Master STL - M1
Module CA (MI190)

Compilation Avancée
Documents autorisés

Université Pierre et Marie Curie
Année 2005–2006

Examen du 20 juin 2006

1ère partie (10 points) - à traiter sur une copie séparée

1ère partie : interprétation et compilation d'un langage

Le but de ce problème est de réaliser une implantation du langage EXAM, langage à pile à la FORTH ou à la PostScript. Le texte suivant est repris de l'encyclopédie Wikipedia sur FORTH.

langage EXAM : Description

EXAM est un langage à pile simple et extensible. Une de ses importantes caractéristiques est l'utilisation d'une pile de données pour passer des arguments entre les «mots», qui sont les constituants d'un programme EXAM.

Exemple

L'expression `2+3*4` qui s'écrit en polonaise inversée en `2 3 4 * +` est considérée comme une suite de mots dont l'évaluation agit sur la pile de données de la manière suivante :

- d'empiler successivement les valeurs 2, 3, puis 4
- de remplacer ensuite les 2 nombres du sommet de la pile (3 et 4) par leur produit 12.
- et enfin de remplacer les 2 nombres en haut de pile (2 et 12) par leur somme 14.

Opérateurs

Les commentaires entre parenthèses indiquent l'effet qu'a l'opérateur sur la pile, qui par convention croît vers la droite. Les commentaires de fin de ligne commencent au caractère \.

- opérateurs de pile

```
DUP  ( n -- n n )       \ dupliquer le sommet de pile
DROP ( n -- )           \ supprimer le sommet de pile
SWAP ( a b -- b a )     \ échanger
ROT  ( a b c -- b c a ) \ rotation
```

- opérateurs arithmétiques

```
*, /, +, -, mod   ( a b -- entier )
```

- opérateurs de comparaison

```
=, <>, <, > ,<=, >=   ( a b -- booléen)
```

Définitions

Le programmeur construit le vocabulaire de son application en définissant ses propres mots. La définition d'un nouveau mot suit la grammaire suivante :

définition := `: NOM liste_de_mots ;`
liste_de_mots := `mot*`

L'appel à un nouveau mot est identique aux opérateurs prédéfinis. Un mot EXAM est l'équivalent des sous-programmes, fonctions ou procédures dans les autres langages.

```
: CARRE DUP * ;       ( définition de CARRE )
11 CARRE              ( on obtient 121 en sommet de la pile )
: CUBE DUP CARRE * ;  ( usage de CARRE dans une définition )
2 CUBE                ( on obtient 8 en sommet de la pile )
```

Enfin on introduit une structure de contrôle conditionnelle pour les définitions de mots en suivant la règle syntaxique suivante :

```
conditionnelle := liste_de_mots IF liste_de_mots THEN
                | liste_de_mots IF liste_de_mots ELSE liste_de_mots ENDIF
```

On peut alors définir le mot `FACTORIELLE` de la manière suivante :

```
: FACTORIELLE  DUP 1 >  IF  DUP 1 - FACTORIELLE * THEN ;
```

Les mots d'EXAM peuvent être interprétés et compilés, c'est-à-dire convertis en une forme exécutable et ajoutés au dictionnaire des mots. La forme exécutable diffère suivant le compilateur/interpréteur utilisé: génération directe de code machine ou "bytecode" par exemple.

Exécution

La syntaxe d'EXAM est tellement simple qu'elle nécessite ni analyse syntaxique, ni analyse lexicale. En fait un système EXAM a deux états possibles. L'état par défault est le mode interactif, qui correspond à la boucle suivante :

1. lire un mot sur l'entrée standard (on reconnait la fin d'un mot par la présence d'une espace ou retour chariot) ;
2. chercher la définition du mot dans le dictionnaire ;
3. Si le mot est défini alors l'exécuter, sinon essayer de le convertir en nombre pour mettre sur la pile.
4. aller à 1.

Le deuxième mode de fonctionnement du système est le mode compilation, qui permet d'ajouter de nouvelles définitions au dictionnaire. On a déjà vu le mot spécial :, qui permet de rentrer en mode compilation. Ainsi la phrase `: CARRE DUP * ;` crée une nouvelle entrée `CARRE` dans le dictionnaire et y met le code du corps de la définition (adresses des mots DUP et *). Le `;` permet de revenir en mode interactif.

Questions

1. trace : Soit le mot `TUCK` suivant : `: TUCK DUP ROT SWAP ;` Indiquer ce qu'il fait en décrivant l'état de la pile `a b c --` avant et après son appel.
2. interprétation : Ecrire le mode interpétation pour le langage EXAM sous la forme d'une fonction prenant un dictionnaire, une pile des données et si besoin est une pile de contrôle (pour les appels de mots). On supposera que le code d'un mot correspond à la liste de mots de sa définition. Indiquer tout d'abord le mécanisme général de votre évaluateur, puis l'implanter en utilisant un langage connu (O'Caml, Java ou C) sans être nécessairement pointilleux sur la syntaxe. Dans le cas où un mot n'existe pas dans le dictionnaire, l'exécution s'arrête.
3. machine abstraite : Pour pouvoir augmenter les performances, on cherche à définir une machine abstraite pour compiler le corps d'une définition sous la forme d'une liste d'instructions de la machine abstraite. Indiquer la structure de la machine (registre, piles, ...) et le comportement pour l'opérateur DUP ainsi que le mécanisme d'appel aux sous-programmes (mots).
4. compilation : Donner les schémas de compilation de l'opérateur DUP, de la structure IF et de l'appel à un sous-programme.
5. liaison statique : On cherche à ajouter la liaison statique ou portée lexicale (tous les mots d'une définition sont connus au moment de la définition et une redéfinition de ceux-ci n'interfèrent pas dans les définitions passées) au langage EXAM. Donner une description d'un tel mécanisme en indiquant ce qu'il faut changer dans la représentation du dictionnaire, de la machine abstraite et des schémas de compilation pour avoir dans les définitions des mots la liaison statique ? Vous pouvez vous inspirer des fermetures des langages fonctionnels.
6. extension : Indiquer sans les implanter les modifications à apporter au langage, interprète, compilateur et machine abstraite pour intégrer un mécanisme d'exceptions.