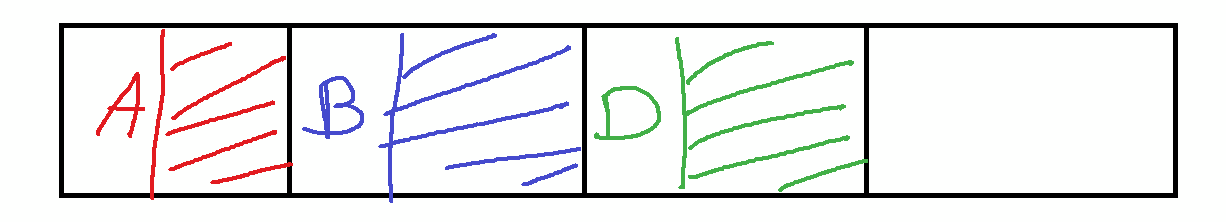
# Exercice 1

* Q1: 4 programmes max (256/64=4)
* Q2: C et E ne rentrent pas dans la mémoire car leur taille est supérieure à 64.

Fragmentation interne : 59 + 39 + 32 = 130

Inconvénients : beaucoup de fragmentation interne, degré de multiprogrammation limité



* Q3: On a pu prendre en charge tout le monde

Un joli schéma svp

* Q4: Non
* Q5:

On ne peut pas car les espaces libres sont plus petits que 70ko, il faut compacter.

* Q6:

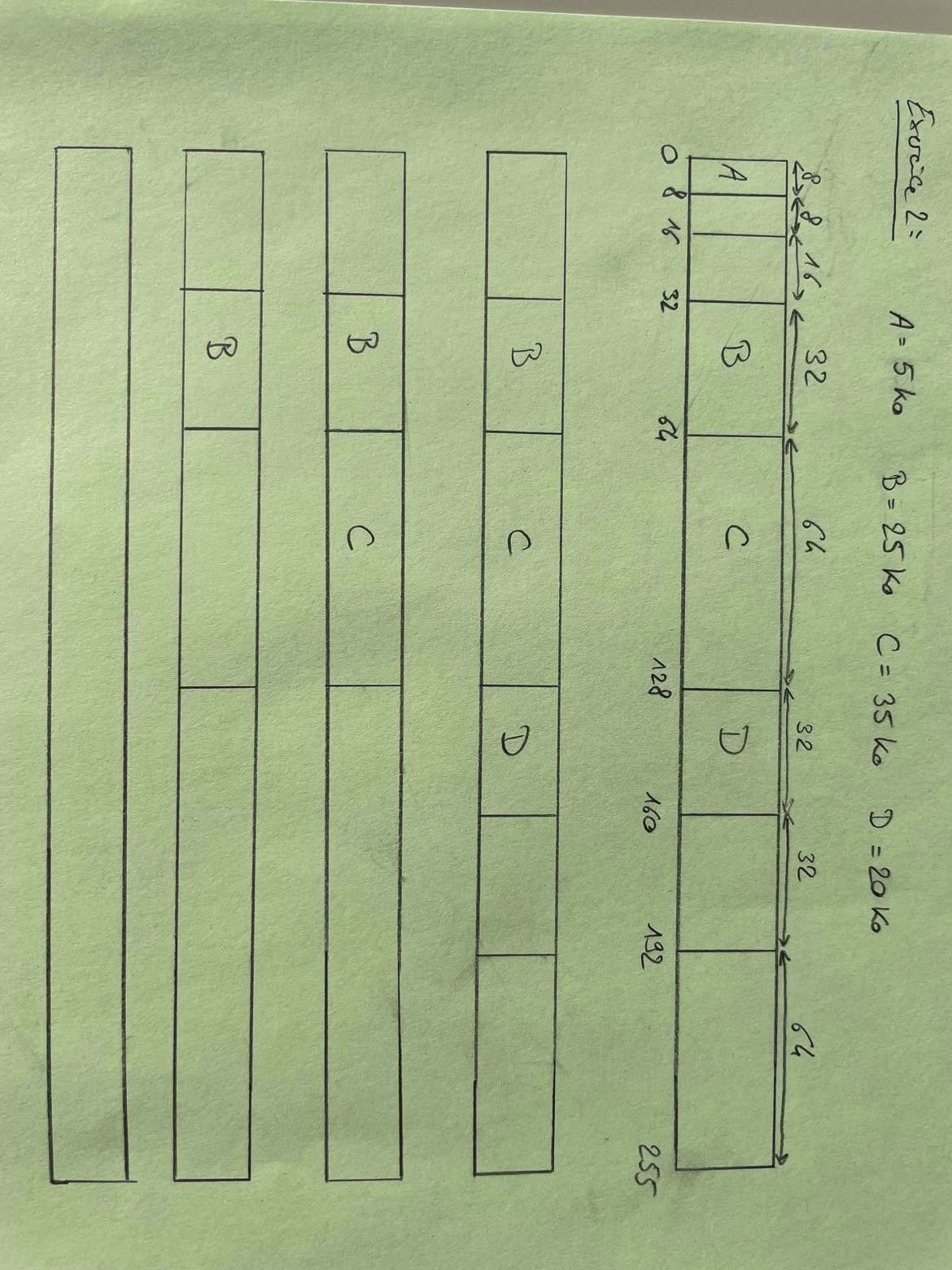
Taille moyenne fragmentation interne avec pagination : ((p-1)+1)/2=p/2

* Q7:

2A 7B 17C 8B 25 E reste 5 cases

2F 16F reste 1 case

# Exercice 2

* Q8
* Q9:

if i > MAX then return -1;

elif Li = ∅ then

ad = trouver\_trou(i+1);

if ad ≠ -1 then

Diviser ce trou en 2 trous de taille 2i+1 à mettre dans Li

else return -1;

return premier trou Li

fi

≠∅

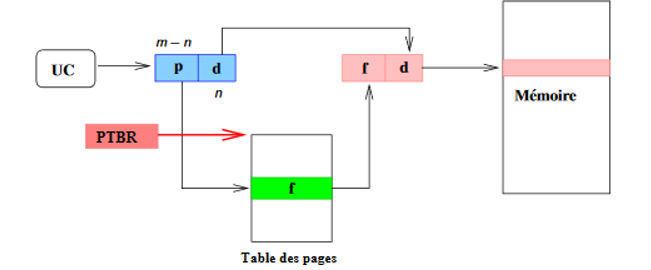
# Exercice 3

* Q10 :

2ko = 2^11, donc d = 11 et m = 32 donc m-d = 21 (taille du n° de page)

P (page) : 21 bits  
d (déplacement) : 11 bits

* Q11 (diapo 20/65 CM 2)



* Q12 :

2^21 entrées

mémoire virtuelle 4Go (2^2 \* 2^30)

32 bits pour l’entrée, donc 2^21\*2^11=2^32

* Q13 :

A : 0-2

B: 3-8

v : validé/vérifié, i : marche pas

Table des pages de A

0 : v - 0

1 : v - 1

2 : v - 2

3 : i - ?

Table des pages de B

0 : v - 3

1 : v - 4

2 : v - 5

3 : v - 6

4: v - 7

5 : v - 8

6: i - ?

7 : i - ?

* Q14 :

n°case : PID n°page

0 : A 0

1 : A 1

2 : A 2

3 : B 0

4 : B 1

5 : B 2

6 : B 3

7 : B 4

8 : B 5

* Q15 :

Prend moins de place en mémoire que la table des pages précédente (car il y a autant d'entrées que de cases) mais la conversion adresse virtuelle - adresse physique est plus lente (Recherche en O(n))

+ Mémoire cache pour stocker la table des pages inversée

+ Utilisation de hachages

2^n

2^(n/p \*e)

1/2p

-2^n\*e\*p^-2

* Q16

# Exercice 4

Fonction initiale : f(p) = 2^n/p \* e + p/2

f’(p)=½-1/p²\*2^n\*e

f’(p)=0

1/p²=1/(2\*2^n\*e)

sqrt(2\*2^n\*e)=p=sqrt(2^36)=2^18

# Exercice 5

* Q17:

p=4Ko=2^12 donc champ deplacement sur 12 bits

Il reste 20 bits pour le numéro de page

Pour 4 octets par entrée, la table des pages fait 2^20\*2^2=2^22

* Q18:

Champ p1 : 10bits

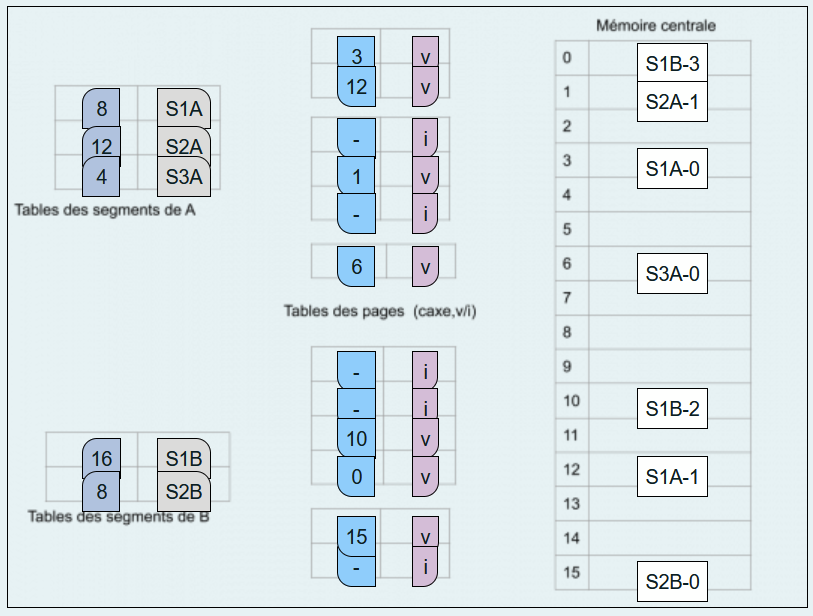
Champ p2 : 10bits

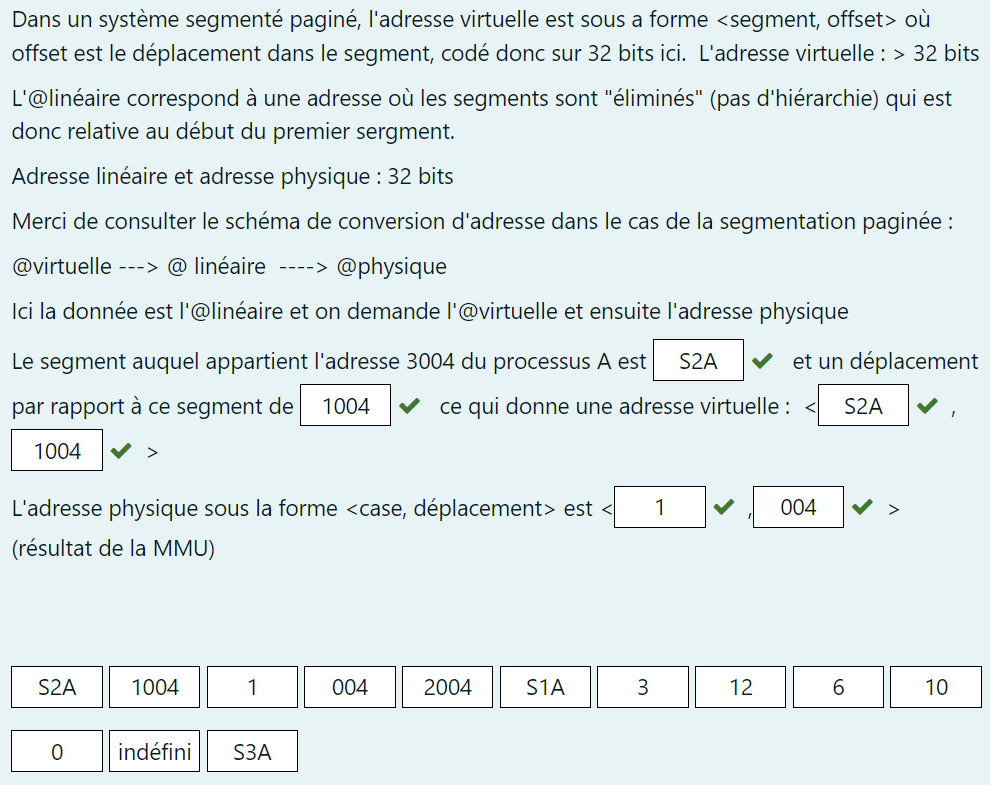
Taille champ déplacement : 12 bits

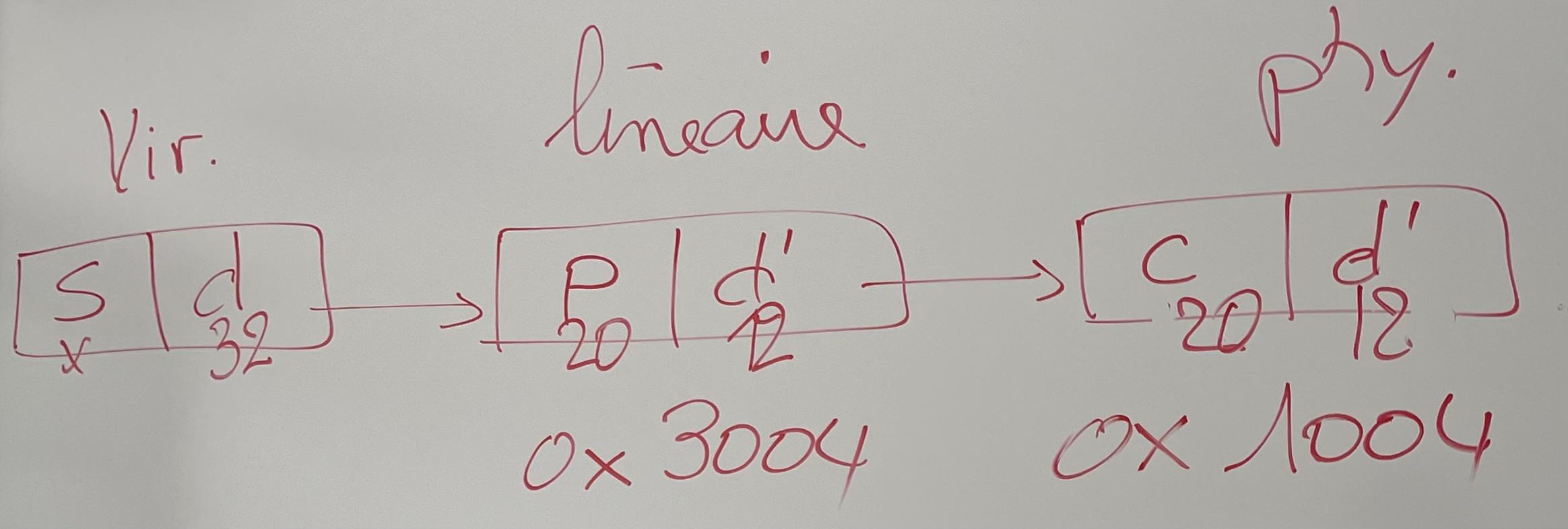
Taille page 1 : 2^10\*2^2=2^12

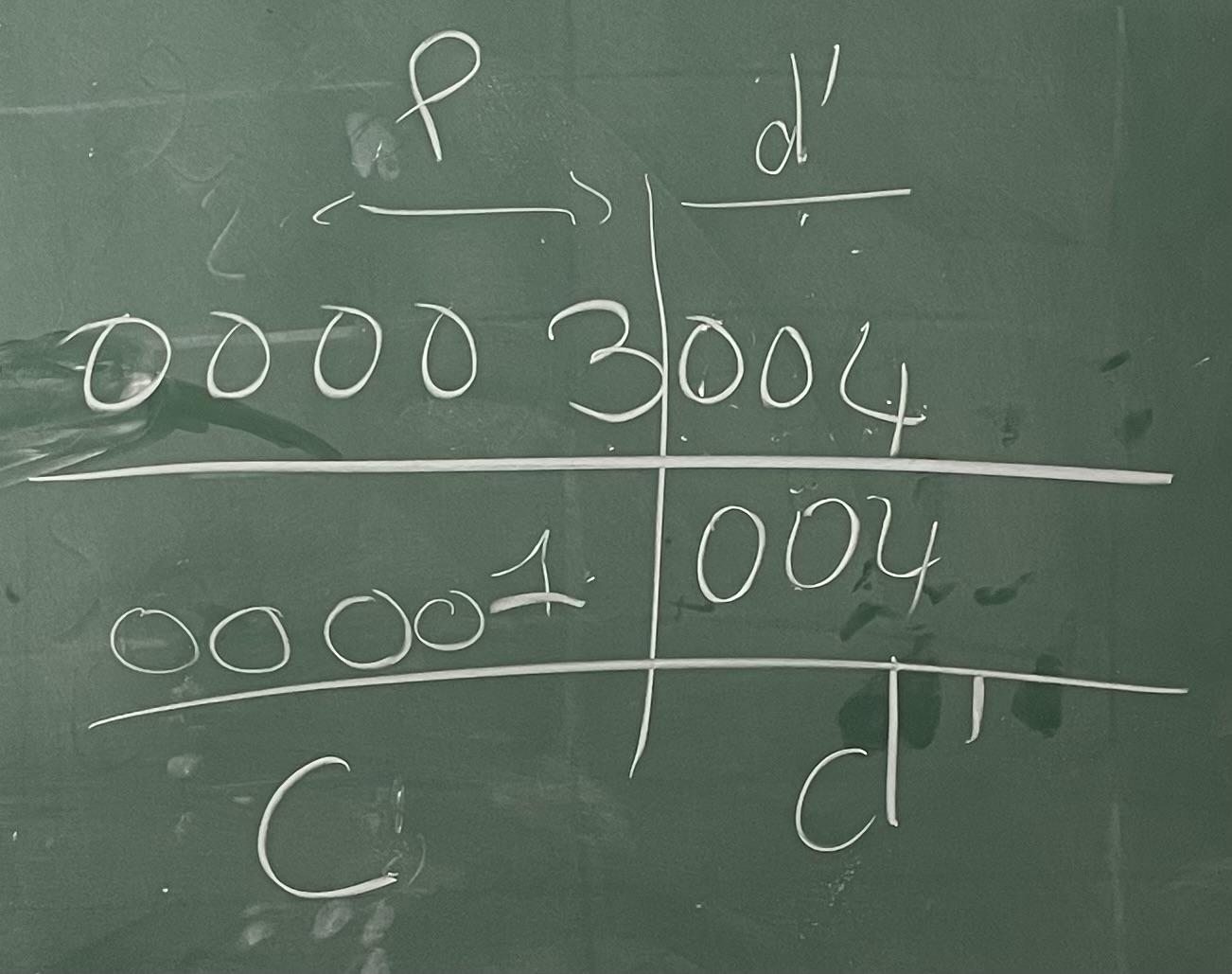
**Taille totale :(2^10 + 2^10 × 2^10) × 4 octets**

# Exercice 6









0000 3004

**Adresse linéaire= 0x3004**

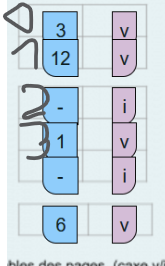
p= 0000 3 et d’=004

d’ a une taille de 3 car c’est 3 fois 4 bits = 12 bits nécessaires au déplacement

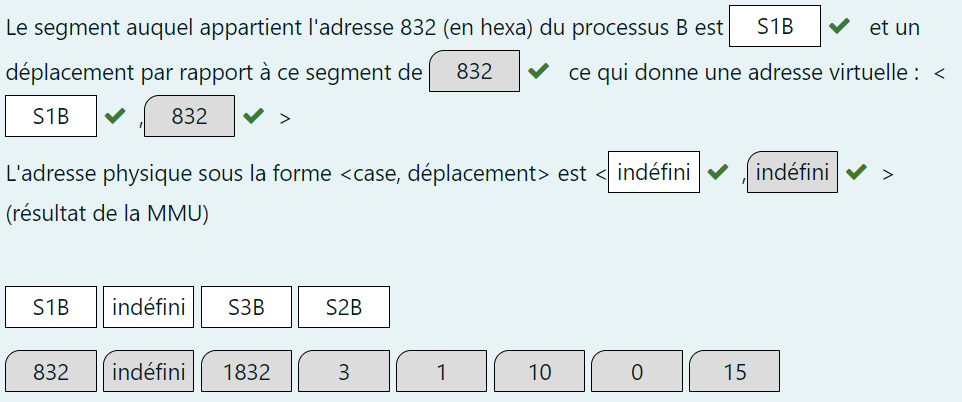
Adresse physique

c= 00001 d’=004 pour trouver c il faut prendre dans la table des pages avec le 3

Donc **@ physique = 0x1004**



**Adresse Virtuel = <S2A,0x1004>**

****

**Q22 :**

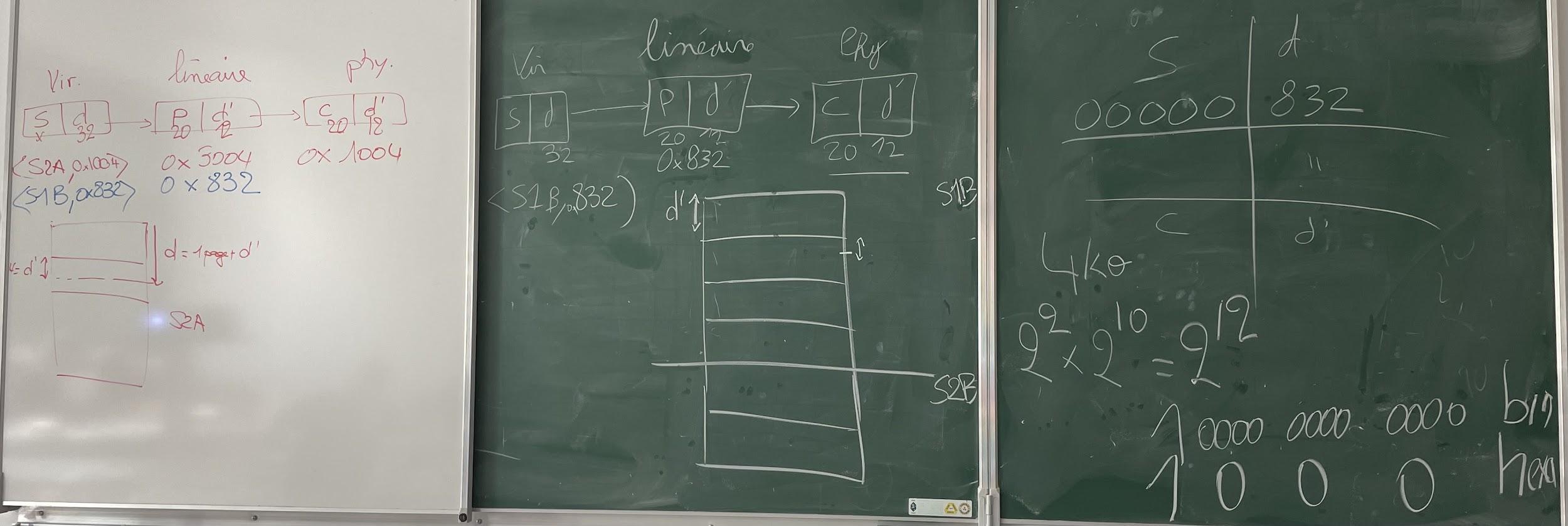
4 Ko = 4Kio = 2^2 \* 2^10 = 2^12

Donc 12 est le déplacement maximal au sein d’une page.

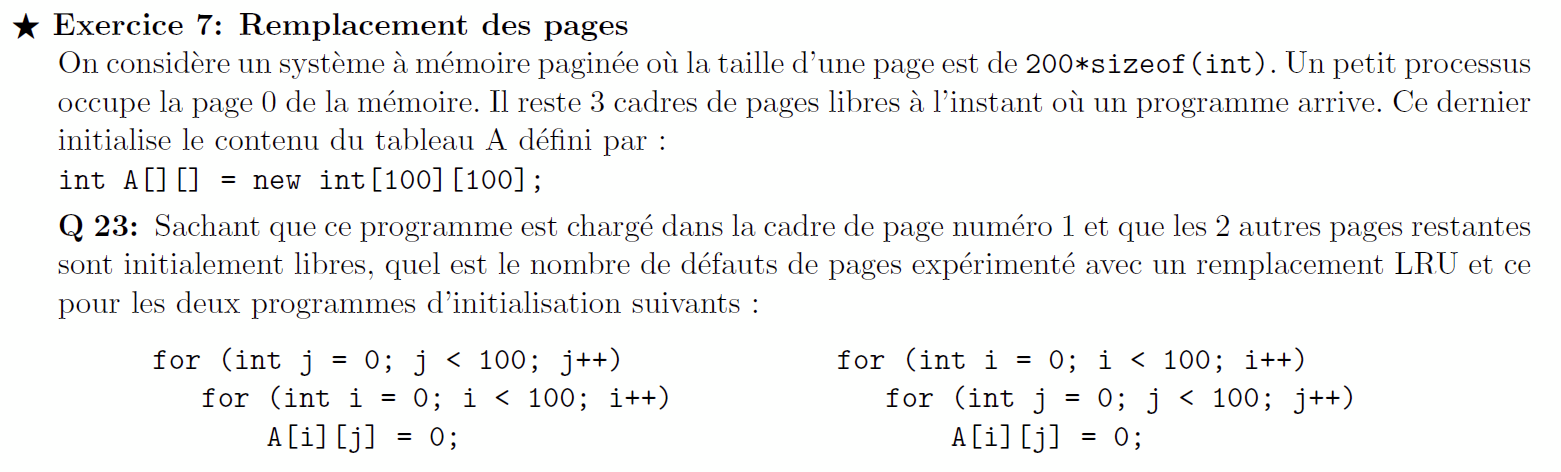
C’est pour cela que l’on coupe 32 en 20 : 12

2^12 = 1 0000 0000 0000 binaire = 1 0 0 0 hexadécimal

Donc une page = 1000



# Exercice 7

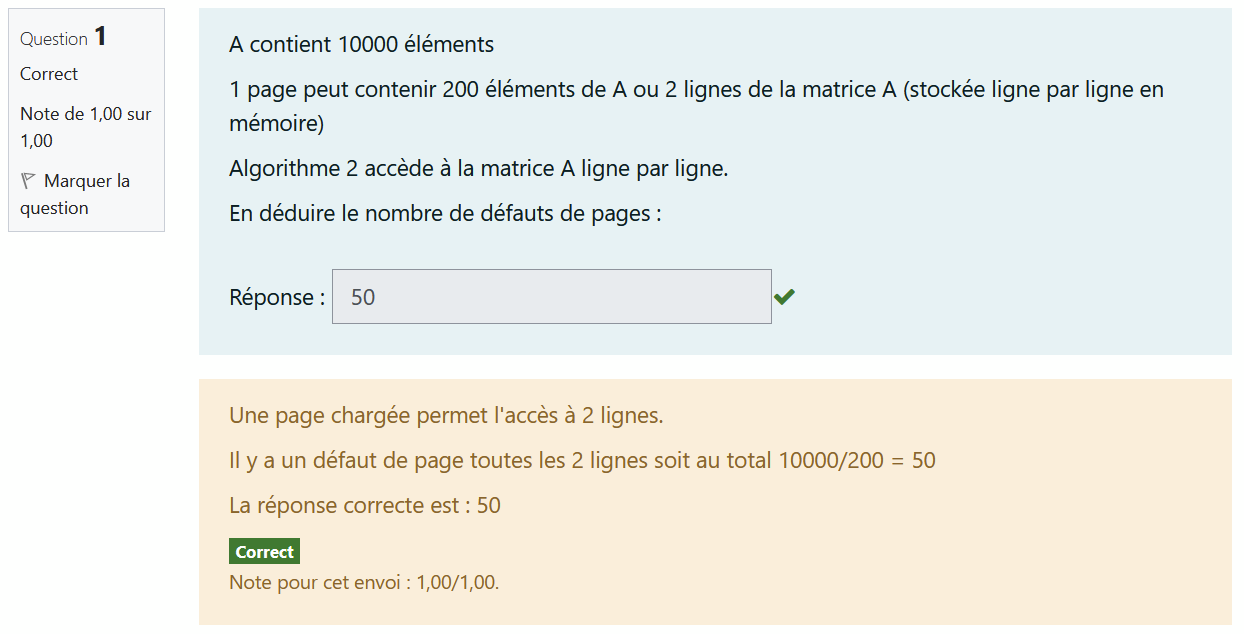


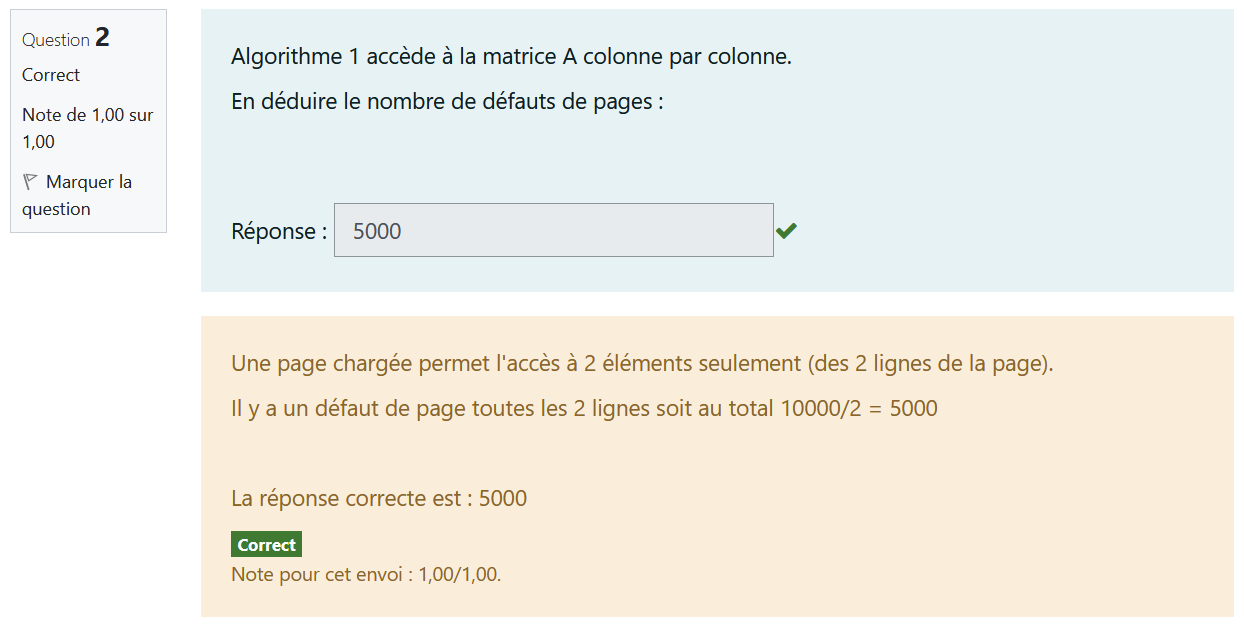
Il y a 4 pages,   
Page 0 : SE (système d’exploitation)

Page 1 : Code

Il reste 2 pages de taille 200 soit deux lignes de 100.

Si on accède aux données ligne par lignes,





# QCM - M1

