Conventions d’appels

Règles définissant le protocole d’échange de valeurs entre une fonction appelante et une fonction appelée, notamment :

* Comment sont passés les arguments (appelante → appelée)
* Comment est passée l’adresse de retour (appelante → appelée)
* Comment est passée la valeur de retour (appelée → appelante)
* Quels registres sont à sauvegarder, comment et par qui (appelée ou appelante)

Les conventions d’appel sont définies dans l’Application Binary Interface d’une architecture

Utilité et importance

* Les respecter permet à une fonction d’être appelée par tout code qui connait sa signature (nom, nombre, ordre et type des arguments)
* Les respecter permet d’appeler une fonction juste en connaissant sa signature
* Les respecter permet d’écrire du code compatible avec du code qui les respecte aussi : fonctions de bibliothèques, code généré par un autre compilateur, code assembleur écrit par collègue ou une entité tierce

Fonctions imbriquées et récursives

* Fonctions imbriquées : il n’y a rien de plus à faire pour appeler une fonction depuis une fonction que de suivre scrupuleusement les conventions d’appel !
* Fonctions récursives : c’est une fonction qui s’appelle elle-même mais cela ne change rien, il suffit de suivre les conventions d’appel

Fonctions

1. Alloue son contexte d’exécution sur la pile
2. Sauvegarde les registres persistants qu’elle utilise + $31
3. Récupère la valeur des arguments de l’appel dans $4-$7 + les suivants sur la pile
4. Réalise un traitement à partir des valeurs des arguments
5. Place la valeur du résultat dans $2
6. Restaure les registres persistants + $31
7. Désalloue son contexte d’exécution 8 Effectue un saut à l’adresse de retour avec jr $31

Élément d’une suite récursive

* La fonction u\_n s’appelle elle-même
* C’est un appel comme un autre, le problème de la terminaison est algorithmique et non au niveau assembleur
* Ici la récursion est non terminale