

Python

Java

Théorie des langages de programmation

Le projet (V)

Alain Chillès – 祁冲

ParisTech Shanghai Jiao Tong
上海交大-巴黎高科卓越工程师学院

23 novembre 2020 – 2020年11月23日 – 庚子九月初九

Plan

Projet

Compilation d'une fonction L_{AC}

Compilation d'une structure conditionnelle

Traitement des chaînes de caractères

Compilation d'une déclaration

Lisp

Java

C

Prolog

Forth

C

Ada

Pascal

Lisp

APL

Fortran

Position du problème

Soit les fonctions définies par

L_{AC}

```
1  : incr 1 + ;  
2  
3  : 2+. incr incr . ;
```

Elle vont être définies dans la table des symboles par

Numéro	x	$x + 1$	$x + 2$	$x + 3$	$x + 4$	$x + 5$	$x + 6$
LAC	?	4	105	110	99	114	y
Signification	a	/	i	n	c	r	c

Numéro	$x + 7$	$x + 8$	$x + 9$	$x + 10$	$x + 11$	$x + 12$
LAC	$x + 1$	3	50	43	46	z
Signification	a	/	2	+	.	c

Position du problème

et en machine virtuelle par

Numéro	y	$y + 1$	$y + 2$	$y + 3$	$y + 4$	$y + 5$	$y + 6$	$y + 7$	$y + 8$	$y + 9$
VM	1	c_1	1	c_2	c_3	1	y	y	c_4	c_3

où

- c_1 est le Cfa de (lit)
- c_2 est le Cfa de +
- c_3 est le Cfa de (fin)
- c_4 est le Cfa de .

On a donc

$$z = y + 5$$

Plan

Projet

Compilation d'une fonction L_{AC}

Compilation d'une structure conditionnelle

Traitement des chaînes de caractères

Compilation d'une déclaration

Comment coder en machine virtuelle les ... if... [else...] then

- La syntaxe en L_{AC} est

$\langle cond \rangle$ if $\langle sivrαι \rangle$ else $\langle sifaux \rangle$ then

ou

$\langle cond \rangle$ if $\langle sivrαι \rangle$ then

- Il faut aussi être capable de gérer les conditions imbriquées, et ne pas se tromper sur le then et le else !

$\langle cond \rangle$ if $\langle cond' \rangle$ if $\langle action_1 \rangle$ else $\langle action_2 \rangle$ then...

Conditionnelle en machine virtuelle

Java

Le codage en machine virtuelle est facile

L_{AC}

```
1  ... 0= if 1 else 2 then ...
```

...	x	x + 1	x + 2	x + 3	x + 4	x + 5	x + 6	x + 7	x + 8	...
	c ₁	c ₂	x + 6	c ₃	1	c ₄	x + 8	c ₃	2	

où

- c₁ est le Cfa de 0=
- c₂ est le Cfa de if
- c₃ est le Cfa de (lit)
- c₄ est le Cfa de else

Pascal

Fortran

Conditionnelle dans la table des symboles

- Il faut signifier que **if**, **else** et **then** ne sont utilisables que pendant la compilation !
- Notons que **then** n'est même pas codé dans la machine virtuelle ! (Si cela vous manque, vous pouvez toujours créer une fonction qui ne fait rien !)
- Il doit y avoir une erreur sémantique, si les **if**, **else** et **then** ne sont pas bien organisés entre eux (on vide alors toutes les piles).

Plan

Projet

Compilation d'une fonction L_{AC}

Compilation d'une structure conditionnelle

Traitement des chaînes de caractères

Compilation d'une déclaration

Les chaînes de caractères

À la compilation

L_{AC}

```
1 : essai ( -- ) " Bonjour" count type ;
```

À l'exécution

L_{AC}

```
1 " Bonjour" count type
```

Utilisation des chaînes en mode compilé

Pas de problème, cela donne en machine virtuelle

x	x + 1	x + 2	x + 3	x + 4	x + 5	x + 6	x + 7	x + 8	x + 9	x + 10	x + 11	x + 12
1	c ₁	7	66	111	110	106	111	117	114	c ₂	c ₃	c ₄

où

- x est le Cfa de **essai**
- c₁ est le Cfa d'une fonction (**str**) qui indique qu'arrive une chaîne de caractères
- c₂ est le Cfa de **count**
- c₃ est le Cfa de **type**
- c₄ est le Cfa de (**fin**)

Utilisation des chaînes en mode interprété

- Où mettre la chaîne **tant que l'on en a besoin ?**
- Comment mettre les chaînes sans qu'elles se superposent ?

L_{AC}

```
1  " Bonjour" " Alain !" catenate
```

Plan

Projet

Compilation d'une fonction L_{AC}

Compilation d'une structure conditionnelle

Traitement des chaînes de caractères

Compilation d'une déclaration

Déclaration (non demandé, mais intéressant)

1 y union

Java

L_{AC}

```
1  defer u \ Déclaration de u
2
3  : v ( n -- v[n] )
4      dup 0= if drop 1 else dup 1- u 2 * swap 1-
        ↪ recurse - then ;
5
6  : (u) ( n -- u[n] )
7      dup 0= if drop 1 else dup 1- u swap 1- v +
        ↪ then ;
8
9  ' (u) is u \ Résolution de la déclaration
10
11  4 u .
```

Problèmes posés

En machine virtuelle ? **Aucun !**

x	$x + 1$	$x + 2$
1	c_1	c_2

où

- x est le Cfa de u
 - c_1 est le Cfa d'une fonction d'erreur qui dit que la fonction n'est pas définie (et vide les piles !)
 - c_2 est le Cfa de la fonction (**fin**)
- À la résolution de la déclaration, on remplace tout simplement c_1 par le Cfa de la fonction (**u**)

Compléments

Les fonctions `'` et `is` ne fonctionnent qu'en mode interprété. Elles sont définies de la manière suivante :

- `'` trouve le Cfa de la fonction qui suit (et qui n'est donc pas exécutée), elle est donc du type

`' (— — ad)`

- `is` trouve le Cfa de la fonction qui suit (et qui n'est donc pas exécutée) et remplace c_1 par l'adresse qui est sur la pile. Elle est donc du type

`is (ad — —)`