

# Licence 1 - section B

## TD 3 d'éléments d'informatique

Catherine RECANATI – Département d'Informatique – Institut Galilée

Semaine du 14 au 18 novembre 2016

### Jeu d'instructions du mini-assembleur AMIL

stop	Arrête l'exécution du programme.
noop	N'effectue aucune opération.
saut i	Met le compteur de programme à la valeur $i$ .
sautpos ri j	Si la valeur contenue dans le registre $i$ est positive ou nulle, met le compteur de programme à la valeur $j$ .
valeur x ri	Initialise le registre $i$ avec la valeur $x$ .
lecture i rj	Charge, dans le registre $j$ , le contenu de la mémoire d'adresse $i$ .
écriture ri j	Écrit le contenu du registre $i$ dans la mémoire d'adresse $j$ .
inverse ri	Inverse le signe du contenu du registre $i$ .
add ri rj	Ajoute la valeur du registre $i$ à celle du registre $j$ (la somme obtenue est placée dans le registre $j$ ).
soustr ri rj	Soustrait la valeur du registre $i$ à celle du registre $j$ (la différence obtenue est placée dans le registre $j$ ).
mult ri rj	Multiplie par la valeur du registre $i$ celle du registre $j$ (le produit obtenu est placé dans le registre $j$ ).
div ri rj	Divise par la valeur du registre $i$ celle du registre $j$ (le quotient obtenu, arrondi à la valeur entière inférieure, est placé dans le registre $j$ ).

#### Instructions plus avancées

et ri rj	Effectue le et bit à bit de la valeur du registre $i$ et de celle du registre $j$ . Le résultat est placé dans le registre $j$ .
lecture *ri rj	Charge, dans le registre $j$ , le contenu de la mémoire dont l'adresse est la valeur du registre $i$ .
écriture ri *rj	Écrit le contenu du registre $i$ dans la mémoire dont l'adresse est la valeur du registre $j$ .

## 1 Quelques petits programmes

### Exercice 1.1 Echange de valeurs

Ecrire un programme qui échange les valeurs contenues dans les mémoires d'adresses 15 et 16 et s'arrête.

### Exercice 1.2 Initialisation de la mémoire

Écrire les programmes traduisant les algorithmes suivants en mini-assembleur :

1. Soit la valeur 8 contenue dans la case mémoire d'adresse 10. Recopier cette valeur à l'adresse 11.
2. Sans connaître la valeur contenue à l'adresse 10, incrémenter cette valeur de 1.
3. Soit une valeur quelconque,  $a$ , contenue à l'adresse 10. Initialiser la case d'adresse 11 à  $a \times 2 + 1$ .
4. Soit une valeur quelconque,  $a$ , contenue à l'adresse 10. Initialiser la case d'adresse 11 à  $a \times (2 + 1)$ .

## 2 Trace de programme assembleur

Nous allons produire une représentation de l'exécution pas à pas de nos programmes. La *trace* d'un programme assembleur sera un tableau dont chaque ligne correspondra à l'exécution d'une instruction. Une colonne contiendra le numéro du cycle d'horloge du processeur, et une autre le compteur de programme (ligne de la prochaine instruction). Il y aura en outre une colonne par registre utilisé, et une par case mémoire où des données sont lues ou écrites par les instructions du programme. La première ligne, notée état initial, montrera l'état des registres et cases mémoires utilisés

avant le début du programme. Ensuite, chaque exécution d'une ligne d'instruction sera représentée par une nouvelle ligne du tableau, jusqu'à exécution de l'instruction d'arrêt stop. Sur chaque ligne nous représenterons le numéro du cycle d'horloge du processeur (en commençant à 0 sur la ligne d'initialisation), la valeur du compteur de programme après exécution de l'instruction, et, s'il y a lieu, la valeur du registre ou de la case mémoire modifiée par le programme.

A titre d'exemple, le programme ci-dessous, suivi du tableau constituant sa trace :

```

1 lecture 10 r0
2 valeur 5 r2
3 soustr r2 r0
4 sautpos r0 8
5 valeur 0 r0
6 ecriture r0 11
7 saut 9
8 ecriture r0 11
9 stop
10 14
11 ?

```

Registre d'instructions	Cycles	CP	r0	r2	10	11
(etat initial)	0	1	?	?	14	?
lecture 10 r0	1	2	14			
valeur 5 r2	2	3		5		
soustr r2 r0	3	4	9			
sautpos r0 8	4	8				
ecriture r0 11	5	9				9
stop	6	10				

### Exercice 2.1 Trace de programme assembleur

Faire la trace du même programme où la valeur 14 de la ligne d'initialisation est remplacée par la valeur 2, à l'adresse mémoire 10. Quelles sont les valeurs contenues dans les registres r0 et r2 après arrêt sur l'instruction stop ?

## 3 Instructions de contrôle

### Exercice 3.1 Saut conditionnel

À l'aide de l'instruction `sautpos`, écrire les programmes correspondant aux algorithmes suivants et les exécuter sur un exemple (en faisant une trace) afin de tester leur correction :

- Soient la valeur  $a$  à l'adresse 15, et la valeur  $b$  à l'adresse 16. Si  $a \geq b$  alors écrire  $a$  à l'adresse 17 sinon écrire  $b$  à l'adresse 17.
- Soit la donnée  $x$  d'adresse 15. On veut écrire la valeur absolue de  $x$  à l'adresse 16.
  - Décrire un algorithme (en langage naturel) réalisant cette tâche.
  - Écrire un programme réalisant cette tâche.
  - Construire une trace de votre programme lorsque  $x$  vaut 5, puis lorsque  $x$  vaut  $-5$ .

## 4 Boucles d'instructions

**Exercice 4.1** Avec l'instruction `saut`, écrire un programme qui ne termine jamais. Même question avec l'instruction `sautpos`.

### Exercice 4.2 Boucle d'écriture

Soit un entier  $n$  d'adresse 15. Écrire un programme qui lorsque  $n \geq 0$ , écrit la valeur  $n$  à l'adresse 16, puis, toujours à l'adresse 16, écrit la valeur  $n - 1$ , puis  $n - 2$ , et ainsi de suite jusqu'à écrire 0, après quoi le programme termine. Si  $n$  est négatif, le programme ne fait rien.