Sélection d'instructions (de PP à UPP)
Création du graphe de flot de contrôle (de UPP à RTL)

David Delahaye

David.Delahaye@lirmm.fr

Faculté des Sciences

Master M1 2017-2018

De PP à UPP

Dans UPP

- Les types sont supprimés (après vérification normalement, mais on ne le fera pas) ;
- Les variables globales et locales sont distinguées (dans les fonctions et procédures) ; les variables globales seront désignées par leurs adresses ;
- Les opérateurs d'accès aux tableaux sont remplacés par les opérations MIPS `lw` et `sw` ;
- Les opérateurs arithmétiques sont remplacés par ceux du MIPS.

Les deux derniers points constituent la sélection d'instructions.

Suppression des types

Dans UPP

- Les types sont simplement supprimés de l'AST ;
- Si on supprime les types, comment connaître la taille allouée à une variable (car son type détermine sa taille) ?
  - Toute variable sera de taille fixe : 1 mot = 4 octets = 32 bits ;
  - C'est toujours possible (pour les entiers, booléens, ou tableaux).

Variables

Dans UPP

- On va distinguer les variables locales et les variables globales, qui peuvent apparaître dans les fonctions et procédures ;
- Pour les variables globales, deux choix possibles (rien d'imposé) :
  - Les variables gardent leurs noms : elles seront traduites par des labels MIPS dans la zone de données (solution assez simple) ;
  - Les variables sont désignées par un « offset » (déplacement) dans la zone de données, qui a un seul label MIPS (solution plus générale).

Vous choisirez une solution dans votre implantation.

Accès aux tableaux

Dans UPP

- Les accès en lecture de la forme $e_1[e_2]$ sont traduits en un accès mémoire en lecture lw $(e_1 + 4 \times e_2)$;
- Les accès en écriture de la forme $e_1[e_2] := e_3$ sont traduits en un accès mémoire en écriture sw $(e_1 + 4 \times e_2)\ e_3$;
- Les allocations de tableaux de la forme new array of τ $[e]$ sont traduits en alloc $(4 \times e)$.

Opérations arithmétiques

Dans UPP

- Les opérations arithmétiques sont traduites en opérations arithmétiques MIPS ;
- Toutes les opérations arithmétiques de PP se retrouvent dans MIPS, à l'exception du « - » unaire, qui doit, a minima, être traduit en UPP :
  - Traduction : $-e \rightarrow 0 - e$.
- Le reste, ce sont des optimisations. Par exemple :
  - $1 + 2 \rightarrow 3$ (calculs faits par le compilateur) ;
  - $e + 0 \rightarrow e$ (simplifications) ;
  - $e + 1 \rightarrow \text{addi}(e, 1)$ (addition avec un immédiat).

Implantation

Code Java

- Couches PP et UPP : « PP.java » et « UPP.java » ;
- Structures de données imposées à partir de maintenant ;
- Fonctions de transformation à écrire par vos soins ;
- Fonction de transformation « toUPP » dans « PP.java ».

Organisation

- Écrire la transformation « toUPP » ;
- Éventuellement, coder des « printers » pour PP et UPP ;
- Prochain rendu : **11 octobre 2017**.

RTL (« Register Transfer Language »)

Le langage

- Expressions et instructions structurées sont décomposées en instructions élémentaires organisées en graphe de flot de contrôle ;
- Les variables locales sont remplacées par des pseudo-registres (en nombre infini et locaux à chaque fonction/procédure).

Intuitivement

- On met tout à plat (pour les calculs complexes) ;
- Le graphe représente les chemins possibles de l'exécution.

RTL (« Register Transfer Language »)

Motivations

- L'organisation en graphe facilite l'insertion ou la suppression d'instructions par les phases d'optimisation ultérieures ;
- Elle est simple et générale : elle peut refléter toutes les constructions (while, repeat, for, if, case, break, continue, et même goto) ;
- La structure arborescente des expressions, exploitée lors de la sélection d'instructions, ne sera plus utile au-delà.

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 RTL

function $f(\%0) : \%1$
var $\%0, \%1, \%2, \%3$
entry $f6$
exit $f0$
$f6 : \text{li } \%1, 0 \rightarrow f5$
$f5 : \text{blez } \%0 \rightarrow f4, f3$
$f3 : \text{addiu } \%3,\ \%0, -1 \rightarrow f2$
$f2 : \text{call } \%2, f(\%3) \rightarrow f1$
$f1 : \text{mul } \%1, \%0, \%2 \rightarrow f0$
$f4 : \text{li } \%1, 1 \rightarrow f0$

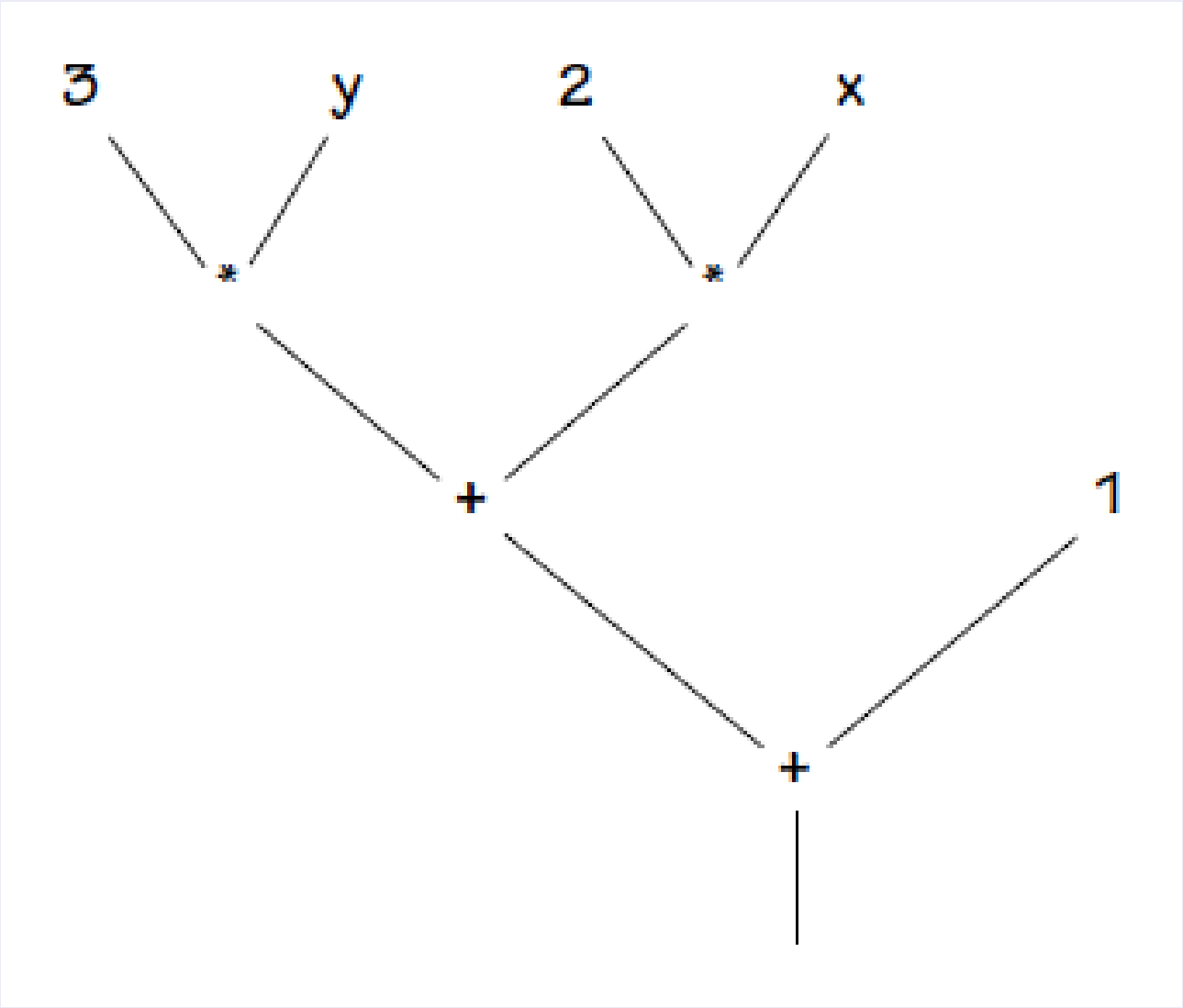
Un exemple de traduction

Remarques

- Paramètres, résultat, variables locales sont des pseudo-registres ;
- Le graphe est donné par ses labels d'entrée et de sortie et par une table qui à chaque label associe une instruction ;
- Chaque instruction mentionne explicitement le ou les labels de ses successeurs (pour les instructions de branchement).

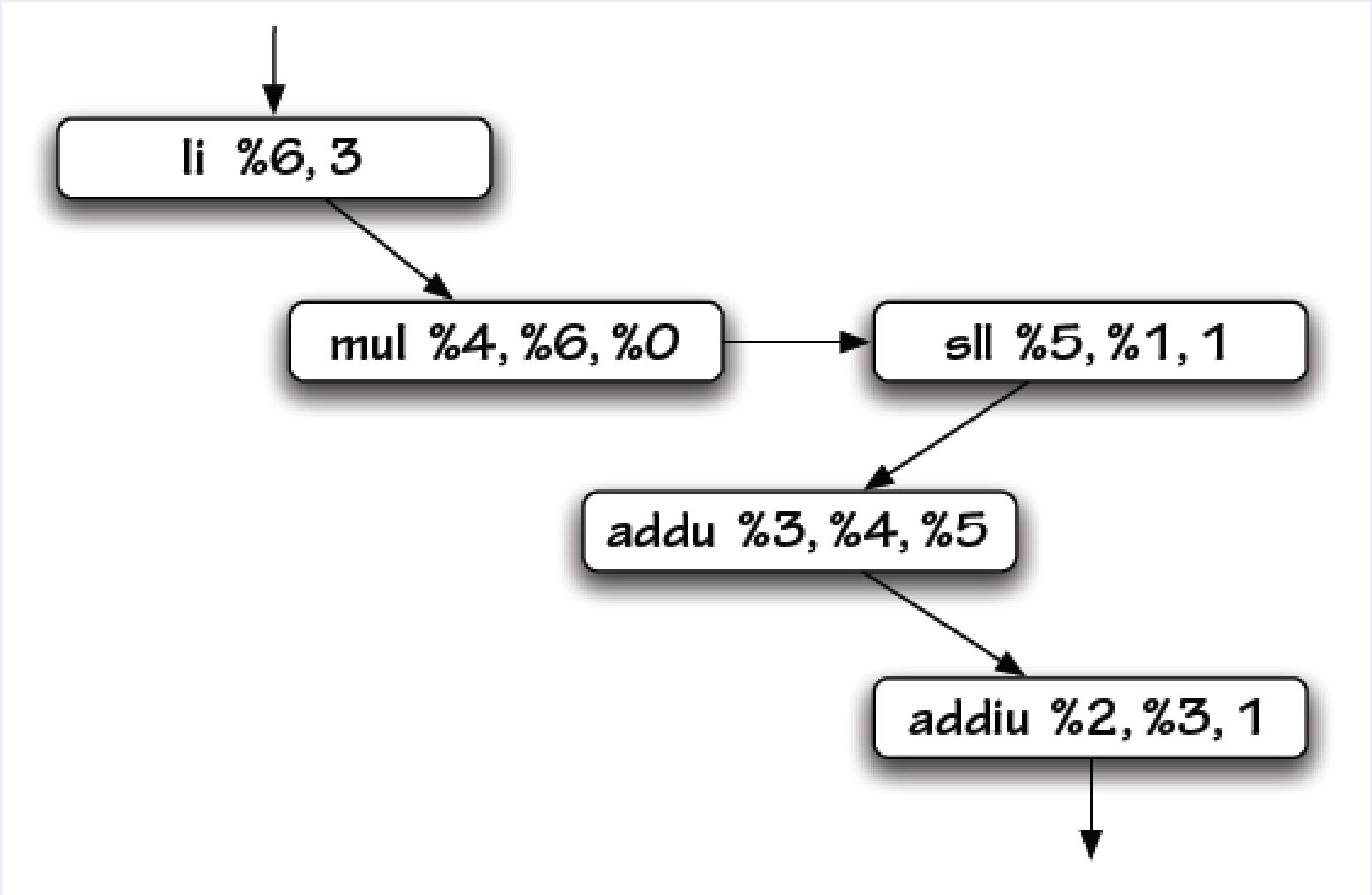
De UPP vers RTL : expressions

Traduction d'une expression arithmétique



De UPP vers RTL : expressions

Traduction en RTL



De UPP vers RTL : expressions

Remarques

- sll : décalage à gauche (multiplication par 2) ;
- Un environnement est nécessaire pour mémoriser le fait que x devient %1, y devient %0, etc. ;
- Un pseudo-registre frais reçoit le résultat de chaque sous-expression ;
- Chaque (sous-)expression est traduite par un fragment de graphe doté d'un label d'entrée, un label de sortie, et un pseudo-registre destination distingués ;
- Les fragments de graphe correspondant aux différentes sous-expressions sont reliés les uns aux autres d'une façon qui reflète l'ordre d'évaluation imposé par la sémantique de PP et UPP.

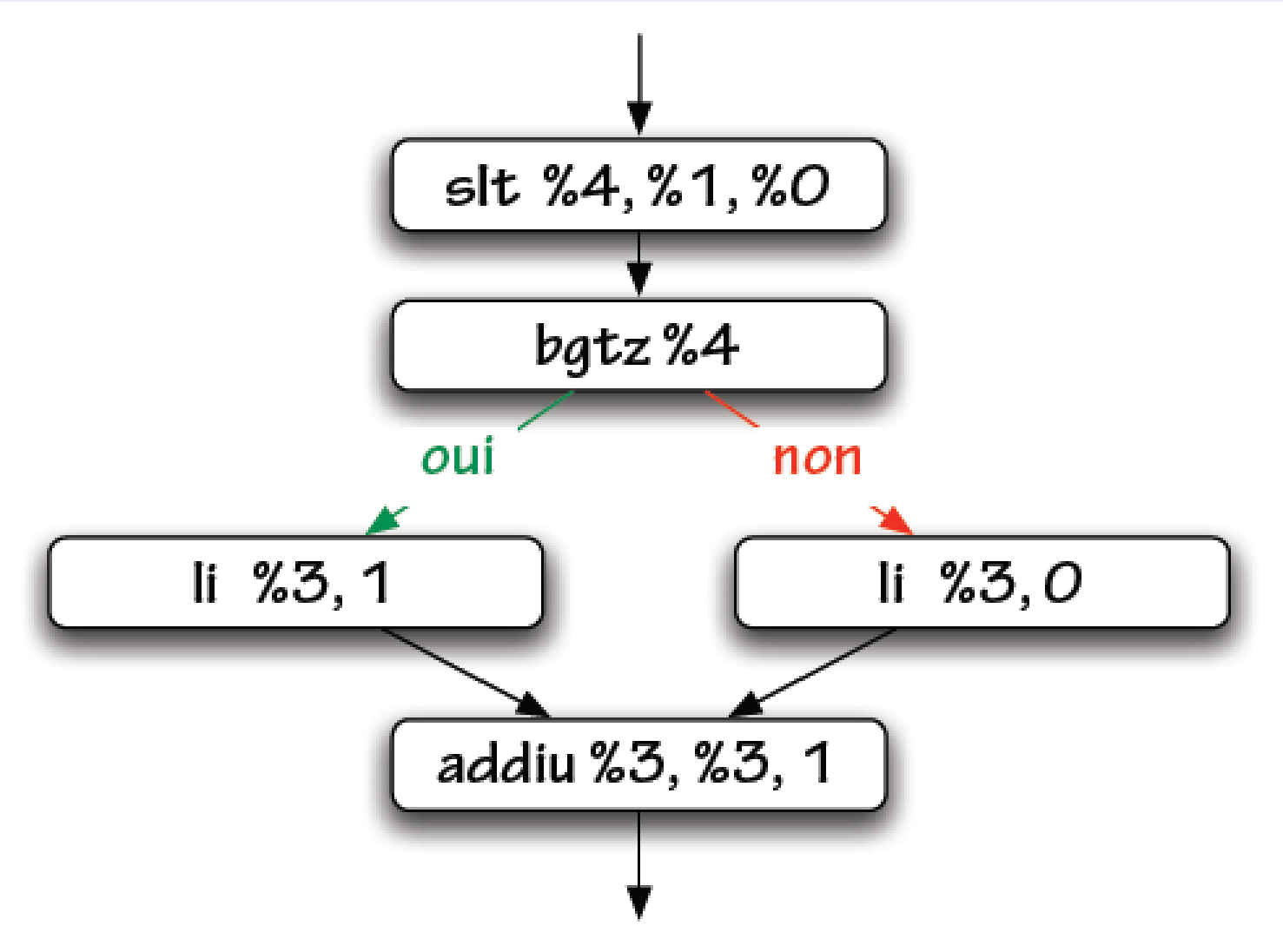
De UPP vers RTL : instructions

Traduction d'une conditionnelle

if $x < y$ then
    $z := 1$
else
    $z := 0$;
$z := z + 1$

De UPP vers RTL : instructions

Traduction en RTL



De UPP vers RTL : instructions

Remarques

- La traduction la plus simple de la conditionnelle consiste à évaluer la condition vers un pseudo-registre, qui contient alors 0 ou 1, puis à utiliser (par exemple) l'instruction bgtz ;
- Les deux branches se rejoignent à l'issue de la conditionnelle : c'est une structure de graphe acyclique et non simplement de liste ;
- Chaque instruction est traduite par un fragment de graphe doté d'un label d'entrée et d'un label de sortie distingués.

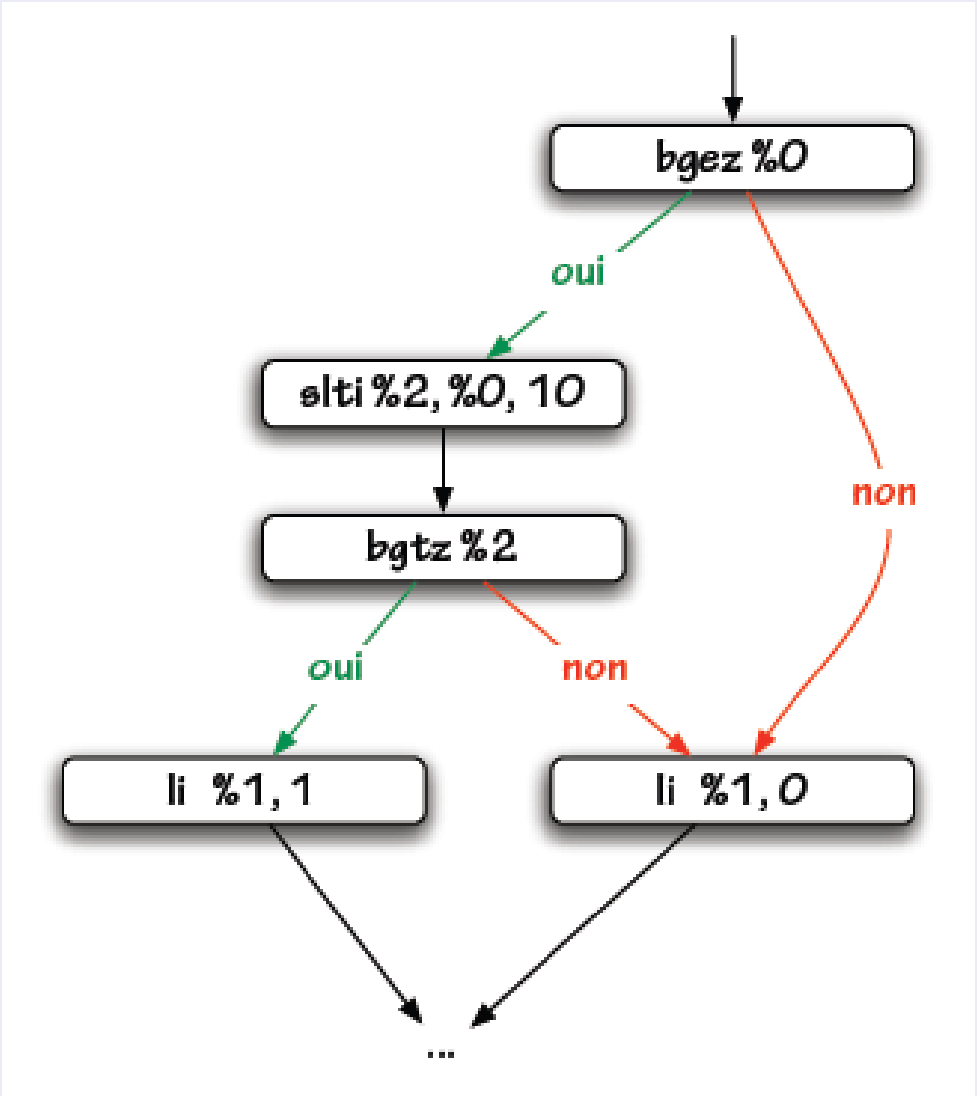
De UPP vers RTL : instructions

Traduction d'une conditionnelle plus complexe

if $x \geq 0$ and $x \leq 9$ then
    chiffre := true
else
    chiffre := false

De UPP vers RTL : instructions

Traduction en RTL



De UPP vers RTL : instructions

Remarques

- Une conditionnelle peut parfois être traduite sans évaluer explicitement la condition : c'est ce que permettent les instructions spécialisées bgez, bgtz, blez, bltz, ble, bne ;
- Si le test $x \geq 0$ échoue, on n'effectue pas le test $x \leq 9$, ce qui reflète le comportement « paresseux » du « and » imposé par la sémantique de notre langage.

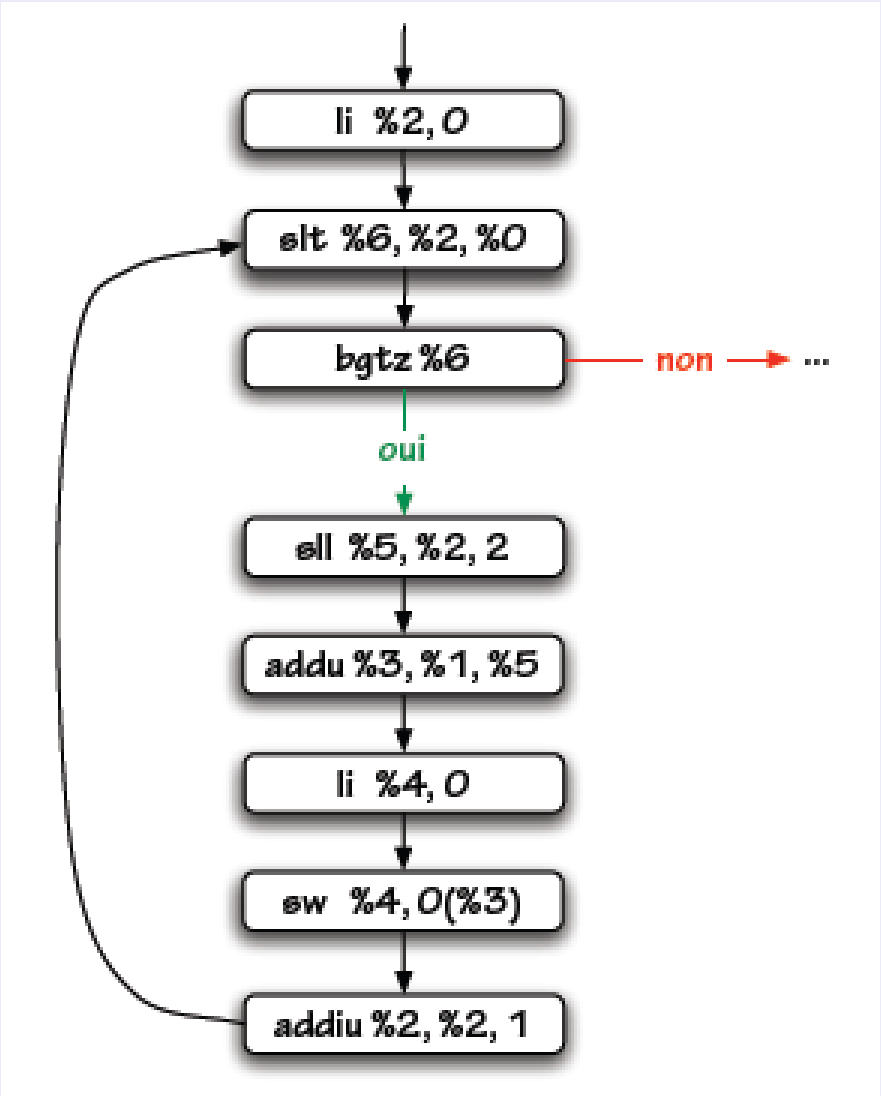
De UPP vers RTL : instructions

Traduction d'une boucle

$i := 0;$
while $i < n$ do
    $t[i] := 0;$
    $i := i + 1$

De UPP vers RTL : instructions

Traduction en RTL



De UPP vers RTL : instructions

Remarques

- Les boucles rendent le graphe cyclique (on s'en doutait) ;
- Toutefois, en l'absence de construction goto dans le langage source, les graphes obtenus restent réductibles : leurs boucles sont imbriquées de façon structurée.

Exercice

Dessiner le graphe de flot de contrôle

```
m := 0;
v := 0;
if v > n then
    skip
else
    r := v;
    s := 0;
    if r < n then
        x := t[r];
        s := s + x;
        if s < m then
            r := r + 1
        else
            skip
    else
        v := v + 1;
m := 1
```

Exercice

Traduire en RTL

while $2 \times b < a$ do
    $a := a - 1;$
$b := 3$

Implantation

Code Java

- Couches UPP et RTL : « UPP.java » et « RTL.java » ;
- Structures de données imposées ;
- Fonctions de transformation à écrire par vos soins ;
- Fonction de transformation « toRTL » dans « UPP.java ».

Organisation

- Écrire la transformation « toRTL » ;
- Éventuellement, un « printer » pour RTL ;
- Prochain rendu : **11 octobre 2017**.