

La mémoire principale

M. Dubacq

S1D 2009

1 Alignement de données

Objectif : *Comprendre la déclaration d'un segment de données, comprendre les problèmes d'alignement de données.*

Soit le segment de données suivant sur une machine MIPS 32 bits :

```
.data
part0: .word 0xFE084312
part1: .word 25,32,65,-30
part2: .ascii "paracetamol"
part3: .ascii "totoro"
part4: .asciiz "paracetamol"
part5: .asciiz "dino2"
part6: .half 28
part7: .word 42
part8: .space 15
part9: .space 40 # tableau 10 entiers
```

La déclaration `.asciiz` déclare une chaîne de caractère avec un caractère nul à la fin, contrairement à la déclaration `.ascii`.

1. Sur une machine 32 bits, rappelez la taille prise en mémoire par une donnée de type `word`, `half`, `byte`. Est-ce que ceci changerait sur une machine 64 bits ?
2. Sachant que l'adresse de `part0` est `0x10010000`, dire quelle est l'adresse de `part1`, `part2`, `part3`, `part4`.
3. On s'intéresse à la représentation du premier nombre (à l'adresse `part0`). Quel octet trouve-t-on à l'adresse `0x10010001` si on est sur un processeur *little-endian* ? Et sur un processeur *big-endian* ? Expliquez en faisant une représentation de la mémoire entre l'adresse `part0` et l'adresse `part1` dans les deux cas.
4. Donnez ensuite l'adresse de `part5`. Expliquez la différence avec `part3`.
5. Donnez l'adresse de `part6`. Expliquez pourquoi.
6. Donnez de même l'adresse de `part7`.
7. Concluez en donnant l'adresse de `part8` et de `part9`. Que va-t-il se passer lorsqu'on essaiera de faire `Load R1,part9` ?

2 Overclocking et mémoire

Objectif : *Comprendre les rapports numériques entre la mémoire et le processeur.*

L'*overclocking* est une technique qui consiste en l'accélération de l'horloge d'un processeur pour obtenir des performances meilleures. Toutefois, la cadence du bus de données, du northbridge et de la mémoire sont toutes déterminées à partir de la cadence du processeur en divisant par un entier (le multiplicateur). Cet entier est, sur certaines cartes-mères, réglable.

Une RAM est prévue pour fonctionner à 333 MHz. La vitesse du processeur qui va avec est de 3 GHz. En fait, cette RAM est capable de fonctionner sans problèmes jusqu'à 400 MHz, et le processeur peut être accéléré jusqu'à 10% sans défauts (il peut être accéléré uniquement par paliers de 100 MHz).

1. Expliquez pourquoi accélérer modérément la cadence du processeur peut amener à des meilleures performances, et pourquoi l'accélérer énormément peut ne pas fonctionner du tout.
2. Quel est le multiplicateur utilisé pour obtenir la cadence nominale ?
3. Peut-on réaliser une accélération de la cadence du processeur qui ne crée aucun défaut sans changer le multiplicateur ? Que deviendrait la vitesse de la mémoire si on le faisait ?
4. Peut-on avoir une accélération du processeur et de la mémoire qui privilégie un peu plus la mémoire ?
5. Si on a un bus de données de largeur 32 bits, quel est le débit de la mémoire ? Quel est son débit maximal ?

3 Déboguage en C

3.1 TP en salle machine

Objectif : *Savoir se servir d'un débogueur.*

1. En salle machine, connectez-vous sous Linux. Récupérez le programme `td09.c`. Le programme est disponible sur <http://www-info.iutv.univ-paris13.fr/jcdubacq/td09.c>
2. Ce programme comporte des erreurs. Essayez de les repérez, ne les corrigez pas. Le but de ce programme est de remplir un tableau avec des valeurs, puis de faire des modifications de ces valeurs (multiplier par deux, puis ajouter 1).
3. Compilez le programme, et exécutez-le avec le débogueur `ddd`. Le listing du programme doit s'afficher.
4. Commencez, avant de faire tourner le programme par insérer un point d'arrêt au début du `main`, un point d'arrêt dans la toute première boucle, un point d'arrêt dans la fonction `fact`.
5. Faites afficher la fenêtre de données de `ddd`. En faisant un clic droit, faites afficher toutes les variables du programme. Que se passe-t-il si on essaye d'afficher `NUM`? Que remarque-t-on quand on essaye d'afficher la variable `i` de la fonction `fact`?
6. Remarquez la différence d'affichage entre `b` et `c`. À quoi cela peut-il être dû?
7. Dans un terminal, testez le programme. Vérifiez qu'il ne fonctionne pas.
8. Commencez l'exécution du programme (bouton `run` ou menu équivalent). Après le premier arrêt, utilisez le bouton `step` pour avancer pas à pas dans le programme. Regardez évoluer peu à peu les valeurs des variables. Que remarque-t-on sur la valeur affichée de `i` lorsqu'on rentre dans l'évaluation de la fonction `fact`?
9. Quelle est la première erreur? Corrigez-là, recompilez (de l'intérieur du débogueur, avec le bouton `make`).
10. Refaites une exécution pas-à-pas. Vérifiez que la fonction renvoie maintenant la bonne valeur. Que constate-t-on dans l'affichage du tableau `b`? Trouvez la deuxième erreur.
11. Faites afficher *en partant de la zone de données* le contenu pointé par `c`. Quelle est la troisième erreur? Corrigez-la.
12. Quelle est l'adresse du premier élément du tableau `b`?
13. Surveillez pas-à-pas l'exécution de la deuxième boucle du programme (qui multiplie par 2). Que remarque-t-on sur le tableau `b`? Expliquez.
14. Il serait plus simple de faire afficher `c` comme un tableau de 6 éléments. Utilisez la syntaxe donnée dans le cours pour le faire. Vérifiez que les valeurs sont changées aussi bien dans `c` que dans `b`.
15. La dernière boucle utilise l'arithmétique des pointeurs : un pointeur (`c`) change de valeur et pointe tour à tour. Elle comporte une erreur. Trouvez-la et corrigez-la.
16. Est-il correct d'affecter une valeur à `*d`? Pourquoi?

3.2 Pour ceux qui finissent en avance

- Réécrivez le programme précédent en faisant de `b` non pas un tableau statique, mais un tableau alloué dynamiquement. N'oubliez pas de libérer l'espace mémoire à la fin du tableau.
- Est-il possible de lire dans ce tableau après avoir libéré la mémoire?
- Après avoir libéré la mémoire (à la fin du programme), créez un nouveau tableau d'entiers (de taille similaire mais pas identique). Que constate-t-on?