Explicitation de la convention d'appel (de RTL à ERTL)

David Delahaye

David.Delahaye@lirmm.fr

Faculté des Sciences

Master M1 2019-2020

Expliciter la convention d'appel

Convention d'appel

La convention d'appel est le résultat d'un contrat entre plusieurs parties :

- l'appelant (« caller ») ;
- l'appelé (« callee ») ;
- le système d'exécution (« runtime system ») ;
- le système d'exploitation (« operating system »).

Elle permet d'abord la communication entre appelant et appelé.
Elle peut dans certains cas être exploitée par le système d'exécution (par exemple, par le glaneur de cellules) ou par le système d'exploitation (par exemple, par un gestionnaire d'interruptions).

Expliciter la convention d'appel

Quand expliciter la convention d'appel ?

La convention d'appel est toujours invisible dans le langage source, car celui-ci est de haut niveau.

L'expliciter fait apparaître de nouvelles notions : registres physiques, trames de pile, emplacements de pile. Cela complique le langage intermédiaire, donc autant le faire aussi tard que possible.

Néanmoins, l'allocation de registres dépend de la convention d'appel. Nous explicitons donc la convention d'appel immédiatement avant la phase d'allocation de registres.

ERTL (« Explicit Register Transfer Language »)

Dans ERTL, la convention d'appel est explicitée

- Paramètres et, le cas échéant, résultat des procédures et fonctions sont transmis à travers des registres physiques et/ou des emplacements de pile ;
- Procédures et fonctions ne sont plus distinguées ;
- L'adresse de retour devient un paramètre explicite ;
- L'allocation et la désallocation des trames de pile devient explicite ;
- Les registres « callee-save » sont sauvegardés de façon explicite.

Un exemple de traduction

Fonction factorielle (en récursif)

$f(n : \text{integer}) : \text{integer}$
    if $n = 0$ then
        $f := 1$
    else
        $f := n \times f(n-1)$

Un exemple de traduction

Traduction en ERTL

procedure $f(1)$
var %0, %1, %2, %3, %4, %5, %6
entry $f11$

$f11$: newframe $\rightarrow f10$
$f10$: move %6, \$$ra \rightarrow f9$
$f9$: move %5, \$$s1 \rightarrow f8$
$f8$: move %4, \$$s0 \rightarrow f7$
$f7$: move %0, \$$a0 \rightarrow f6$
$f6$: li %1, 0 $\rightarrow f5$
$f5$: blez %0 $\rightarrow f4, f3$
$f3$: addiu %3, %0, $-1 \rightarrow f2$
$f2$: j $\rightarrow f20$
$f20$: move \$$a0$, %3 $\rightarrow f19$
$f19$: call $f(1) \rightarrow f18$
$f18$: move %2, \$$v0 \rightarrow f1$
$f1$: mul %1, %0, %2 $\rightarrow f0$
$f0$: j $\rightarrow f17$
$f17$: move \$$v0$, %1 $\rightarrow f16$
$f16$: move \$$ra$, %6 $\rightarrow f15$
$f15$: move \$$s1$, %5 $\rightarrow f14$
$f14$: move \$$s0$, %4 $\rightarrow f13$
$f13$: delframe $\rightarrow f12$
$f12$: jr \$$ra$
$f4$: li %1, 1 $\rightarrow f0$

Nouvelles instructions en ERTL

Registres physiques

- « move » d'un registre physique vers un pseudo-registre, et vice versa.

Pile

- Passage de paramètres sur la pile (si les registres physiques $a0-$a3 ne suffisent pas, c'est-à-dire il y a plus de 4 paramètres à passer) ;
- Écriture dans un emplacement « sortant » de la pile ;
- Lecture depuis un emplacement « entrant » de la pile.

Utilisation de la pile

Régions de pile

- Chaque procédure utilise la pile pour accéder aux paramètres qu'elle a reçus d'un appelant et pour transmettre des paramètres aux procédures qu'elle-même appelle ;
- Du point de vue de chaque procédure, on distingue deux régions de la pile : celle des paramètres entrants et celle des paramètres sortants ;
- Après l'allocation de registres, chaque procédure aura besoin d'une troisième région pour contenir des données locales.

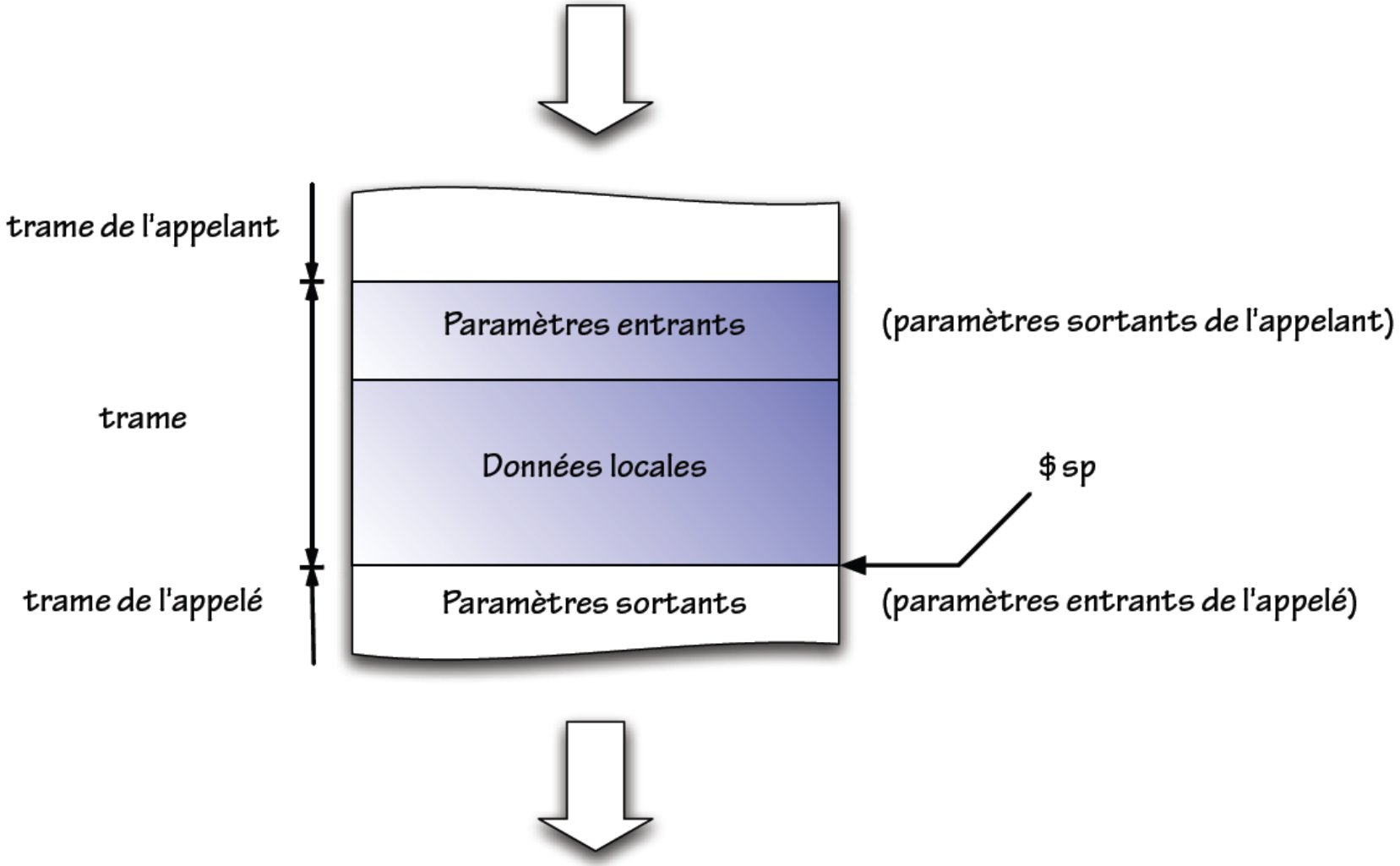
Utilisation de la pile

Trames de pile

- Chaque procédure a sa propre vision de la pile ;
- On appelle trame de pile (« stack frame ») la portion de pile allouée pour chaque procédure ;
- Chaque trame est sous-divisée en régions :
  - La région des paramètres entrants ;
  - La région des paramètres sortants ;
  - La région des données locales (n'existe pas en ERTL).
- Lors d'un appel de procédure, la région des paramètres sortants de l'appelant coïncide avec celle des paramètres entrants de l'appelé ; c'est ainsi que communiquent appelant et appelé ; ceci exige bien sûr de s'accorder sur l'emplacement exact de chaque paramètre.

Utilisation de la pile

Trames de pile



Utilisation de la pile

Régions et emplacements

- Un emplacement de pile est identifié par la région dans laquelle il se trouve et par un décalage (« offset ») au sein de cette région ;
- Les décalages commencent à 0 et augmentent de 4 en 4 à chaque fois qu'un nouvel emplacement est alloué ;
- Ces emplacements seront traduits en décalages vis-à-vis de $sp lors d'une phase ultérieure, une fois la taille de chaque région connue.

Nouvelles instructions en ERTL

Trames de pile

- newframe : début de trame, $sp est décrémenté ;
- delframe : fin de trame, $sp est incrémenté ;
- La taille de la décrémentation et de l'incrémentation de $sp ne sera connue qu'au terme de l'allocation de registres ;
- L'accès aux pseudo-registres et aux emplacements de pile n'est permis qu'après avoir exécuté newframe et avant d'exécuter delframe.

Nouvelles instructions en ERTL

Appels et retours de procédures

- call $f(n)$:
  - Appelle la procédure f avec n paramètres effectifs ;
  - les 4 premiers sont passés par les registres physiques \$a0-\$a3, le reste est passé sur la pile ;
  - L'adresse de retour est stockée dans le registre physique \$ra.
- Retour de procédures : simple saut (inconditionnel) à l'adresse stockée par l'appelant dans \$ra.

À propos des registres paramètres $a0-$a3

Quelques remarques

- Ils sont à sauvegarder (dans des pseudo-registres) s'il y a des appels de procédures dans la procédure en question et que ces paramètres sont utilisés après l'un de ces appels ;
- Les pseudo-registres utilisés pour la sauvegarde de ces paramètres peuvent être réalisés :
  - Soit par des registres physiques « callee-save » (certainement pas par des « caller-save » car si on les sauvegarde c'est qu'il y a d'autres appels de procédures et les « caller-save » ne sont pas sauvegardés par les procédures appelées) ;
  - Soit en le « spillant » (en le mettant sur la pile) ; dans ce cas, il n'y a aucune différence avec les mécanismes de passage de paramètres qui utilisent exclusivement la pile.

Exercice

Fonction factorielle avec sauvegarde par « callee-save »

```
f17:  addiu $sp, $sp, -8
      sw $ra, 4($sp)
      sw $s0, 0($sp)
      move $s0, $a0
      blez $s0, f4
      addiu $a0, $s0, -1
      jal f17
      mul $v0, $s0, $v0
f28:  lw $ra, 4($sp)
      lw $s0, 0($sp)
      addiu $sp, $sp, 8
      jr $ra
f4:   li $v0, 1
      j f28
```

Combien de lw concernant $a0 ?

Exercice

Fonction factorielle avec sauvegarde par « spill »

```
f17: addiu $sp, $sp, -8
     sw $ra, 4($sp)
     sw $a0, 0($sp)
     lw $t0, 0($sp)
     blez $t0, f4
     ...
```

Écrire la suite.
Combien de lw concernant $a0 ?