VM

vm-exec-move

Cette fonction déplace une valeur entre deux registres ou entre un registre et une valeur littérale.

- Si la source (*src*) est une valeur littérale (vérifiée via is-litteral), on extrait sa valeur (cadr src) et on l'affecte au registre de destination avec set-registre.
- Si *src* n'est pas une valeur littérale, il est considéré comme un registre dont la valeur est obtenue avec get-registre et affectée au registre de destination.

Cette distinction est essentielle pour savoir si on doit utiliser directement la valeur ou la récupérer depuis un registre.

vm-exec-push

Prend en argument une machine virtuelle (*nom*) et une source (*src*), puis pousse la valeur de *src* sur la pile.

- Si la pile est pleine, une erreur est levée.

vm-exec-jmp

Met à jour le compteur de programme (*PC*) avec la valeur du label s'il s'agit d'un entier, permettant ainsi de sauter à l'instruction correspondante dans le code machine.

vm-exec-rtn

Simule l'opération de retour d'une machine virtuelle.

- La pile est réduite pour replacer le pointeur sur le dernier élément ajouté (une adresse).
- L'exécution reprend à cette adresse, permettant ainsi de retourner à une sous-routine précédemment interrompue.

vm-exec-resoudre-refNR

Utilise map pour parcourir une liste de références non résolues obtenue par getReferenceNR nom symb. Pour chaque adresse mémoire (*co*), la fonction :

1. Obtient l'instruction stockée à *co* avec get-memoire.
2. Extrait le premier élément de l'instruction avec car.
3. Récupère la valeur actuelle du symbole avec getSymbole.
4. Crée une instruction avec le premier élément original et la nouvelle valeur du symbole.
5. Met à jour la mémoire à *co* avec set-memoire.

Compilateur

comp-defun

Compile une déclaration de fonction en Lisp vers un langage à pile.

- Initialise la position dans la pile.
- Associe chaque paramètre de la fonction Lisp à une position dans la pile.
- Génère des instructions : saut initial, étiquette de début, compilation de l'expression, gestion de la pile, retour et étiquette de fin.

Objectif : rendre la fonction Lisp exécutable dans un environnement à pile.

comp-call

Génère des instructions pour un appel de fonction en respectant les conventions d'appel :

1. Compile chaque argument et le pousse sur la pile.
2. Sauvegarde l'ancien pointeur de pile de cadres.
3. Met à jour ce pointeur pour la nouvelle position de la pile.
4. Pousse le nombre d'arguments et met à jour la pile.
5. Compile l'appel de fonction.
6. Restaure les pointeurs de pile.

comp-primitive-call

Génère du code bas niveau pour les appels de fonctions arithmétiques et les comparaisons.

Affichage et Expressions

Fonctions d'affichage

- **getline** : Affiche le numéro de ligne et sa valeur pour chaque élément d'une liste.
- **printem** : Affiche une liste d'arguments séparés par des espaces.

Expressions conditionnelles

- **generate-label** : Génère des étiquettes uniques pour les instructions.
- **compile-section** : Compile une section de code avec un saut vers une étiquette.
- **comp-if** : Compile une instruction conditionnelle if en langage bas niveau.

Expressions déclaratives

- **comp-var** : Génère du code pour récupérer la valeur d'une variable.
- **comp-defun** : Compile une déclaration de fonction en Lisp.

Compilation des expressions

- **comp-expr** : Identifie le type d'expression (ég. constante, symbole, if, defun) et appelle la fonction appropriée (comp-var, comp-cons, comp-if, comp-defun).
- **comp-list** : Compile une liste d'expressions.

Gestion des littéraux

- **comp-cons** : Charge un littéral dans un registre.

Procédures

- **comp-call** : Compile un appel de fonction, en gérant correctement la pile et les pointeurs.
- **comp-primitive-call** : Traduit les appels de fonctions arithmétiques et de comparaison en code bas niveau.