Ressource R3.05 - TD 1

Mémoire

1. Soit le programme suivant (on suppose que les adresses et le type `long int`) sont sur 64 bits) :

```
#include <stdlib.h>
#include <stdio.h>

struct ptrs {
    long int** x;
    long int* y;
};
struct ptrs global;

void init(struct ptrs* p) {
    long int* a = malloc(sizeof(long int));
    long int* b = malloc(sizeof(long int));
    long int* c = malloc(sizeof(long int));
    long int* d = malloc(sizeof(long int));
    long int* e = malloc(sizeof(long int) * 2);
    long int** f = malloc(4 * sizeof(long int*));
    long int** g = malloc(sizeof(long int*));

    *a = 0;
    *b = 0;
    *c = (long int) a;
    *d = *b;
    e[0] = 29;
    e[1] = (long int) &d[100000];

    f[0] = b;
    f[1] = c;
    f[2] = 0;
    f[3] = 0;

    *g = c;

    global.x = f;
    global.y = e;

    p->x = g;
    p->y = &e[1];
}

int main(int argc, char** argv) {
    struct ptrs p;
    init(&p);
}
```

En supposant que `a` contient la valeur `0x5573de4342a0`, et que `malloc` a retourné des adresses croissantes, associez correctement les expressions et leurs valeurs (contexte du `main`) :

	Valeur		Epression
1	`0x7ffc159ab650`	A	`global.y`
2	`0x5573de434320`	B	`&global.y`
3	`0x5573de4342a0`	C	`(long int*)*p.y`
4	`0x5573de4f7800`	D	`&p`
5	`0x5573dcdac058`	E	`(long int*)*p.x[0]`

Rappel sur l'ordre de priorité des opérateurs du C :

priorité	opérateur	associativité
16	`[] .`	gauche
15	`* &`	droite
14	`conversion`	droite

2. Donnez les types par ordre croissant de taille :

```
1  char
2  struct minipoint { uint8_t x; uint8_t y; uint8_t z; }
3  int
4  char **
5  double[0]
```

3. Pour chaque déclaration, donnez la taille, l'alignement correspondant :

```
struct s1
{
  float x;
  char n[1];
};
```

```
struct s2
{
  short s;
  char n[3];
};
```

```
struct s3
{
  char Data1;
  short Data2;
  int Data3;
  char Data4;
};
```

```
struct s4
{
    char Data1;
    char Data4;
    short Data2;
    int Data3;
};
```

4. En utilisant une seule fois les membres `char a; unsigned char b; short c; int d;`, et en plaçant `a` en premier, peut on construire une structure de taille 16, 12, 8 ?

5. Une table (simplifiée) des pages possède les entrées suivantes (`U : User` , `P : Present`, `W : Writeable`)

index	numéro physique de la page	permissions
0	0x00	PTE_U
1	0x01	PTE_P
2	0x02	PTE_P|PTE_W
3	0x03	PTE_P|PTE_W|PTE_U
4	0xFF	PTE_W|PTE_U
5	0xFE	PTE_U
6	0x80	PTE_W
7	0x92	PTE_P|PTE_W|PTE_U
8	0xAB	PTE_P|PTE_W|PTE_U
9	0x09	PTE_P|PTE_U
10	0xFE	PTE_P|PTE_U
11	0x00	PTE_W
12	0x11	PTE_U

Pour chaque action ci-dessous, donnez l'adresse physique correspondante, ou identifiez la faute qui sera produite (défaut de page, voilation de privilège, violation de permission). La taille des pages est 1ko.

(a) le noyau déréférence un pointeur null.
(b) le noyau écrit à l'adresse `0x8432`.
(c) un processus utilisateur écrit à l'adresse `0xB123`.
(d) le noyau lit à l'adresse `0x9876`.
(e) un processus utilisateur lit à l'adresse `0x7654`.
(f) un processus utilisateur écrit à l'adresse `0xABCD`
(g) un processus utilisateur écrit à l'adresse `0x2321`.

6. On considère la mémoire conventionnelle dont les paramètres sont :

   - taille adresse : 16 bits;
   - unité d'adressage : 8 bits;
   - mot : 16 bits.

   et la mémoire cache telle que

   - une ligne contient un mot;
   - nombre de ligne dans le cache : 256;
   - le cache est direct

   Dans quelle ligne vont les données d'adresse `0xabcd` ? Quelle le contenu de cette ligne ?

7. (a) Un cache fait 256 Ko, 4-associatif. Ses lignes font 32 octets. l'adresse est 32 bits, et l'unité d'adressage l'octet. Chaque entrée du cache, en plus de l'étiquette, stocke 2 bits de validité, 1 bit de modification et 1 bit de remplacement.
      i. Quelle est la taille de l'étiquette (tag) ?
      ii. Quelle est la taille totale des entrées du cache en bit ?
   (b) Un cache direct de taille 8 Ko est organisé en ligne de 32 octets. Les adresses sont sur 32 bits, l'unité d'adressage est l'octet. Chaque entrée, en plus du tag, stocke 2 bits (validité et modification). Quelle est la taille totale en bit des entrées ?

8. Soit la liste de pages virtuelles référencéees au cours du temps

   7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

   La mémoire centrale est compoée de 3 cases initialement vides. Représentez l'évolution de la mémoire centrale au fur et à mesure des accès pour les deux politiques de remplacement FIFO et LRU. Comparez avec la solution optimale.