
Introduction aux machines virtuelles

Cours 2 : La Machine Virtuelle Caml

Pierre Letouzey

Prenom.Nom@pps.jussieu.fr

A la découverte d'un cmo

Phrase exemple: $1+2+3*4/5$ dans test.ml

Exploration brut: **hexedit** sur test.cmo

Le message secret au milieu du cmo:

103.5.108.4.107.112.113.105.127.2.110.58.57.0

Les instructions de bytecode associées:

CONSTINT 5

PUSHCONSTINT 4

PUSHCONST3

MULINT

DIVINT

PUSHCONST1

OFFSETINT 2

ADDINT

(... plus 2 autres sans intérêt)

A la découverte d'un cmo

Exploration structurée:

- `dumpobj` donne directement le bytecode
- `objinfo` donne quelque infos sur la zone finale

Mieux encore:

- `ocaml -dinstr` pour voir le bytecode au vol

Aperçu d'un programme complet

- avec hexedit
- avec dumpobj

Que fait ocamlrun ?

Voir la fonction `caml_interprete` dans `byterun/interp.c`

Moralement, une grande boucle contenant un switch sur toutes les instructions de bytecode possible.

Cf <http://cadmium.x9c.fr/distrib/caml-instructions.pdf>

Elements de la VM Caml

Éléments principaux:

- un programme (liste de bytecodes)
- un registre **pc** : pointeur de code dans le programme
- un registre **accu**
- une **pile** $\text{accu} + \text{pile} = \text{a-pile}$
- un **tas** (heap) où placer les données **allouées**

Éléments qui nous intéresseront moins:

- un environnement global (pour les choses “à toplevel”)
- un pointeur **env** vers une archive d’environnement
- un registre **extra_args**

Retour sur l'exemple

instruction	accu	pile
CONSTINT 5	5	...
PUSHCONSTINT 4	4	5 ...
PUSHCONST3	3	4 5 ...
MULINT	12	5 ...
DIVINT	2	...
PUSHCONST1	1	2 ...
OFFSETINT 2	3	2 ...
ADDINT	5	...

Instructions de la VM Caml

Gestion de la pile

- **PUSH** : empilement de **accu** sur la **pile**
- **POP n** : dépilement de **n** éléments au sommet de la **pile**
- **ACC n** : **accu** reçoit la valeur de la **n**-ième case de la **pile** ($n=0$ pour le sommet de pile). Variantes:
 - **ACC0 ... ACC7** : instructions spécialisées pour $n < 8$
 - **PUSHACC n** : on sauve **accu** via **PUSH** avant un **ACC n**
 - **PUSHACC0 ... PUSHACC7** : cf. **PUSHACC n**
- **ASSIGN n** : le **n**-ième élément de la **pile** est remplacé par le contenu de **accu**. Celui-ci est rempli avec **()**.

Arithmétique

- **CONSTANT n** : l'entier **n** est placé dans **accu**. Variantes:
 - **CONST0 ... CONST3** : cf **CONSTANT** pour $n < 4$
 - **PUSHCONSTANT n** : **PUSH** + **CONSTANT n**
 - **PUSHCONST0...3** : cf. **PUSHCONSTANT n**
- **OFFSETINT n** : **accu** est incrémenté de **n**
- **NEGINT** : **accu** reçoit l'opposé de sa valeur.
- **ADDINT** : **accu** reçoit **accu** + sommet de la **pile** (qui est dépilé au passage). Cf. vision sous forme de **a-pile**.
- idem avec les autres opérateurs: **SUBINT**, **MULINT**, **DIVINT**, **MODINT**, **ANDINT**, **ORINT**, **XORINT**, **LSLINT**, **LSRINT**, **ASRINT**, **(U)LTINT**, **LEINT**, **GTINT**, **(U)GEINT**, **(N)EQ**.

Aiguillages

- **BRANCH** *o* : saut obligatoire vers *o* instructions plus loin (ou en arrière selon le signe de *o*).
- **BRANCHIF** *o* : saute de *o* instructions si **accu** représente **true**
- **BRANCHIFNOT** *o* : saute de *o* instructions si **accu** représente **false**

Variantes:

- **BEQ** *n o* : compare *n* avec le contenu de **accu** et saute de *o* si le test est positif.
- Idem avec **BNEQ**, **B(U)LTINT**, **BLEINT**, **BGTINT**, **B(U)GEINT**.

A suivre

Mine de rien, on a déjà vu environ la moitié des instructions de la VM Caml. La suite au prochain numéro...