**Adresse de retour :**

* L’adresse de retour dépend du site d’appel, elle correspond à l’adresse qui suit le saut vers le code de la fonction appelée
* L’adresse de retour est passée a la fonction appelée via le registre $31
* Le registre $31 est écrit avec l’adresse de l’instruction en séquence dans le code par toutes les instructions dont le nom se termine par al signifiant and link
* Le registre $31 doit être enregistré en pile si une fonction utilise une fonction pour y retourne

**Valeur de retour :**

La valeur de retour est passée a la fonction appelante via le registre $2 (+$3 si taille>32 bits)

**Passage de paramètres :**

* Les 4 premiers paramètres sont passés par les registres $4, $5, $6, $7
* Les paramètres suivants sont passés dans la pile :
  + Emplacements dédiés dans le contexte d’exécution de la fonction appelante sont nécessaires + écriture de le leur valeur juste avant d’aller exécuter le code de la fonction appelée
* Une place est TOUJOURS réservée sur la pile pour les 4 premiers paramètres

Explication : La fonction peut aussi appeler d’autre fonction donc utiliser les registres $4, $5, $6, $7

**Fonction et appels de fonctions en assembleur :**

* Fonctions :
  + Morceau de programme qui :
    - * Reçoit des arguments
      * Revoie un résultat
      * Peut accéder aux variables globales
      * A des variables locales qui lui sont propres
* Appel de fonction
  + Valeur : celle du résultat renvoyé par la fonction
* Effet :
  + Aller exécuter le code d’une fonction
  + Avec les valeurs spécifiées pour ses arguments
  + En revenir avec la valeur du résultat

**Ecriture d’une fonction :**

1. Se représenter la pile à l’entrée de la fonction
2. Ecrire le code du corps de la fonction
   * On commence par le corps car on ne connait pas les registres persistants
   * Ecrire les lectures et écriture en pile (de paramètres où variables locales)
   * Choisir les registres qu’in veut associer avec les variables locales
3. Déterminer la taille du contexte

**Appel de la fonction f :**

* Passage des paramètres dans $4-$7 + pile
* Saut à la fonction f avec passage de l’adresse de retour dans $31 avec l’instruction jal f
* Récupération de la valeur de retour dans $2

**Convention Mips : sauvegarde des registres :**

* La convention Mips définit des registres dits persistants : ce sont ceux dont la valeur doit rester intègre au travers les appels de fonction.
  + Registres persistants : $16-$23
  + Si une fonction utilise ces registres elle doit
    1. Sauvegarder sur la pile leur valeur dans son prologue
    2. Restaurer leur valeur à partir la pile dans son épilogue
* Les registres sont sauvegardés par ordre de numéro croissant des plus petites adresses aux plus grandes sur la pile : le registre de plus petit numéro est le plus proche du sommet pile.
* Dans le prologue, il faut allouer de la place sur la pile pour la sauvegarder des registres persistants + $31 puis y écrire leur contenu
* Dans l’épilogue, il faut restaurer les registres puis désallouer les emplacements
* Si une fonction n’utilise pas ces registres, une seule sauvegarde=celle de $31