**Explication Li2Asm**

Le but de Li2Asm est de transformer le langage intermédiaire généré au préalable par Lisp2LI en un ensemble d’instructions ASM qui seront chargé en mémoire par vm-load puis exécutés par vm-run.

Pour ce faire, il me semble que la meilleur solution est de faire un CASE avec chaque cas:

:LIT, :CAR, :VAR, :SETVAR, :IF, :CALL, :MCALL, :PROGN, :LET, :COND, :CASE

et d’appeler la fonction qui exécutera le bon pattern.

Il existe des cas qui seront exécutes par vm-run de façon consécutive, le cas de base est :LIT qui fait simplement un MOVE mais il existe des cas qui ne doivent pas s'exécuter consécutivement, c’est le cas du :IF où l’on doit vérifier la condition pour savoir si on exécute le bloque si vrais ou le bloque si faux.

*(if condition (bloque si vrais) (bloque si faux))*

Pour pouvoir aller dans un bloque ou dans l’autre il existe une instruction qui s'appelle JMP. (voir poly page 62 65 et 66 pour en savoir plus)

JMP fait un “saut” à une adresse donnée mais Li2Asm ne peut pas connaître les adresses pour cela il faut utiliser des LABELs qui seront résolut en adresses par vm-load.

Par exemple (:IF (:CALL < (:LIT . 1) (:LIT . 2)) (...) …) savoir si 1 plus petit que 2:

I) Appel de l’instruction de comparaison CMP (CMP (:CONST 1) (:CONST 2))

pour l’exemple j’utilise :CONST mais on peut utiliser des registre…

II) Appel du jump JGE (JGE monLabelSiFaux ) si le test a dit plus grand ou égale on jump directement dans le bloque faux

III) Appel des instructions pour le bloque vrais et finir par un (JMP finSi)

IV) Débuter le bloque faux avec le label (LABEL monLabelSiFaux) et le finir avec (JMP finSi)

V) Finir le test avec le label (LABEL finSi)

notre if donne donc:

(CMP (:CONST 1) (:CONST 2)) (JGE monLabelSiFaux ) (BLOQUE V) (JMP finSi) (LABEL monLabelSiFaux) (BLOQUE F) (JMP finSi) (LABEL finSi)