TD n°1

Premiers pas avec une machine à a-pile

Exercice 1. Exécutez à la main la machine virtuelle sur l'états initials (Pile vide, Accumulateur quelconque, PC=0) associés aux trois programmes suivants :

ex1 : Consti 3; Push; Consti 2: Addi
ex2 : Push; Addi; Push; Addi; Push; Consti 3; Addi
ex3 : Consti 3; Pop; Push

Le jeu d'instructions de la machine virtuelle présenté en cours est insuffisant pour implémenter des fonctions simples comme la multiplication. Nous ajoutons donc les instructions suivantes :

- `Acc n :` copie dans l'accumulateur A la valeur du n-ème élément de la pile S sans dépiler l'élément (la tête de la pile étant le 0-ème element) ;
- `Assign n :` écrit la valeur de l'accumulateur comme n-ème élément de la pile S (en écrasant la valeur précédente de ce n-ème élément et en supposant que la pile contient au moins $n + 1$ éléments). L'accumulateur prend la valeur 0 ;
- `Popi n :` dépile les premiers n éléments du sommet de la pile S (c'est tout) ;
- `Branch n :` saute n instructions en avant (ou en arrière si n est négatif), c'est-à-dire $PC = PC+n$;
- `Branchif n :` saute n instructions si l'accumulateur A est égal à 1, sinon il avance à l'instruction suivante ;
- `Leqi :` dépile un mot n de S, remplace A par 1 si $n \leq A$, 0 sinon

Qu'est-ce qu'il devrait se passer si l'argument `n` dépasse les limites de la pile dans `Acc n`, `Assign n` ou `Popi n` ? et s'il dépasse le tableau d'instructions dans `Branch n` ou `Branchif n` ?

Exercice 2. Exécutez à la main la machine virtuelle sur les états initials (Pile vide, Accumulateur quelconque, PC=0) associés aux programmes suivants :

ex4 : Consti 3; Push; Branchif 5; Consti -1; Addi; Push; Branch -4; Popi 1
ex6 : Consti 2; Push; Consti 1; Push; Acc 0; Push; Acc 2; Assign 1; Pop; Assign 1

Exercice 3. Programmer la conjunction : c'est-à-dire un programme tel que l'exécution sur

$$A = m \text{ et } S = n$$

termine avec :

$$\begin{cases} A = 0, S = \varepsilon & \text{si } m = 0 \text{ ou } n = 0, \\ A = 1, S = \epsilon & \text{sinon.} \end{cases}$$

Exercice 4. Programmer la multiplication : $m, n \to m \times n$, c'est-à-dire un programme tel que l'exécution sur

$$A = m \text{ et } S = n$$

termine avec

$$A = m \times n \text{ et } S = \epsilon$$

On supposera m et n strictement positifs. Avec un tel language basique, il faut un programme alambiqué pour calculer la multiplication. Dans les prochaines TP nous utiliserons un jeu d'instructions plus riche, contenant la multiplication comme instruction primitive.

Exercice 5. Programmer la fonction de Fibonacci : c'est-à-dire un programme tel que l'exécution sur

$$A = n \text{ et } S = \varepsilon$$

termine avec

$$A = F(n) \text{ et } S = \varepsilon$$

où $F(n)$ est l'n-ème numero di Fibonacci, défini par $F(0) = F(1) = 1$, et $F(n+2) = F(n) + F(n+1)$. On supposera n strictement positif. Verifier que le programme proposé soit correct, en executant à la main les premiers cas pour $n \leq 3$.