TD8 : Visite de la mémoire

Première année d'IUT Informatique

1 Compléments à propos de GDB

`info address étiquette` : pour voir l'adresse d'une étiquette.
`break *address` : pour placer un point d'arrêt à une adresse donnée du code

2 Stockage des nombres

Les nombres sont stockés en mémoire de deux façons :
- little endian, c'est le cas du pentium où en mémoire les nombres sont stockés octets de poids faibles en tête.
- big endian, comme sur les SPARCs, les G4 et G5, ici le stockage correspond à celui des registres.

Nous allons donc faire une série de manipulations pour nous familiariser avec ce mode de stockage mémoire. Pour ceci nous plaçons dans le segment de données les variables suivantes :

```
.data
entier: .long 22,0xabcdef,0b0111,07263
court: .word  12,011,0b11,0x2ff
octet: .byte 12,255,0xa2
.text
.globl main
main:  mov $1,%eax
       mov $0,%ebx
       int $0x80
```

1. Récupérer l'adresse de l'étiquette `entier`. Comment ces quatre valeurs sont-elles stockées ? En quelle base chacune de ces valeurs sont-elles données ?
2. Ajouter au programme précédent les lignes nécessaires pour placer dans les registres `%eax` `%ebx`, `%ecx` et `%edx` des valeurs immédiates identiques. Comment sont elles stockées dans les registres ?
3. Faites de même avec les étiquettes `court` et `octet`.
4. Placer la ligne correspondant à l'étiquette octet en début de segment. Quelle est maintenant l'adresse de entier ? est-ce un multiple de 4 ? Est-ce que la directive `.align 4` est une solution ?

3 Les modes d'adressage

Pour chaque question, assemblez et examinez sous `gdb`. La commande `readelf -s programme` donne la table des symboles avec les adresses correspondantes.

1. Ajouter au programme précédent les instructions permettant de placer dans les registres `%eax` `%ebx`, `%ecx` le premier élément de `entier`, `court` et `octet`.
2. Quelles instructions doit on ajouter pour déplacer les quatre valeurs de `entier` dans `%eax` `%ebx`, `%ecx` et `%edx`
3. Avec certaines directives nous pouvons réserver de l'espace en mémoire.
   – `.skip size , value` ou `.space size , value`
   – `.fill repeat , size , value`
   où `size` représente un nombre d'octets (limité à 8 pour `.fill`), `value` est une valeur,et `repeat` un nombre de répétitions.
   Ajouter les lignes suivantes au segment de données et vérifier en mémoire l'application de ces directives

   ```
   titi:   .fill 12,7,5
   toto: .skip 12,7
   ```

4. Ajouter des instructions permettant de déplacer le contenu de `entier` dans `titi`.
5. Les chaînes de caractères sont de deux types `.string` et `.ascii`, la première ajoute un caractère 0 en fin de chaîne (utile pour certaines fonctions).

   ```
   chaine: .ascii "voici une chaine ascii"
   phrase: .string "voici une phrase"
   ```

   Insérer les deux lignes ci-dessus en début de segment de données, puis vérifier sous `gdb` le contenu de la mémoire.
6. Ajouter les instructions permettant d'afficher ces textes.

CORRECTION

4 Stockage des nombres

1. ```
   (gdb) break main
   Breakpoint 1 at 0x80483c0
   (gdb) info address entier
   Symbol "entier" is at 0x80494a7 in a file compiled without debugging.
   (gdb)
    22,  base dix
   0xabcdef,  hexadécimal
   0b0111, binaire
   07263  octal
   ```
2. ```
   mov $22,%eax
   mov $0xabcdef,%ebx
   mov $0b0111,%ecx
   mov $07263, %edx
   ```
3. idem
4. ```
   (gdb) info address entier
   Symbol "entier" is at 0x80494b7 in a file compiled without debugging.
   (gdb) x /100xb 0x80494b7
   0x80494b7 <entier>:     0x16    0x00    0x00    0x00    0xef    0xcd    0xab    0
    à voir avec les étudiants
   ```

5

1. % readelf -s testeur

   ```
       55: 080494b7     0 NOTYPE  LOCAL  DEFAULT   15 entier
       56: 080494c7     0 NOTYPE  LOCAL  DEFAULT   15 court
       57: 080494cf     0 NOTYPE  LOCAL  DEFAULT   15 chaine
       58: 080494e5     0 NOTYPE  LOCAL  DEFAULT   15 octet
   ...
       67: 080483c0     0 NOTYPE  GLOBAL DEFAULT   12 main
   ```

   et

   ```
   mov  entier,%eax
   mov court,%bx
   mov octet,%cl
   ```
2. ```
   lea entier,%esi
   mov (%esi),%eax
   mov 4(%esi),%ebx
   mov 8(%esi),%ecx
   mov 12(%esi),%edx
   ```
   et

```
(gdb) info reg eax ebx ecx edx
eax            0x16     22
ebx            0xabcdef 11259375
ecx            0x7      7
edx            0xeb3    3763
(gdb)
```

3.
```
(gdb) info address titi
Symbol "titi" is at 0x8049474 in a file compiled without debugging.
(gdb) x /100xb 0x8049474
0x8049474 <titi>:   0x05    0x00    0x00    0x00    0x00    0x00    0x00    0x05
0x804947c <titi+8>: 0x00    0x00    0x00    0x00    0x00    0x00    0x05    0x00
0x8049484 <titi+16>: 0x00    0x00    0x00    0x00    0x00    0x05    0x00    0x00
0x804948c <titi+24>: 0x00    0x00    0x00    0x00    0x05    0x00    0x00    0x00
0x8049494 <titi+32>:0x00    0x00    0x00    0x05    0x00    0x00    0x00    0x00
0x804949c <titi+40>:0x00    0x00    0x05    0x00    0x00    0x00    0x00    0x00
0x80494a4 <titi+48>:0x00    0x05    0x00    0x00    0x00    0x00    0x00    0x00
0x80494ac <titi+56>:0x05    0x00    0x00    0x00    0x00    0x00    0x00    0x05
0x80494b4 <titi+64>:0x00    0x00    0x00    0x00    0x00    0x00    0x05    0x00
0x80494bc <titi+72>:0x00    0x00    0x00    0x00    0x00    0x05    0x00    0x00
0x80494c4 <titi+80>:0x00    0x00    0x00    0x00    0x07    0x07    0x07    0x07
0x80494cc <toto+4>: 0x07    0x07    0x07    0x07    0x07    0x07    0x07    0x07
0x80494d4 <bidule>: 0x0c    0xff    0xa2    0x16
```

4.
```
lea entier,%esi
mov (%esi),%eax
mov 4(%esi),%ebx
mov 8(%esi),%ecx

lea titi,%esi
mov %eax,(%esi)
mov %ebx,4(%esi)
mov %ecx,8(%esi)
```

5.
```
(gdb) info address chaine
Symbol "chaine" is at 0x8049484 in a file compiled without debugging.
(gdb) x /50cb 0x8049484
0x8049484 :    118 'v' 111 'o' 105 'i' 99 'c'  105 'i' 32 ' '  117 'u' 110 'n'
0x804948c :  101 'e' 32 ' '  99 'c'  104 'h' 97 'a'  105 'i' 110 'n' 101 'e'
0x8049494 : 32 ' '  97 'a'  115 's' 99 'c'  105 'i' 105 'i' 118 'v' 111 'o'
0x804949c <phrase+2>:   105 'i' 99 'c'  105 'i' 32 ' '  117 'u' 110 'n' 101 'e' 32 ' '
0x80494a4 <phrase+10>:  112 'p' 104 'h' 114 'r' 97 'a'  115 's' 101 'e' 0 '\0'  5 '\005
0x80494ac <titi+1>:     0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  0 '\0'  5 '\005'
```