FONDEMENTS DE LA PROGRAMMATION

MASTER 1 INFORMATIQUE 2017-2018
INSTITUT GALILÉE - UNIVERSITÉ PARIS 13

Paulin de Naurois - Domenico Ruoppolo
(d'après un cours par Virgile Mogbil et Pierre Boudes)

TD 3: Machines à adressage indirect concurrente (CRAM)

Remarque: à partir de maintenant vous êtes autorisé.e.s à utiliser non seulement les macros vues en séance précédente pour les SRAM, c'est-à-dire `goto` ℓ | $\mathtt{X_i} := \mathit{constante}$ | $\mathrm{X}_i := \mathtt{X_j}$, mais aussi les instructions suivantes, également dérivables dans toute SRAM:
$\mathtt{X_i} := \mathtt{X_j} + \mathtt{X_z}$ | $\mathtt{X_i} := \mathtt{X_j} \dot{-} \mathtt{X_z}$ | `if` $\mathtt{X_i} = \mathtt{X_j}$ `goto` ℓ `else` ℓ' | `if` $\mathtt{X_i} < \mathtt{X_j}$ `goto` ℓ `else` ℓ' | `if` $\mathtt{X_i} \leq \mathtt{X_j}$ `goto` ℓ `else` ℓ' | `if` $\mathtt{X_i} = \mathit{constante}$ `goto` ℓ `else` ℓ' | `if` $\mathtt{X_i} < \mathit{constante}$ `etc.`

Exercice 3. Décrire une CRAM qui prend en entrée un nombre naturel n, deux suites de naturels $(a_i)_{1\leq i\leq n}$ et $(b_i)_{1\leq i\leq n}$ de longueur n, et qui rend $\sum_{i=1}^{n} i\,a_i + (n+1-i)\,b_i$. Par exemple, pour $n = 3$ l'on retrouvera $a_1 + 3b_1 + 2a_2 + 2b_2 + 3a_1 + b_3$ en sortie (i.e. dans le compteur $\mathtt{G_0}$).

Solution. En entrée, dans le processeur centrale $\mathtt{G}$, le compteur $\mathtt{G_0}$ contient la valeur n et les compteurs $\mathtt{G_1}, \ldots, \mathtt{G_n}$ contiennent les valeurs $a_1, \ldots, a_n$.

L'idée de l'algorithme est de faire calculer à chaque processeur SRAM $\mathtt{X^k}$ le terme ka_k et aussi le terme $(n+1-k)b_k$, pour qu'il puisse ensuite les écrire respectivement dans $\mathtt{G_k}$ et $\mathtt{G_{2k}}$.

Voici un programme CRAM pour cela. Pour tout $k \geq 1$ en parallèle, je copie n en $\mathtt{X_0^k}$:

```
1:  X0 := G0
```

Pour tout $k \geq 1$, je copie a_k dans $\mathtt{X_2^k}$ (à ce but, le processeur $\mathtt{X^k}$ met son propre adresse k dans $\mathtt{X_1^k}$):

```
2:  X1 := PID
3:  if X2 :=< X1 >^g
```

L'on remarquera que formellement même la macro utilisée en ligne 1 est faite par un adressage indirect à la mémoire centrale.

Maintenant l'on calcule ka_k dans $\mathtt{X_4^k}$ (ici on copie k également dans $\mathtt{X_3^k}$ pour pouvoir le décrementer sans le perdre dans $\mathtt{X_1^k}$, mais cela n'est pas vraiment nécéssaire, puisque à tout moment on peut utiliser `PID` pour retrouver l'adresse):

```
4:  X3 := X1
5:  X4 := 0
6:  if X3 = 0 goto 10 else 7
7:  X4 := X4 + X2
8:  X3 := X3 ∸ 1
9:  goto 6
```

Le résultat est copié dans la mémoire centrale:

```
10 :  < X1 >^g := X4
```

On fait quelque chose de similaire pour le calcul de $(n+1-k)b_k$:

```
11:  X5 := X0 + X1
12:  X6 := < X5 >^g
13:  if X7 = X0 + 1 goto 10 else 7
14:  X7 := X7 ∸ X1
15:  X8 := 0
16:  if X7 = X0 + 1 goto 20 else 17
17:  X8 := X8 + X6
18:  X7 := X7 ∸ 1
19:  goto 16
20:  < X5 >^g := X8
```

Maintenant que tous les termes de la somme à calculer sont dans la SRAM centrale `G`, il ne reste qu'à les sommer là-dedans (ci-dessous `P` et `A` sont des compteurs auxilière de `G`):

```
21:  P := 1
22:  A := G0 + G0
23:  G0 := 0
24:  if P ≤ A goto 25 else 28
25:  G0 = G0 ∸ < P >
26:  P := P + 1
27:  goto 24
28:
```