Pagination

Attention : Dans ce TD, toutes les *adresses* et *portions d'adresses* sont données en hexadécimal.

Exercice 1. Questions de cours Les questions de cours sont à destinées à vous permettre de vérifier votre compréhension du cours. Elles sont à travailler à l'avance et ne seront pas traitées en TD ou TP.

1. En quoi consiste la pagination de la mémoire ?
2. Quel est le principal problème de la pagination ?
3. Quel est l'interet de la pagination hierarchique ?
4. Donnez deux avantages de la mémoire virtuelle.

Exercice 2. Fondements On suppose un espace d'adresses logiques de 8 pages de 1024 octets chacune permettant d'accéder à une mémoire physique de 32 cadres de pages.

1. Expliquez pourquoi les tailles de pages sont toujours une puissance de 2.
   Correction: *Si ça n'était pas le cas, il y aurait de bouts de mémoire qui ne seraient pas référençables sans "sortir" de la page, puisque les adresses sont de la forme* 2^{offset}*, donc de la fragmentation inutile.*
2. Combien de bits comporte l'adresse logique ? L'adresse physique ?
   Correction: $1024 = 2^{10}$ *donc* 10*bits pour l'offset dans la page,* $8 = 2^3$ *donc* 3*bits pour le numéro de page et* $32 = 2^5$ *donc* 5*bits pour les cadres de pages. Adresse logique* $= 13$*bits, adresse physique* $= 15$*bits.*
3. On suppose maintenant que le système dispose de 2048ko de mémoire logique organisé avec des pages de 8ko. Décrivez le système d'adressage logique.
   Correction: $2048\text{Ko}/8\text{Ko} = 2^{11+10}/2^{3+10} = 2^8$ *pages. Les adresses logiques sont donc sur* $8 + 13$*bits (*8 *pour le numéro de pages et* 13 *pour le décalage)*
4. Quelle est la taille maximum de la table des pages ?
   Correction: *Pour adresser* 2^8 *pages, il faut 1 octet. On a donc une table des pages qui peut faire* 2^8 *octets* $= 256$*o (par processus). En pratique, il faut aussi probablement* 1*bit de validité, ce qui fait* 9*bit par ligne, que l'on arrondi à* 2 *octets et donc une taille de table de* 512 *octet.*
5. On suppose que, dans le système de la question précédente, on a trois processus qui s'exécutent sur le système : P1 nécessistant 200ko, P2 de 545ko et P3 de 337ko. Quelle est la quantité de mémoire réellement utilisée par l'exécution de ces trois processus ? Quel est le taux de fragmentation ?
   Correction: *La table des pages doit être stockée dans un cadre séparé pour chaque processus, il nous faudra donc déjà 3 cadre pour ces tables. P1 nécessite* 25 *cadres de pages, P2 nécessite 69 cadres et P3 nécessite 43 pages. Donc le total est de* $3 + 25 + 69 + 43 = 140$ *pages de 8ko, soit* 1120*ko (un peu plus de 1Mo).*

   Fragmentation = 0Ko perdus pour P1 + 7Ko perdus pour P2 + 7Ko perdus pour P3 soit 14Ko. On ne compte pas ici la mémoire perdue dans les cadre contenant les tables des pages, dans de vrais systèmes les tailles sont calculées de maière à ce que une table occupe exactement un cadre afin de ne pas gaspiller de mémoire.

Exercice 3. Pagination à 1 niveau On considère un système 32bits utilisant la technique de pagination avec des cadres de pages de 64ko. Chaque processus peut utiliser au plus 4Go de mémoire virtuelle et le système peut accueillir jusqu'à 2048 processus.

1. Quelle est la taille maximale de la mémoire physique ?
Correction: *Système 32bits signifie que l'adressage se fait au plus sur 32bits : chaque processus peut adresser au plus $2^{32} = 4Go$ de données et la RAM ne peut pas excéder 4Go.*

2. Quelle est la taille de l'adresse logique ?
Correction: *Chaque processus peut adresser 4Go de données donc l'adresse logique est sur 32bits.*

3. Quelle est la quantité maximale de mémoire virtuelle utilisée par le système ?
Correction: *On peut avoir 2^{11} processus utilisant chacun 2^{32} octets, soit en tout $2^{43} = 8To$.*

4. On suppose que le système dispose de 1Go de mémoire physique. Quelle est la taille de l'adresse physique ?
Correction: *L'adresse physique est sur 30bits ($2^{30} = 1Go$).*

5. Sur combien de bits est codé le décalage dans l'adresse logique ? Dans l'adresse physique ?
Correction: *Le décalage est sur 16bits car $2^{16} = 64ko$. C'est évidemment le même pour l'adresse logique et l'adresse physique !*

6. Sur combien de bits est codé le numéro de page dans l'adresse logique ?
Correction: *16bits (le reste sur 32bits d'adresse logique).*

7. Sur combien de bits est codé le numéro de cadre dans l'adresse physique ?
Correction: *14bits (le reste sur les 30bits d'adresse physique).*

8. Quelle est la taille maximale de la table des pages d'un processus ?
Correction: *2^{16} lignes de $14 + 1$ bits (numéro de cadre + bit de validité). On arrondit à une taille qui soit un multiple d'octets donc 16bits par lignes. Ce qui donne $2^{16} \times 16 \div 8 = 2^{17}o$ (exactement 128ko).*

9. En considérant les huit premières entrées de la table de page donnée ci-après, calculez les adresses physiques correspondant aux adresses logiques `00030B72` et `00060D81` ?

Page	Cadre	Validité
0	000	0
1	010	0
2	000	1
3	0B3	1
4	2A0	0
5	09B	0
6	0F0	1
7	2DD	1

Correction: *Il suffit de prendre les 16 premiers bits de l'adresse logique, de trouver la page correspondante dans la table, de vérifier qu'elle est valide et de remplacer le numéro de page par le numéro de cadre, ce qui donne :* `0B30B72` *et* `0F00D81`.

10. À l'inverse, pouvez-vous indiquer si la donnée située à l'adresse physique `2A0DE37` appartient au processus ? Expliquez.
Correction: *On prend les 14 premiers bits de l'adresse (*`2A0`*) et on regarde si cela correspond à un numéro de cadre utilisé par le processus. En l'occurrence, la page 4 mentionne ce cadre mais il est marqué invalide. Sous réserve que ce cadre n'apparaisse pas dans la suite de la table, cette adresse physique n'est donc* **pas** *utilisée par le processus.*

11. Est-ce que `37` est une adresse physique valide ? Et `90003145` ? Expliquez.
Correction: *37 est une adresse valide : il faut compléter l'adresse par des 0 :* `00000037` *(numéro de cadre,décalage).*
`90003145` *n'est pas une adresse physique valide car les 4 permiers chiffres (*`9000`*) ne peuvent pas se coder sur 14bits. En effet, avec 4bits par chiffre hexadécimal, sur 14bits, on peut au plus aller jusqu'à* `3FFF` *(3×4bits pour chacun des 3 derniers chiffres, et 2 pour le premier chiffre qui ne peut dont pas être plus grand que 3).*

Exercice 4. Pagination à 1 niveau On considère un système disposant de 4Mo de mémoire physique utilisant la technique de la pagination à 1 niveau avec des cadres de page de 1ko. Chaque processus dispose de 16Mo de mémoire virtuelle.

1. Quelle est la taille des adresses physiques ?
   Correction: *22bits car* $4Mo = 2^{22}o$.
2. Sur combien de bits est codé le numéro de page dans l'adresse physique ?
   Correction: *12bits car le décalage est codé sur 10bits (*$1ko = 2^{10}$*).*
3. Quelle est la taille des adresses logiques ?
   Correction: *24bits car* $16Mo = 2^{24}$.
4. Combien de lignes y a-t-il dans la table des pages d'un processus ?
   Correction: $2^{14} = 16384$ *car le numéro de page est codé sur 14bits.*

   Le tableau suivant donne un extrait de la table des pages d'un processus :

Page	Cadre	Validité
AC2	000	0
AC3	35E	1
AC4	12A	1
AC5	000	0
AC6	000	0
AC7	000	0
AC8	000	0
AC9	1A7	0

5. Calculez les adresses physiques correspondant aux adresses logiques 2B0E4C et 2B27F0.
   Correction: 2B0E4C *se découpe en un numéro de page* AC3 *et un décalage* 24C. *Cela permet de trouver le numéro de cadre* 35E, *de vérifier qu'il est valide et de le recombiner pour obtenir l'adresse physique* D7A4C.
   Pour 2B27F0, *le découpage donne* 3F0 *comme page qui d'après la table est invalide, on a donc un page fault.*
6. Calculez l'adresse logique que le processus doit utiliser afin de réaliser un accès mémoire à l'adresse physique 4A8F1.
   Correction: 4A8F1 *se découpe en un numéro de cadre* 12A *et un décalage* 0F1. *La table des pages nous indique que ce cadre contiens actuellement la page* AC4, *en la recombinant au décalage on obtient l'adresse logique* 2B10F1.
7. Que doit faire l'OS pour que le processus puisse réaliser un accès mémoire à l'adresse physique D12A ?
   Correction: *Cette adresse est située dans le cadre* 034, *il faut donc ajouter une entrée valide dans la table des pages afin qu'une des pages du processus soit stockée dans ce cadre.*

Exercice 5. Pagination à 2 niveaux On se place dans un système de mémoire de 4Go de mémoire géré de manière paginée avec des cadres de page de 4Ko. Chaque processus peut utiliser jusqu'à 64Mo de mémoire. Le système d'exploitation autorise jusqu'à 1024 processus.

1. Quelle est la taille (en bits) de l'adresse logique
   Correction: $64\text{Mo} = 2^{2}6o$ *donc l'adresse logique est sur 26 bits.*
2. Quelle est la taille (en bits) de l'adresse physique
   Correction: $4\text{Go} = 2^{32}o$ *donc l'adresse physique est sur 32 bits.*
3. Combien y a-t-il de cadres de page dans la RAM ?
   Correction: $4\text{Ko} = 2^{12}$. *Il y a donc* $2^{(32-12)} = 2^{20}$ *cadres de pages.*
4. Combien chaque processus peut-il contenir de pages ?
   Correction: *Chaque processus peut contenir jusqu'à* $2^{(26-12)} = 2^{14}$ *pages.*
5. On suppose que la pagination se fait sur un seul niveau. Quelle quantité de mémoire est consommée par les tables des pages ?
   Correction: *Chaque processus a une table de* 2^{14} *lignes de 20 bits chacune. On arrondi la taille des lignes à 4 octets soit environ 64ko. Il y a jusqu'à 1024 processus. Donc en tout, on peut consommer jusqu'à 64Mo de RAM pour la pagination.*
6. On suppose que la pagination se fait sur deux niveaux avec un répertoire sur 6 bits. Quelle quantité de mémoire est consommée par un processus qui utilise les plages d'adresses logiques suivantes ?

de 0 00 1B 07 à 0 00 2C 08
de 0 03 CA 00 à 0 0E 00 00
de 0 0D AB 10 à 0 0F B1 CC
de 0 2B 10 21 à 0 2B 3A 72

Correction: *Pour commencer, il faut remarquer que dans chaque adresse, les 6 premier bits (les 2 premiers « chiffres ») désignent le numéro de la table des pages concernée dans le répertoire, les 8 suivants (2 « chiffres ») la page et les 12 derniers bits (3 « chiffres ») constituent le décalage. Nous avons donc un processus qui utilise uniquement 2 tables des pages dans son répertoire. Ainsi, la quantité de mémoire utilisée est de 2^6 lignes de 4 octets chacune pour le répertoire (adresse de la table des pages), soit 256o, plus 2 tables de 2^8 lignes de 4 octets chacune, soit 1ko, donc un total de moins de $1,25$ko. Cela représente quand même 20 fois moins de mémoire que ce qu'on pourrait craindre...*

Exercice 6. Pagination à 2 niveaux On considère un système de gestion de mémoire paginée à deux niveaux tel que :

- Les adresses virtuelles et physiques sont toutes deux codées sur 32bits ;
- Les 10 premiers bits de l'adresse virtuelle forment le premier index ($n_1 = 10$), les 10 bits suivant forment le deuxième index ($n_2 = 10$) et les 12 bits restant le déplacement ($m = 12$) ;
- Les tables de pages et le répertoire de chaque processus sont stockées dans la RAM et chaque entrée correspond à une adresse différente dans la RAM.

1. Sur combien d'octets seront codées les données du répertoire ? Quelle est la taille maximum du répertoire ?

Correction: *Le répertoire contient $2^{n_1} = 2^{10}$ lignes de 32bits (adresses physiques) chacune pour adresser les tables des pages, ce qui nécessite $\frac{32}{8} = 4$ octets par lignes. Le répertoire occupe donc au plus 4ko.*

2. Sur combien d'octets seront codées les données d'une table de page ? Quelle est la taille maximum d'une table de pages ?

Correction: *Chaque table des pages contient $2^{n_2} = 2^{10}$ lignes de 21bits chacune ($32 - 12 = 20$ bits de numéro de cadre + 1bit de validité), que l'on arrondis à 4 octets par lignes. Une table de page occupe donc au plus 4Ko.*

3. Quelle est la quantité maximale de mémoire nécessaire pour stocker toutes ces tables ?

Correction: *Le répertoire et l'ensemble des pages, pour adresser 4Go de données (le maximum adressable ici), occupe donc 4ko (répertoire) + $2^{10} \times 4$Ko (il y a $2^{n_1} = 2^{10}$ table de pages de 4Ko chacune), soit un peu plus de 4Mo.*

4. On considère un processus avec le répertoire suivant :

Index	Table
0	0A7C 0002
1	0A7C 006E
2	0A7C 0072
3	0A7C 005A

Et on souhaite résoudre l'adresse logique : 00C02DF6. Quelle table est concernée ?

Correction: *C'est difficile à voir sans repasser au binaire (0000 0000 1100 0000 0010 1101 1111 0110), mais les $n_1 = 10$ premiers bits de l'adresse donnent la valeur 11 (en base 2) $= 3$ (en base 16) : il s'agit de la table des pages située à l'adresse 0A7C 005A qu'il faut aller chercher (entrée no 3 du répertoire).*

5. On suppose que la table des pages concernée (par le résultat de la question précédente) commence par les entrées suivantes :

Page	Cadre	Validité
0	17AC0	0
1	02BCF	1
2	07F93	1
3	15BB0	1

Donnez l'adresse physique correspondant à l'adresse logique 00C0 2DF6.

Correction: *Les 12 derniers bits de l'adresse (le décalage) valent DF6. Il suffit de les mettre derrière le numéro de cadre donné correspondant au numéro de page indiqué dans les bits 10 à 19. Reprenons notre adresse binaire :* 0000 0000 1100 0000 0010 1101 1111 0110. *Les 10*

premiers bits nous ont permis de trouver la table de page no 3 du répertoire. Les 10 bits suivants (00 0000 0010) nous amènent sur la page no 2 de cette table, c'est-à-dire le cadre 07F93 qui est marqué "valide". L'adresse physique est donc : 07F9 3DF6.

Exercice 7. Segmentation paginée On considère un système muni de 64 ko de mémoire physique gérée de manière segmentée et paginée. Chaque processus peut utiliser 16 segments de 1 ko et le système supporte jusqu'à 256 processus. Les cadres de page font 512 o.

1. Quelle est la taille de l'adresse physique ?

Correction: *16 bits (2^{16} = 64 ko de RAM).*

2. Quelle est la taille de l'adresse logique ?

Correction: *14 bits (2^{14} = 16 ko pour 16 segments de 1 Kio).*

3. Quelle est la taille maximale de la mémoire virtuelle ?

Correction: *256 processus de 16 ko = jusqu'à 4 Mo de mémoire virtuelle.*

4. Rappeler ce qu'est un segment global.

Correction: *Il s'agit d'un segment partagé par l'ensemble des processus.*

5. Dans cette question, on suppose que la moitié des segments sont globaux. Dans ce contexte, quelle est la taille de la mémoire virtuelle ?

Correction: *On a 256×8 ko $+ 1 \times 8$ ko $\simeq$ 2 Mo de mémoire virtuelle.*

6. Combien de pages un processus peut-il utiliser au maximum ?

Correction: *Chaque processus peut avoir au plus 32 pages de 512 o (32×512 = 16 ko max par processus).*

7. On rappelle que l'adresse linéaire doit permettre d'adresser toute la mémoire d'un processus, mais qu'elle n'a pas besoin d'adresser l'ensemble de la mémoire virtuelle. Quelle est la taille et la composition de l'adresse linéaire ?

Correction: *Il faut un numéro de page et un décalage (couvrant un cadre de page, donc $\log_2(512)$ = 9 bits). Le tout est sur 14 bits pour couvrir les 16 Kio par processus. Donc 5 bits pour le numéro de page et 9 bits pour le décalage.*

8. Quelle est la taille (en nombre de bits) de chaque ligne de la table des descripteurs lorsque tous les segments sont locaux ?

Correction: *Chaque ligne contient les informations suivantes : adresse de base, limite. La limite est forcément inférieure à 1024 (taille max d'un segment), donc sur 10 bits. L'adresse de base peut couvrir l'ensemble de la mémoire d'un processus, donc sur 14 bits (pas besoin de 22 bits qui couvrirait la mémoire virtuelle complète). En tout, nous avons donc 14+10 = 24 bits. (on arondi le plus souvent à 32bits)*

9. Même question si la moitié des segments sont globaux (ce qui est la configuration habituelle sur un OS moderne).

Correction: *Il y a deux approches. Le cas le plus fréquent, c'est que les 2^{n-1} premiers segments soient locaux et que les 2^{n-1} suivants soient globaux (n désignant ici le nombre de bit du numéro de segment). Dans ce cas, il n'y a pas besoin d'indiquer quel segment est local ou global et cela ne change rien à la taille des lignes.*

Il est aussi possible de choisir d'utiliser plutôt un bit supplémentaire pour indiquer si le segment est global ou local auquel cas il faut un bit de plus par ligne.

On reste donc sur 14 bits d'adresse de base et 24 bits en tout par ligne de table des descripteurs ou (25bits dans le deuxième cas).

À un moment de l'exécution, plusieurs processus P_1, P_2, ..., P_N sont en exécution dans le système. L'état du système est partiellement décrit ci-après :

Table des segments de P_1 :

Segment	Limit	Base
00	085	0000
01	3B6	3000
02	341	2000
03	225	0086
04	05F	1B80

Table des segments de P_2 :

Segment	Limit	Base
00	0A5	1001
01	107	0C00
02	3B6	3000
03	0A3	01CF

Table des pages de P_1 :

Page	Cadre	Valide
00	36	0
01	7A	0
02	32	1
0E	00	1
0F	2A	0
10	2B	0
11	14	1
18	6C	1
19	55	1
1A	31	1
1B	30	0

Table des pages de P_2 :

Page	Cadre	Valide
00	24	0
01	32	0
08	2A	1
09	76	1
0A	54	0
18	6C	1
19	55	1
1A	31	1

10. Quelle est l'adresse physique qui correspondant à l'adresse logique 0B50 pour le processus P_1 ?
Correction: *Attention : sur 14 bits, si on ne passe pas par l'écriture binaire, on va se tromper car il faut 4 bits pour le numéro de segment, c'est-à-dire les 2 derniers bits du premier chiffre (qui est forcément entre 0 et 3) et les 2 premiers du suivant ! Il ne faut pas prendre le premier chiffre comme le numéro de segment !*
Sur cet exemple, l'adresse binaire est (00)00 1011 0101 0000 qui se décompose en décalage (10 bits) + numéro de segment (4 bits) + 2 bits nuls. En l'occurrence : segment = 0010 = 2 et décalage = (00)11 0101 0000 = 0x350.
On est donc sur le segment 2, qui a une limite à 0x341<0x350 donc il y a une erreur de segmentation.
La difficulté ici est dans la décomposition binaire. Il faut s'assurer que les étudiants ont bien compris. Ceux qui prennent le premier chiffre comme le numéro de segment auront aussi une erreur de segmentation (car 0x085<0x350). Il faut bien leur demander de décomposer le calcul, pas juste la réponse finale.

11. Quelle est l'adresse physique correspondant à l'adresse logique 0B50 pour le processus P_2 ?
Correction: *Même calcul que précédemment, mais cette fois sur le processus 2. On prend donc le segment 2, dont la limite est 0x3B6 dont il n'y a pas d'erreur de segmentation. On fait ensuite Base + décalage = 0x3000 + 0x350 = 0x3350 (soit 0011 0011 0101 0000 en binaire) pour l'adresse linéaire.*
Ensuite, partons de notre adresse linéaire. Elle est composée du décalage (les 9 derniers bits, soit 1 0101 0000) et du numéro de page (les 5 premiers bits, soit 1 1001 = 0x19). La page concernée est donc la page 19.
D'après la table des pages, la page 19 est valide et correspond au cadre 55, soit en binaire : (0)101 0101, seuls les 7 derniers bits nous intéressent car nous allons les concaténer avec les 9 bits du décalage pour obtenir l'adresse physique.
L'adresse physique est donc 1010 1011 0101 0000 = AB50 en hexadécimal.

12. Quelle est l'adresse physique correspondant à l'adresse logique 0C0D pour le processus P_1 ?
Correction: *Il faut refaire le même travail. L'adresse binaire est (00)00 1100 0000 1101 qui se décompose en numéro de segment (0011 = 3) et décalage (0000 0000 1101 = 0x00D). Il n'y a pas d'erreur de segmentation (sauf si les étudiants ont pris 0 comme numéro de segment, mais bon, là ils devraient avoir compris).*
On calcule d'adresse linéaire : 0x86+0xD = 0x93 soit 0000 0000 1001 0011 en binaire. On y trouve le décalage (0 1001 0011) et le numéro de page (0). La page n'est pas valide d'après la table des pages : il y a donc un défaut de page. On ne peut pas donner l'adresse physique ! L'OS va appeler une routine de remplacement de page pour attribuer un cadre à la page 0 manquante.

Exercice 8. Segmentation paginée On considère un système de gestion de mémoire de 64Mo de mémoire physique, gérée de manière segmentée et paginée avec deux niveaux de pagination. Un processus peut avoir au plus 256 segments. Chaque segment peut adresser au plus 64ko de mémoire. Enfin, la taille des cadres de page est fixée à 4ko et le répertoire contient 4 lignes.

1. Quelle est la taille et la composition de l'adresse logique ? Indiquez sa composition.
Correction: *Pour adresser 256 segments par processus, il faut 8bits. Donc le sélecteur dans l'adresse logique est sur 8bits. Si chaque segment peut adresser au plus 64ko de données, le décalage est sur 16bits. L'adresse logique est donc sur 24bits. (remarque : Chaque processus adresse donc au plus 16Mo)*

2. Quelle est la taille et la composition de l'adresse linéaire ?
Correction: *Chaque processus adresse au plus 16Mo, l'adresse linéaire doit donc faire 24bits. Les pages font 4ko, on a donc un décalage sur 12bits. Le répertoire contient 4 ligne, il lui faut donc 2 bits pour les indexer ce qui laisse 10 bits pour le numéro de page.*

3. Quelle est la taille et la composition de l'adresse physique ?
Correction: *Si on a 64Mo de RAM, l'adresse physique est forcément sur 26bits ($64M = 2^{26}$). Si on veut décomposer : avec des cadres de pages de 4ko, il faut 12bits pour le décallage. Comme on a 64Mo de mémoire totale, on a $64/4 = 16$ko cadres de pages possibles, ce qui nécessite 14bits.*
Soit un processus muni de la table des segments et du répertoire suivant :

Segments :

Segment	Base	Limite
00	BE 0A 75	05 00
01	BE 23 D1	00 BF
02	BE 00 DA	03 61
03	BE 1A 26	05 D2
04	BE 0F F0	00 CF

Répertoire :

Répertoire	Table	Valide
0	0	1
1	3	0
2	1	1
3	2	0

et de deux tables de pages (on ne fait figurer ici que quelques pages, les autres ont leur bit de validité à 0) :

Table 0 :

Page	Cadre	Valide
2A0	23 40	1
2A1	05 BB	1
2A2	00 00	0
2A3	14 E0	1

Table 1 :

Page	Cadre	Valide
3E0	00 00	0
3E1	27 FD	1
3E2	00 00	0
3E3	3A F6	1

4. Quelle est l'adresse physique correspondant à l'adresse logique `03 00 F0` ? Indiquez clairement les valeurs calculées pendant le processus de traduction.
Correction: *Adresse logique :* `03 00 F0`

Table des descripteurs : 03 est associé au répertoire BE1A26 et à la limite 05D2. 00F0 est bien inférieur à 05D2 donc l'adresse ne provoque pas d'erreur : on additionne base + décalage pour obtenir l'adresse linéaire.

Adresse linéaire : BE 1B 16, soit en binaire 1011 1110 0001 1011 0001 0110

Puisque le répertoire des tables de pages contient 4 éléments, les deux premiers bits sont ceux du répertoire. Ici, il s'agit du répertoire 2, qui correspond à la table des pages 1.

Les 10 bits suivants correspondent nécessairement au numéro de page dans la table des pages, puisque le décallage doit être sur 12 bits (pour les cadres de page de 4 ko) et qu'il reste 22 bits. Notons au passage (en prévision de la question suivante) qu'il y a au plus $2^{10} = 1024$ pages dans chaque table.

Les 10 bits en questions sont 11 1110 0001, soit 3E1 en hexadécimale. D'après la table de pages 1, la page 3E1 est active (pas de déroutement) et correspond au cadre 07 FD.

Le reste (B16) est le décallage. L'adresse physique est donc : 2 7F DB 16

Note : sur 26 bits, la première lettre ne peut pas dépasser 3, l'adresse physique maximum étant 3 FF FF FF.

5. Quelle est la quantité de mémoire physique utilisée par le processus ? Justifiez.
Correction: *80 Commençons par la mémoire "perdue" en table de pages, répertoire et table des segments. Nous avons deux tables qui contiennent chacune 1024 entrées (10 bits) composées du numéro de cadre (12 bits) et des deux bits de validité et de liberté, soit 2 octest par ligne, soit un total de 2ko. Nous avons aussi un répertoire de 4 lignes de 4 octets (de quoi stocker une adresse*

physique pour chaque ligne) et la table des segments (4 lignes de 5 octets = 20 octets). Nous avons donc un peu plus de 2ko perdus pour la gestion de la mémoire.

Ensuite, il y a plusieurs manières de voir le problème. Soit on regarde les segments, et on constate qu'on a 5 segments dont les tailles sont données par leurs limites respectives. Tous ces segments font moins d'une page, donc on a besoin de 5 pages de 4ko, ce qui nous fait un total de 20ko. Soit on regarde les tables de pages et on constate qu'on a 5 cadres actifs (de 4ko chacun)... ce qui amène au même résultat.

Exercice 9. Pagination Sv32 des processeur RISC-V L'architecture de processeur RISC-V propose différent modes de gestion de la mémoire en fonction des besoins de l'utilisateur. Lorsque les besoins en mémoire sont limités, le mode le plus simple Sv32 est utilisé. Dans ce mode les adresses logiques sont codées sur 32bits, les adresses physiques sur 34bits et une pagination à deux niveaux est utilisée avec des pages standard de 4ko. Le répertoire est les tables de pages contiennent le même nombre de lignes.

1. Quelle est la composition des adresses logiques ?
Correction: *Les pages faisant 4ko, on a un décalage de 12bits, cela laisse 20bits pour le numéro de table et le numéro de page. Les deux tables ayant le même nombre de lignes, chacun de ces numéro est donc codé sur 10bits.*

2. Quelle est la composition des adresses physiques ?
Correction: *On a de même un décalage sur 12bits et donc un numéro de cadre sur 22bits. Ces adresses permettent d'utiliser un total de 16Go de mémoire même si un processus ne peut en exploiter que 4Go à la fois.*

Le répertoire et la table des pages on la même structure, chaque ligne est composé d'un numéro de cadre sur 22bits et de 10bits d'informations supplémentaires parmis lesquels des bits de protection R, W et X indiquant si la ligne peut être utilisée pour un accès en lecture, écriture ou exécution. Pour le répertoire, ces trois bits sont normalement à zéro.

3. Quelle est la taille du répertoire et d'un table des pages ?
Correction: *Chaque ligne fait 32bits soit 4o et dans les deux cas il y en à $2^{10} = 1024$ pour une taille totale de 4ko soit exactement la taille d'une page.*

4. Il n'y a que 22bits disponibles dans le répertoire pour indiquer la position en mémoire physique de la table des pages. Pourquoi n'y en a-t-il pas 34 ?
Correction: *Une table des pages fais 4ko soit exactement la taille d'une page, il est donc logique de lui attribuer exactement un cadre en mémoire physique et de ne stocker que le numéro de ce cadre. Cela simplifie énormément la gestion de ces structure et économise de la mémoire.*

Chaque ligne du répertoire dispose des bits R, W et X qui n'ont pas vraiment de sens si l'on considère qu'une ligne indique seulement ou se trouve la table de pages à consulter. Si par contre on considère que le répertoire découpe la mémoire logique en $2^{10} = 1024$ zones de 4Mo qui sont ensuite découpées en $2^{10} = 1024$ pages de 4ko par les tables de pages correspondantes, on peut améliorer la gestion de la mémoire d'une manière intéressante.

Une ligne du répertoire ou les trois bits R, W et X ont pour valeur 0, indiquent que le numéro de cadre pointe vers une table des pages gérant cette zone de 4Mo sous la forme de petites pages de 4ko. Une autre combinaison de ces trois bits indique que cette zone est gérée sous le forme d'une seule grosse page de 4Mo dont l'adresse physique est donnée par le numéro de cadre.

Cette technique permet d'allouer de gros blocs de mémoire contigüe sans avoir besoin d'utiliser de tables de pages ce qui économise de la mémoire.

5. Quelle quantité de mémoire est économisée par l'utilisation d'une "superpage" par rapport à une table de pages normale ?
Correction: *On économise seulement la place occupée par la table elle-même c'est-à-dire 4ko ce qui est peu. En pratique, le système doit maintenir d'autres structure de données dont la taille dépend du nombre de pages utilisées par exemple pour mémoriser à quelle processus appartient chaque cadre, l'économie réelle est en général 4 à 5 fois plus importante.*

6. Quel autre avantage présent l'utilisation de ces "superpages" ?
Correction: *Lors de la traduction d'adresse la MMU à une table de moins à consulter, donc un accès à la mémoire physique en moins à réaliser ce qui accélère considérablement la vitesse des accès à la mémoire. En pratique, cet avantage est la principale raison à l'utilisation des "superpages".*

L'architecture RISC-V propose d'autres modes de gestion de la mémoire allant jusqu'à 57bits d'adresses logiques pour 66bits d'adresses logiques avec 5 niveaux de pagination. Chaque niveau peu

indiquer soit une table de plus bas niveau, soit une superpage de grande taille. Pour ces modes permettant de gérer de très grandes quantité de mémoire, l'utilisation des superpages deviens critique pour gérer efficacement la mémoire.