# Analyse de durée de vie et construction du graphe d'interférences

David Delahaye

David.Delahaye@lirmm.fr

Faculté des Sciences

Master M1 2020-2021

# Principe

## Durée de vie et interférence

Cette réflexion est illustrée par les trois points suivants :

- Il faut savoir en quels points du code chaque variable est vivante, c'est-à-dire contient une valeur susceptible d'être utilisée à l'avenir ;
- Deux variables peuvent être réalisées par le même emplacement si elles n'interfèrent pas, c'est-à-dire si l'on n'écrit pas dans l'une tandis que l'autre est vivante ;
- Les instructions « move » suggèrent des emplacements préférentiels.

La phase d'allocation de registres doit donc être précédée d'une analyse de durée de vie, d'où on déduit un graphe d'interférences.

La définition du mot « variable » sera précisée.

# Analyse de durée de vie

## Registres allouables et variables

- On souhaite réaliser chaque pseudo-registre par un registre physique, ou, à défaut, un emplacement de pile ;
- On s'intéresse uniquement aux registres physiques dits allouables, c'est-à-dire non réservés pour un usage particulier, comme le sont par exemple \$gp ou \$sp ;
- Les registres exploités par la convention d'appel, à savoir \$v0-\$v1, \$a0-\$a3, et \$ra sont allouables. Les registres \$t0-\$t9 (« caller-save ») et \$s0-\$s7 (« callee-save ») le sont également ;
- L'analyse de durée de vie concerne les pseudo-registres et les registres physiques allouables : ils sont collectivement appelés « variables » ;
- L'analyse ne concerne pas les emplacements mémoire.

# Analyse de durée de vie

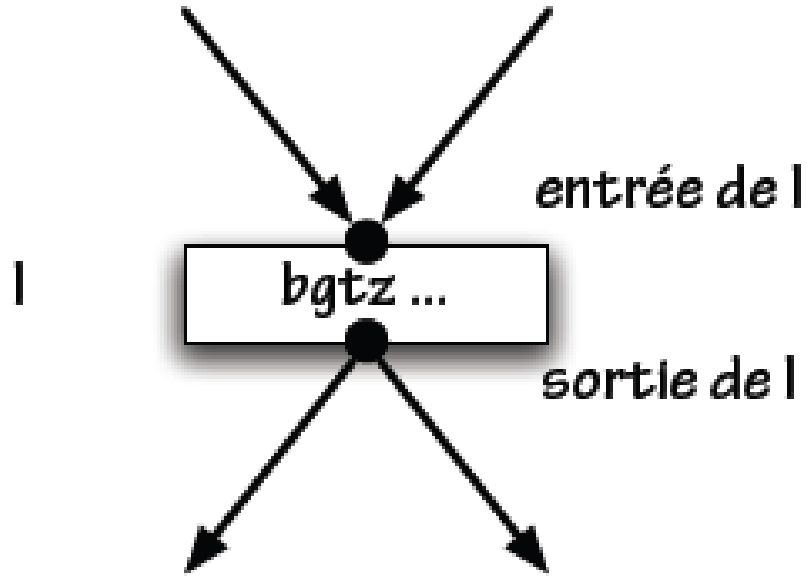
## Variables vivantes, variables mortes

- Une variable $v$ est vivante au point $p$ (d'un programme) s'il existe un chemin menant de $p$ à un point $p'$ où $v$ est utilisée et si $v$ n'est pas définie (affectée) le long de ce chemin ;
- Une variable est morte au point $p$ (d'un programme) si tous les chemins à partir de $p$ mènent à des points $p'_i$ où $v$ est définie (affectée) et si $v$ n'est pas utilisée le long de ces chemins ;
- Une variable peut être ni vivante ni morte (typiquement quand la variable n'est pas utilisée par le programme).

# Analyse de durée de vie

## Point de programme

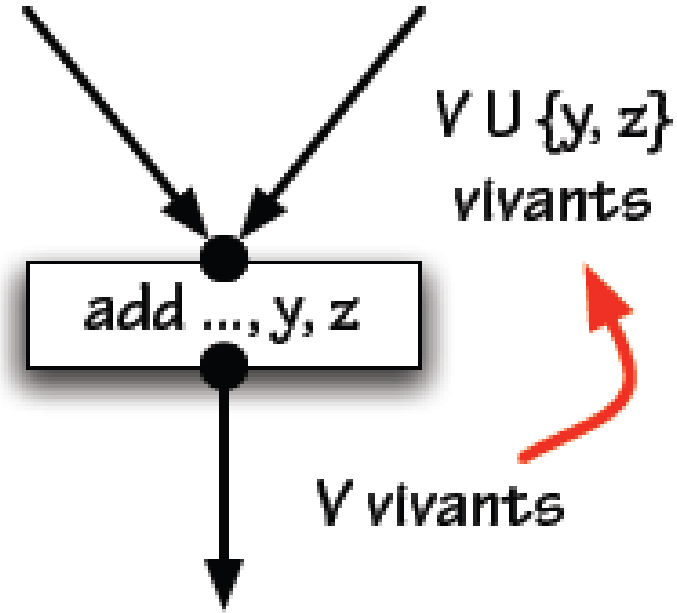
- On va travailler sur le graphe de flot de contrôle (ERTL) ;
- On distinguera les points à l'entrée d'un sommet *l* du graphe de flot de contrôle, et ceux à la sortie d'un sommet *l*.



# Analyse de durée de vie

## Naissance d'une variable

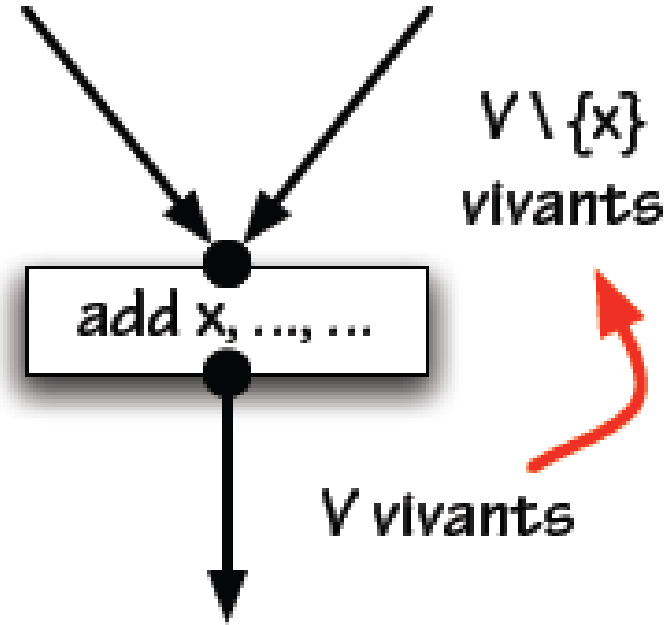
- Une variable $v$ est engendrée par une instruction $i$ si $i$ utilise $v$, c'est-à-dire si $i$ lit une valeur dans $v$ ;
- Dans ce cas, $v$ est vivante au point qui précède immédiatement $i$.



# Analyse de durée de vie

## Mort d'une variable

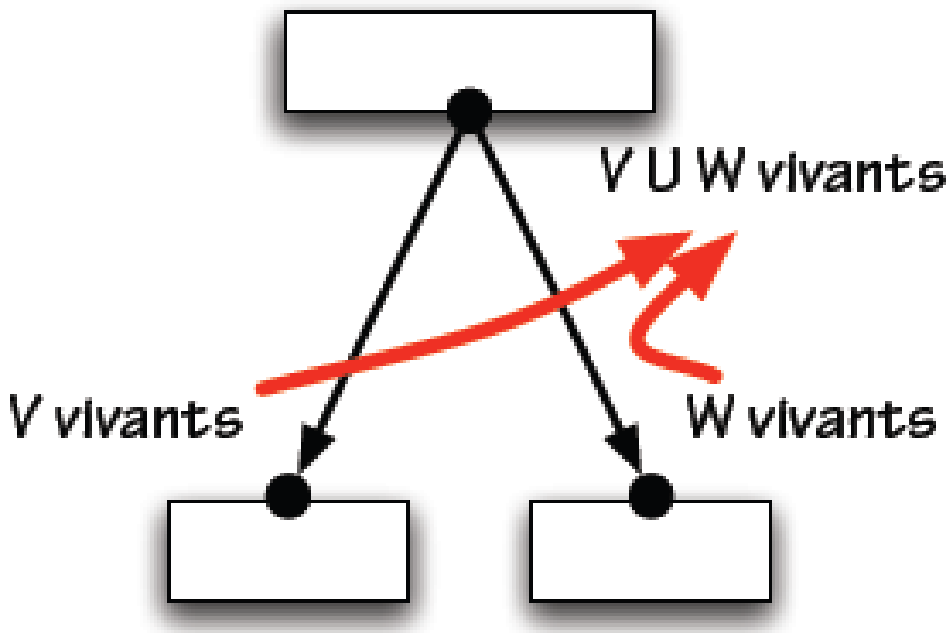
- Une variable $v$ est tuée par une instruction $i$ si $i$ définit $v$, c'est-à-dire si $i$ écrit une valeur dans $v$ ;
- Dans ce cas, $v$ est morte au point qui précède immédiatement $i$.



# Analyse de durée de vie

## Vie d'une variable

- Si $i$ n'engendre ni ne tue $v$, alors $v$ est vivante immédiatement avant $i$ si et seulement si elle est vivante immédiatement après $i$ ;
- Une variable est vivante après $i$ si et seulement si elle est vivante avant l'un quelconque des successeurs de $i$.



# Analyse de durée de vie

## Calcul des variables vivantes aux différents points du programme

- Il faut connaître les ensembles de variables vivantes après les instructions qui sont les feuilles du graphe de flot de contrôle (ce sont les données du problème, on ne peut pas le deviner) ;
- Ensuite, pour calculer les ensembles de variables vivantes aux autres points du programme, on fait une analyse arrière (ce qui nous permettra d'obtenir la plus petite solution) : la vivacité se propage dans le sens inverse des arêtes du graphe de flot de contrôle.

# Analyse de durée de vie

## Exemple

$t := x;$
$x := y;$
$y := t$
$\{x, y\}$

Quels sont les variables vivantes dans les différents points du programme ?

# Analyse de durée de vie

## Exemple

$\{x, y\}$
$t := x;$
$\{y, t\}$
$x := y;$
$\{x, t\}$
$y := t$
$\{x, y\}$

Quels sont les variables vivantes dans les différents points du programme ?

# Analyse de durée de vie

## Exemple

$t := x;$
$x := y;$
$y := t$
$\{\}$

Même question.
Quels sont les variables vivantes dans les différents points du programme ?

# Analyse de durée de vie

## Exemple

$\{x, y\}$
$t := x;$
$\{y, t\}$
$x := y;$
$\{t\}$
$y := t$
$\{\}$

Même question.
Quels sont les variables vivantes dans les différents points du programme ?
Quel est l'ensemble minimal de variables en début de programme ? $\{x, y\}$.

# Analyse de durée de vie

## Autre exemple

$x := x + 1;$
$y := x + 1$
$\{x, y\}$

Quels sont les variables vivantes dans les différents points du programme ?

# Analyse de durée de vie

## Autre exemple

$\{x\}$
$x := x + 1;$
$\{x\}$
$y := x + 1$
$\{x, y\}$

Quels sont les variables vivantes dans les différents points du programme ?

# Exercice

## Soit le programme suivant

$g = j + 12;$
$h = k - 1;$
$f = g \times h;$
$e = j + 8;$
$m = j + 16;$
$b = f;$
$c = e + 8;$
$d = c;$
$k = m + 4;$
$j = b$

Quels sont les variables vivantes à l'entrée du programme sachant qu'à la fin du programme, les variables vivantes sont $d$, $k$, et $j$ ?

# Analyse de durée de vie en ERTL

## Vivacité des registres physiques liés à l'appel de procédures

- À l'entrée d'une procédure, les registres $ra, $a0-$a3, et les « callee-save » $s0-$s7 sont potentiellement vivants (au minimum, il y aura $ra, qu'il soit sauvegardé ou non) ;
- À la sortie d'une procédure, les « callee-save » $s0-$s7, et les registres $v0-$v1 (si on retourne un ou des résultats) sont potentiellement vivants (au minimum, il n'y a rien) ;
- Un appel de procédure tue tous les « caller-save », à savoir $a0-$a3, $v0-$v1, $ra, et $t0-$t9 (on ne sait pas si c'est le cas effectivement pour tous ces registres, il faudrait faire une analyse plus fine de tous les appels, mais au minimum, $ra est tué) ;
- Un retour de procédure (jr $*ra*) n'engendre que $ra.

# Exercice

## Fonction factorielle 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 $\rightarrow f0$

Faire l'analyse de durée des variables de cette procédure (partir d'un ensemble vide de variables vivantes en fin de procédure).

# Graphe d'interférences

## Interférence

- Deux variables distinctes interfèrent si l'une est vivante à la sortie d'une instruction qui définit l'autre ;
- Deux variables qui n'interfèrent pas peuvent être réalisées par un unique emplacement (registre physique ou emplacement de pile) ;
- Inversement, deux variables qui interfèrent doivent être réalisées par deux emplacements distincts.

# Graphe d'interférences

## Exception

- Supposons $y$ vivante à la sortie d'une instruction qui définit $x$ ; si la valeur reçue par $x$ est certainement celle de $y$, alors il n'y a pas lieu de considérer que les deux variables interfèrent ;
- Cette propriété est en général indécidable, mais il en existe un cas particulier simple, celui d'une instruction move $x, y$ ;
- Ce cas particulier est important car, dans ce cas, on souhaite justement que $x$ et $y$ soient réalisés par le même emplacement, de façon à supprimer l'instruction move.

# Graphe d'interférences

## Graphe d'interférences

- On construit un graphe dont les sommets sont les variables et dont les arêtes représentent les relations d'interférence et de préférence ;
- On crée une arête d'interférence entre deux variables qui interfèrent ;
- On crée une arête de préférence entre deux variables reliées par une instruction move ;
- Ce graphe permet de spécifier à l'allocateur de registres les contraintes sous lesquelles il doit travailler.

# Graphe d'interférences

## Exemple

$\{x, t, z\}$
$v := 0;$
$\{x, t, z\}$
$y := z + t;$
$\{x, y, t\}$
$u := t;$
$\{x, y\}$
$z := x + y;$
$\{z\}$
$v := z$
$\{\}$

Interférences : $(v, x), (v, t), (v, z), (y, x), (y, t), (u, x), (u, y)$.
Préférences : $(u, t)$.

# Exercice

## Soit le programme suivant

$g = j + 12;$
$h = k - 1;$
$f = g \times h;$
$e = j + 8;$
$m = j + 16;$
$b = f;$
$c = e + 8;$
$d = c;$
$k = m + 4;$
$j = b$

Dessiner le graphe d'interférences de ce programme sachant qu'à la fin du programme, les variables vivantes sont $d$, $k$, et $j$ ?