Université Paris-Nord — L1
Institut Galilée — Année 2011-2012
Licence 1 - 1er semestre

Travaux Pratiques 1 : Programmation en mini-assembleur

Il est demandé, pendant ce TP, d'écrire des programmes simples en mini-assembleur et de les exécuter sur un processeur simulé en `amil` (*assembleur miniature pour l'informatique de licence*). Le simulateur peut être utilisé en ligne ici :
`http://www-lipn.univ-paris13.fr/~boudes/amilweb/`

Prise en main : le terminal

Faîtes vous aider par votre chargé de TP pour ajouter un lanceur de terminal dans votre barre d'outils ou votre barre de menu, ainsi qu'un lanceur pour l'éditeur de texte (`gedit`).

Dans un terminal, taper les lignes de commandes suivantes :

`mkdir TP1`	Créer un répertoire TP1.
`cd TP1`	Entrer dans le répertoire TP1.

Dans la suite, vous gagnerez à taper vos programme dans l'éditeur de texte, les charger dans le simulateur par copier/coller, et à ***les enregistrer*** *dans le répertoire TP1 (sous des noms comme* `exercice1.txt`*,* `exercice2_1.txt`*, etc.).*

`gedit exercice1.txt &`	Lance l'édition d'un fichier `exercice1.txt`, en *tâche de fond.*

Astuces du terminal. la touche de tabulation vous permet de compléter votre saisie quand vous tapez une commande dans le terminal. La touche flèche vers le haut rappelle une ligne de commande tapée précédemment.

1 Initialisation de la mémoire

Soit la case mémoire, x, d'adresse 10 et la case mémoire, y, d'adresse 11. Écrire et exécuter le programme qui initialise x à 7×2 et y à $x - 1$.

Correction.

```
1  valeur 7 r0
2  mult 2 r0
3  ecriture r0 10
4  valeur -1 r1
5  add r1 r0
6  ecriture r0 11
7  stop
```

2 Exécution conditionnelle d'instructions

À l'aide de l'instruction `sautpos`, écrire les programmes correspondant aux algorithmes suivants et les exécuter avec amil sur un exemple, afin de tester leur correction :

1. Soient la valeur a à l'adresse 20, b à l'adresse 21. Si $a < b$ alors écrire a à l'adresse 22 sinon écrire b à l'adresse 22.

Correction.

– Calcul de $a - b$:

```
1   lecture 20 r0
2   lecture 21 r1
3   soustr r1 r0     # r0 vaut a - b
```

– Si $a < b$ (c'est à dire si $a - b < 0$)

```
4   sautpos r0 8
```

– Alors écrire a à l'adresse 22

```
5   lecture 20 r2
6   ecriture r2 22
7   saut 10
```

– Sinon écrire b à l'adresse 22

```
8   lecture 21 r2
9   ecriture r2 22
10  stop
```

– Données :

```
20  3  # a
21  2  # b
22  ?  # résultat
```

2. Soient trois cases mémoires contenant trois entiers. Calculer et écrire le minimum de ces trois entiers en mémoire.

Correction.

– Soient a, b et c trois entiers

```
20  12 # a
21  23 # b
22  6  # c
23  ?  # min
```

– min est initialise à a (par défaut)

```
1   lecture 20 r0
2   ecriture r0 23
```

– Si $b <$ min alors min vaut b

```
3   lecture 21 r0
4   lecture 23 r1
5   inverse r1
6   add r0 r1    # r1 vaut b - min
7   sautpos r1 9
8   ecriture r0 23
```

– Si $c <$ min alors min vaut c

```
9   lecture 22 r0
10  lecture 23 r1
11  inverse r1
12  add r0 r1 # r1 vaut c - min
13  sautpos r1 15
14  ecriture r0 23
```

– min contient le minimum de a, b et c

```
15  stop
```

3 Boucles d'instructions

1. Avec l'instruction `saut`, écrire un programme qui ne termine jamais.

Correction.

```
1   saut 1
2   stop # <- jamais atteint
```

2. Avec l'instruction `sautpos`, écrire un programme qui ne termine jamais.

Correction.

```
1   valeur 0 r0
2   sautpos r0 2
3   stop # <- jamais atteint
```

On dit de ces programmes qu'ils bouclent à l'infini.