	150	830

	Offset	Size	Limit
P1	0	180	180
P4	180	120	300
P5	300	200	500
EMPTY	500	80	580
P2	580	100	680
P6	680	80	760
EMPTY	760	70	830

modified

	Offset	Size	Limit
P1	0	180	180
P4	180	120	300
P5	300	200	500
P6	**500**	**80**	**580**
P2	580	100	680
EMPTY	**680**	**150**	**830**

- A system manages its 1000 Kilobyte main memory with contiguous allocation with variable partitions and YES/NO compaction (coalescence). The allocation algorithm always allocates processes within a hole by adjusting to the lower addresses (left), leaving the higher hole addresses free.
- The initial state of main memory if the following:



- **a)** Indicate the **base address** for processes P3 (100KBytes), P4 (400KBytes) and P5 (200KBytes) applying the allocation algorithms: Best Fit, First Fit and Worst Fit. Consider the proposed sequence of events and the former initial memory state on each case. If a process cannot be allocated write DOESN'T FIT and continue with the following process.

  Sequence: P3 arrives; P1 ends; P4 arrives; P2 ends; P5 Arrives

- **b)** Indicate for Worst Fit algorithm, the remaining gaps (**start address** and **size**) at the end of the previous sequence of events, and the type of fragmentation generated.

- A system manages its 1000 Kilobyte main memory with contiguous allocation with variable partitions and YES/NO compaction (coalescence). The allocation algorithm always allocates processes within a hole by adjusting to the lower addresses (left), leaving the higher hole addresses free.

a) Indicate the **base address** for processes P3 (100KBytes), P4 (400KBytes) and P5 (200KBytes) applying the allocation algorithms: Best Fit, First Fit and Worst Fit. Consider the proposed sequence of events and the former initial memory state on each case. If a process cannot be allocated write FAIL and continue with the following process.

	Offset	size	limit
OS	0	100	100
Empty	100	150	250
P1	250	200	450
Empty	450	100	550
P2	550	200	750
Empty	750	250	1000

Coalescence = YES

Event	FF	holes FF	BF		WF	
Initial state						
P3 arrives						
P1 ends						
P4 arrives						
P2 ends						
P5 arrives						
Initial state						
P3 arrives						
P1 ends						
P4 arrives						
P2 ends						
P5 arrives						

Coalescence = NO

- A system manages its 1000 Kilobyte main memory with contiguous allocation with variable partitions and YES/NO compaction (coalescence). The allocation algorithm always allocates processes within a hole by adjusting to the lower addresses (left), leaving the higher hole addresses free.

a) Indicate the **base address** for processes P3 (100KBytes), P4 (400KBytes) and P5 (200KBytes) applying the allocation algorithms: Best Fit, First Fit and Worst Fit. Consider the proposed sequence of events and the former initial memory state on each case. If a process cannot be allocated write FAIL and continue with the following process.

	Offset	size	limit
OS	0	100	100
Empty	100	150	250
P1	250	200	450
Empty	450	100	550
P2	550	200	750
Empty	750	250	1000

Coalescence = YES

Event	FF	holes FF	BF		WF	
Initial state		(100, 150), (450, 100), (750, 250)		(100, 150), (450, 100), (750, 250)		(100, 150), (450, 100), (750, 250)
P3 arrives	100	(200, 50), (750, 250)	450	(100, 150), (750, 250)	750	(100, 150), (450, 100), (850, 150)
P1 ends		(200, 350), (750, 250)		(100, 350), (750, 250)		(100, 450), (850, 150)
P4 arrives	Fail	(200, 350), (750, 250)	Fail	(100, 350), (750, 250)	100	(500, 50), (850, 150)
P2 ends		(200, 800)		(100, 350), (550, 450)		(500, 250), (850, 150)
P5 arrives	200	(400, 600)	100	(300, 150), (550, 450)	500	(700, 50), (850, 150)
Initial state		(100, 150), (450, 100), (750, 250)		(100, 150), (450, 100), (750, 250)		(100, 150), (450, 100), (750, 250)
P3 arrives	100	(200, 50), (450, 100), (750, 250)	450	(100, 150), (750, 250)	750	(100, 150), (450, 100), (850, 150)
P1 ends		(200, 50), (250, 200), (450, 100), (750, 250)		(100, 150), (250, 200), (750, 250)		(100, 150), (250, 200), (450, 100), (850, 150)
P4 arrives	Fail	(200, 50), (250, 200), (450, 100), (750, 250)	Fail	(100, 150), (250, 200), (750, 250)	Fail	(100, 150), (250, 200), (450, 100), (850, 150)
P2 ends		(200, 50), (250, 200), (450, 100), (550, 200), (750, 250)		(100, 150), (250, 200), (550, 200), (750, 250)		(100, 150), (250, 200), (450, 100), (550, 200), (850, 150)
P5 arrives	250	(200, 50), (450, 100), (550, 200), (750, 250)	250	(100, 150), (550, 200), (750, 250)	250	(100, 150), (450, 100), (550, 200), (850, 150)

Coalescence = NO

Be a multiple partitions system with **hardware support based** on the **base and limit register** technique.

Given a program P that occupies T words and is allocated in memory from the physical memory position C.

a) What is the value of each record for the program P?

b) What is the range of addresses that P emits?

c) What is the range of real addresses that P emits?

Ejercicio 1.2

Be a multiple partitions system with **hardware support based** on the **base and limit register** technique.

Given a program P that occupies T words and is allocated in memory from the physical memory position C.

a) What is the value of each record for the program P?

Base: C Limit: T

b) What is the range of addresses that P emits?

Logical addresses: 0 .. (T – 1)

c) What is the range of real addresses that P emits?

Physical addresses: C .. (C + T – 1)

A computer with a 32-bit processor address and 128 Mbyte of main memory the operating system, requiring 64 Mbytes (allocated in the lowest memory addresses), uses a variable partition memory manager. In this system the user processes have a minimum size of 16Kbytes. For this system indicate in a justified way:

a) The maximum size of the user processes that can be executed in that system.

b) The maximum degree of multiprogramming allowed.

c) Show an example where there are 2 processes (PA, PB) located in memory and it is necessary to make compaction to allocate a new PC process of 20 Mbytes.

Ejercicio 1.3

A computer with a 32-bit processor address and 128 Mbyte of main memory the operating system requiring 64 Mbytes (allocated in the lowest memory addresses) uses a variable partition memory manager. In this system the user processes have a minimum size of 16Kbytes. For this system indicate in a justified way:

a) The maximum size of the user processes that can be executed in that system.

There are 64 MB of user available memory. Variable partitions => whole process in memory. Maximum memory for a process is min(maxMemory, maxProcessMemory) = min(64 MB, 2^32) = 64MB

b) The maximum degree of multiprogramming allowed.

c) Show an example where there are 2 processes (PA, PB) located in memory and it is necessary to make compaction to allocate a new PC process of 20 Mbytes.

A computer with a 32-bit processor address and 128 Mbyte of main memory the operating system requiring 64 Mbytes (allocated in the lowest memory addresses) uses a variable partition memory manager. In this system the user processes have a minimum size of 16Kbytes. For this system indicate in a justified way:

a) The maximum size of the user processes that can be executed in that system.

b) The maximum degree of multiprogramming allowed.

The maximum degree corresponds to the maximum number of processes in memory. Minimum process size is 16KB. So, ceiling(64M / 16K) = 4K processes = 4096 processes

c) Show an example where there are 2 processes (PA, PB) located in memory and it is necessary to make compaction to allocate a new PC process of 20 Mbytes.

A computer with a 32-bit processor address and 128 Mbyte of main memory the operating system requiring 64 Mbytes (allocated in the lowest memory addresses) uses a variable partition memory manager. In this system the user processes have a minimum size of 16Kbytes. For this system indicate in a justified way:

a) The maximum size of the user processes that can be executed in that system.

b) The maximum degree of multiprogramming allowed.

c) Show an example where there are 2 processes (PA, PB) located in memory and it is necessary to make compaction to allocate a new PC process of 20 Mbytes.

There are many combinations. One of them:

|___PA__|_____|___PB__|_____|

14MB 18MB 14MB. 18MB

- Exercise 1: Contiguous Allocation
  - Exercise 1.1: Variable Partitions
  - Exercise 1.2: Base and limit registration
  - Exercise 1.3: Degree of Multiprogramming
- **Exercise 2: Sparse Allocation**
  - **Exercise 2.1: Formatting logical and physical addresses.**
  - **Exercise 2.2: From logical to physical addresses**
  - **Exercise 2.3: From logical to physical addresses**
  - **Exercise 2.4: From physical to logical addresses**
  - **Exercise 2.5: From physical to logical addresses**
- Exercise 3: Table Sizes and Fragmentation
  - Exercise 3.1: Table size
  - Exercise 3.2: Internal Fragmentation

Ejercicio 2.1

In a system with a Main Memory of 1GByte and a logical addressing size of 4GByte, the aim is to implement a system based on two-level paging, with the same number of bits for each level. The page size is 16KBytes and in all the page tables, each page descriptor requires 8Bytes.

- Specify the format of the logical address and the physical address by displaying their fields and bit numbers.
- Calculate the size of page tables

Ejercicio 2.1

In a system with a Main Memory of 1GByte and a logical addressing size of 4GByte, the aim is to implement a system based on two-level paging, with the same number of bits for each level. The page size is 16KBytes and in all the page tables, each page descriptor requires 8Bytes.

- Specify the format of the logical address and the physical address by displaying their fields and bit numbers.

Physical address: Main memory: 1GByte 30 bits
Logical address: 4GByte => 32-bit
Page size: 16KB => 14 bits
First level number of bits == Second level number of bits
Logical address: 32bits => 9 (first level) + 9 (second level) + 14 (offset)
Physical address: 30bits => 16 (frame number) + 14 (offset)

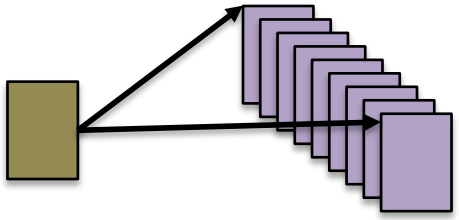
Ejercicio 2.1

In a system with a Main Memory of 1GByte and a logical addressing size of 4GByte, the aim is to implement a system based on two-level paging, with the same number of bits for each level. The page size is 16KBytes and in all the page tables, each page descriptor requires 8Bytes.

- Specify the format of the logical address and the physical address by displaying their fields and bit numbers.
- Calculate the size of page tables

1st level page table size: 2^9 descriptors * 8Bytes =2^{12} Bytes = 4KBytes
2nd level page table size: 2^9 descriptors * 8Bytes =2^{12} Bytes = 4KBytes

In total, 1 1st level table + 512 2nd level tables can be generated



Ejercicio 2.2

- A CPU generates the following logical addresses: 612, 38 and (3,62). Specify the physical addresses according to the proposed memory management schemes. If it is not possible to generate this logical address or translate it, indicate ERROR.

a) **Variable partitions with base and limit register**

Partition table

Register base	Register Limit
150	220

b) **Paging with 128B of page size.**

Process table

0	1
1	4
2	2
3	5

c) **Segmentation**

Segment table

	Base	Limit
0	200	20
1	50	10
2	105	49
3	320	70

- A CPU generates the following logical addresses: 612, 38 and (3,62). Specify the physical addresses according to the proposed memory management schemes. If it is not possible to generate this logical address or translate it, indicate ERROR.

a) **Variable partitions with base and limit register**

Partition table

Register base	Register Limit
150	220

Logical address	**Physical address**	**Message**	**Comment**
612		ERROR	612 > Limit (220)
38	150 +38 =158		
(3, 62)		ERROR	format error

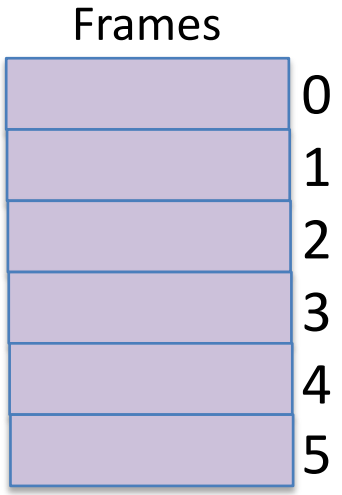
Ejercicio 2.2

- A CPU generates the following logical addresses: 612, 38 and (3,62). Specify the physical addresses according to the proposed memory management schemes. If it is not possible to generate this logical address or translate it, indicate ERROR.

b) **Paging with 128B of page size. Process table =>**

Process table

0	1
1	4
2	2
3	5



Logical address	Physical address	Message	Comment
612		ERROR	Process: 4 pages * 128 = 512 Bytes. Exceeds the limit
38	128 +38 =166		
(3, 62)		ERROR	format error

- A CPU generates the following logical addresses: 612, 38 and (3,62). Specify the physical addresses according to the proposed memory management schemes. If it is not possible to generate this logical address or translate it, indicate ERROR.

c) **Segmentation**, Process table =>

Segment table

	Base	Limit
0	200	20
1	50	10
2	105	49
3	320	70

Logical address	**Physical address**	**Message**	**Comment**
612		ERROR	format error
38		ERROR	format error
(3, 62)	320 + 62 = 382		

Ejercicio 2.3

- Below are scenarios in which one or more processes generate logical addresses. Specify the corresponding **physical addresses** according to each memory management scheme. If it is not possible, indicate **ERROR**.

a) **Variable partitions**.
Logical addresses :
B => 530,
A => (0,130),
C => 1046.

b) **Paging** with size of 256B.
Next logical adresses are generated by a unique process.
530,
(0,130),
1046.

c) **Segmentation** with 4 segments.
Next logical adresses are generated by a unique process.
530,
(0,130),
(1,25)
(4,50)

Partition table

Proceso	Registro Base	Registro Limite
A	0	1360
B	4020	6300
C	1400	2600

Page table

0	4
1	5
2	3
3	6

Segment table

	Base	Límite
0	200	20
1	50	50
2	105	49
3	320	70

Ejercicio 2.3

- Below are scenarios in which one or more processes generate logical addresses. Specify the corresponding **physical addresses** according to each memory management scheme. If it is not possible, indicate **ERROR**.

a) **Variable partitions**.

Logical addresses :

B => 530,

A => (0,130),

C => 1046.

Partition table

Proceso	Registro Base	Registro Limite
A	0	1360
B	4020	6300
C	1400	2600

Process	Logical address	Physical address	Message	Comment
B	530	4020 + 530 = 4550		
A	(0,130)		ERROR	format error
C	1046	1400 + 1046 = 2246		

Ejercicio 2.3

- Below are scenarios in which one or more processes generate logical addresses. Specify the corresponding **physical addresses** according to each memory management scheme. If it is not possible, indicate **ERROR**.

b) **Paging** with size of 256B.
Next logical adresses are generated by a unique process.

530,

(0,130),

1046.

Page table

0	4
1	5
2	3
3	6

Logical address	Physical address	Message	Comment
530	(3 * 256) + (530 mod 256) = 768 + 18= 786		Floor(530, 256) = 2
(0, 130)		ERROR	format error
1046		ERROR	Exceeds process size

Ejercicio 2.3

- Below are scenarios in which one or more processes generate logical addresses. Specify the corresponding **physical addresses** according to each memory management scheme. If it is not possible, indicate **ERROR**.

c) **Segmentation** with 4 segments. Next logical adresses are generated by a unique process.

530,

(0,130),

(1,25)

(4,50)

Segment table

	Base	Límite
0	200	20
1	50	50
2	105	49
3	320	70

Logical address	Physical address	Message	Comment
530		ERROR	format error
(0, 130)		ERROR	Exceeds segment size
(1, 25)	50 + 25 = 75		
(4, 50)		ERROR	Segment 4 does not exist

Ejercicio 2.4

A system with 64K bytes of physical memory, generates a physical address 27214 as a response of a logical address issued by the process P1. The size of P1 is 15535 bytes. Specify the logical addresses issued by the process considering:

1. A paging memory management scheme with 4K byte pages.
2. A segmentation memory management model with 16K byte segments. Assume that segments are always located from a multiple address of 16K bytes and that P1 is contained in a single segment.

A system with 64K bytes of physical memory, generates a physical address 27214 as a response of a logical address issued by the process P1. The size of P1 is 15535 bytes. Specify the logical addresses issued by the process considering:

1. A paging memory management scheme with 4K byte pages.

Physical address: 27214 mod 4096 generates an offset of 2638

This offset can be generated by any page of the process.
P1 size is 15535 => ceiling (15535 / 4094) = 4 pages are required

Logical addresses are: (n * 4096) + 2638 for n = 0 .. 3
2638, 6734, 10830 y 14926.

Ejercicio 2.4

A system with 64K bytes of physical memory, generates a physical address 27214 as a response of a logical address issued by the process P1. The size of P1 is 15535 bytes. Specify the logical addresses issued by the process considering:

2. A segmentation memory management model with 16K byte segments. Assume that segments are always located from a multiple address of 16K bytes and that P1 is contained in a single segment.

Segment size is 16KB.
Segments are allocated in multiple of 16K
P1 is contained in 1 segment that could be allocated in 0K, 16k, 32K, 48K
27214 is a physical address that is in segment 1 (16K).
27214 mod 16*1024 = 10830
Logical address: (0, 10830)

A memory management system uses an scheme based on **two-level paging technique**, with **1K page size**. The logical address is up to 4Mbytes and the first level page table consists of 1024 entries.

- Specify the possible logical address or addresses that can correspond to the physical address 2516.

A memory management system uses an scheme based on **two-level paging technique**, with **1K page size**. The logical address is up to 4Mbytes and the first level page table consists of 1024 entries.

- Specify the possible logical address or addresses that can correspond to the physical address 2516.

4 MB => 2^22 => Logical address 22 bits
1KB page size => 10 bits
1024 entries => 10 bits. => first level
10 + 2 + 10 => 2 bits => second level

2516 mod 1024 = 468 is the page offset (1D4)
All logical addresses with 468 offset
xx xxxx xxxx xx 01 1101 0100

- Exercise 1: Contiguous Allocation
  - Exercise 1.1: Variable Partitions
  - Exercise 1.2: Base and limit registration
  - Exercise 1.3: Degree of Multiprogramming
- Exercise 2: Sparse Allocation
  - Exercise 2.1: Formatting logical and physical addresses.
  - Exercise 2.2: From logical to physical addresses
  - Exercise 2.3: From logical to physical addresses
  - Exercise 2.4: From physical to logical addresses
  - Exercise 2.5: From physical to logical addresses
- **Exercise 3: Table Sizes and Fragmentation**
  - **Exercise 3.1: Table size**
  - **Exercise 3.2: Internal Fragmentation**

- **A system** uses a 2-level paging memory for logical-physical memory addresses.
- The **logical address spaces are 8Gbytes**, the **page size is 2Kbytes** and the **first level page table consists of 256 entries**. Specify the size of the second level page table assuming that the **page descriptors have 4 bytes**.

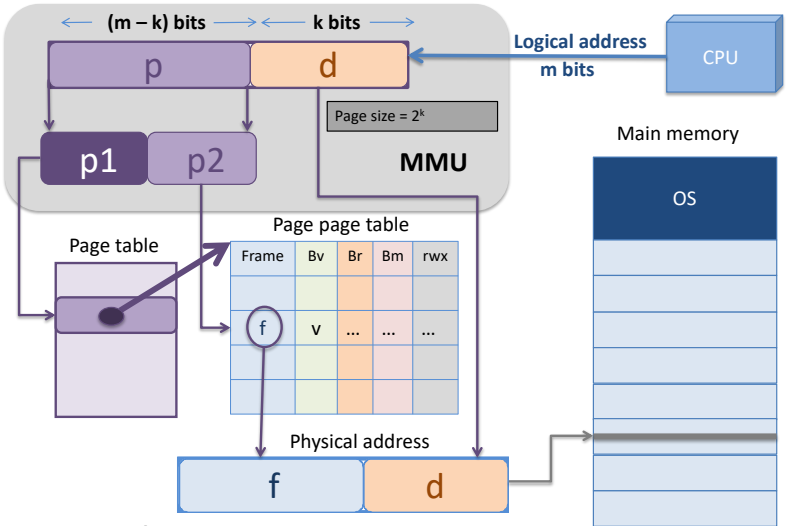
Ejercicio 3.1

- A system uses a 2-level paging memory for logical- physical memory addresses.
- The **logical address spaces are 8Gbytes**, the **page size is 2Kbytes** and the **first level page table consists of 256 entries**. Specify the size of the second level page table assuming that the **page descriptors have 4 bytes**.

Logical address 8GB = 2^{33} => 33 bits
Page size 2KB = 2^{11} => **11 bits for offset**

p1	p2	offset
8 bits	14 bits	11 bits
33 .. 26	25 .. 11	10 0

First level pages 2KB contain 256 entries
It requires **8 bits 2^8 => 256**
Each entry has
Ceiling(2*1024 / 256) => 8 bytes



2^{nd} page bits = 33 – 11 – 8 = 14

Page 2KB contains page descriptors of 4 bytes
Ceiling (2*1024B / 4B) = 512 entries
It requires 2^9 => 512 entries
14 bits for offset in second page of 512 entries means that
With 14 bits, it can address 16K of entries => 16K * 4B => 64 KB => 32 pages of 2KB

Ejercicio 3.2

- A system has a 2 level paging memory management with 16-bit logical addresses, 256Byte page size and 16-entry second level page table.

Calculate:

a) **Maximum size of a process (in bytes)**

b) **1st level page, 2nd level page and offset of the following logical addresses:** 0x9134, 0x5634, 0x4500, 0x0012.

c) **Internal fragmentation of a system of 4 processes with the next sizes:** P0 2688 Bytes, P1 1984 Bytes, P2 1344 Bytes y P3 3264 bytes.

- A system has a 2 level paging memory management with 16-bit logical addresses, 256Byte page size and 16-entry second level page table.

Calculate:

a) Maximum size of a process (in bytes)

Logical address of 16bits => 2^{16} => 64KB => 65536 bytes

Ejercicio 3.2

- A system has a 2 level paging memory management with 16-bit logical addresses, 256Byte page size and 16-entry second level page table.

Calculate:

b) 1st level page, 2nd level page and offset of the following logical addresses: 0x9134, 0x5634, 0x4500, 0x0012.

Logical address

Offset: 256 => 2^8 => 8 bits

Second level with 16 entries => 2^4 => 4 bits

First level bits = 16 – (4 + 8) => 4 bits

	1st level	2nd level	offset
0x9134	0x9	0x1	0x34
0x5634	0x5	0x6	0x34
0x4500	0x4	0x5	0x00
0x0012	0x0	0x0	0x12

- A system has a 2 level paging memory management with 16-bit logical addresses, 256Byte page size and 16-entry second level page table.

Calculate:

c) Internal fragmentation of a system of 4 processes with the next sizes: P0 2688 Bytes, P1 1984 Bytes, P2 1344 Bytes y P3 3264 bytes.

The internal fragmentation of a process corresponds to the not used bytes of the last page.

	address	mod 256	wasted
P0	2688	128	128
P1	1984	192	64
P2	1344	64	192
P3	3264	192	64
			448