Hugues Cassé <casse@irit.fr>
M1 SIAME - FSI - Université de Toulouse

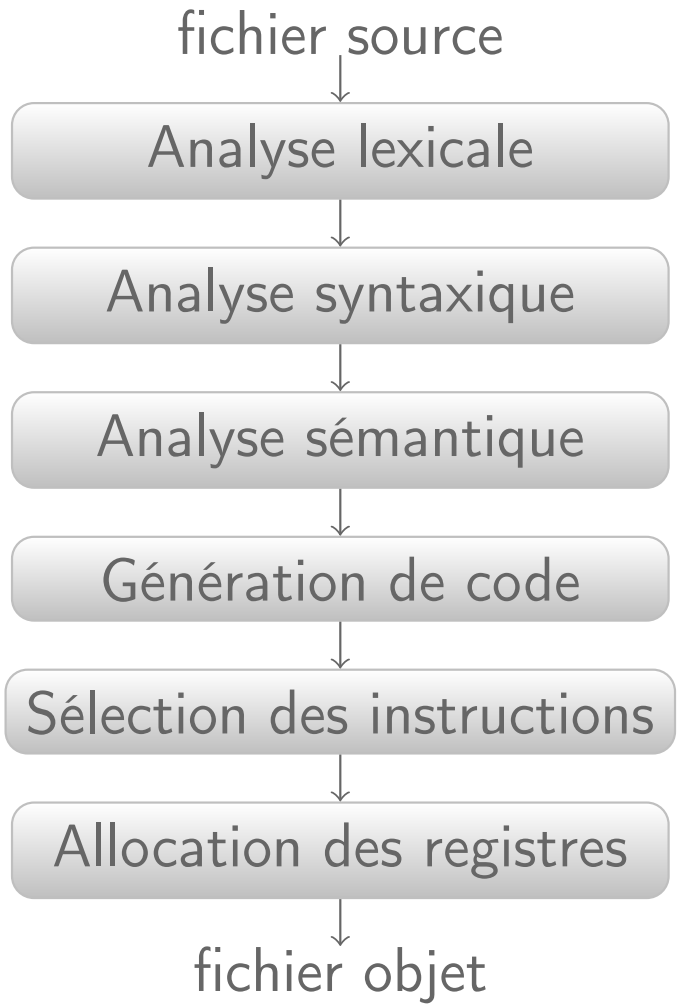
Cours 3 : Traduction en quadruplets

COMC

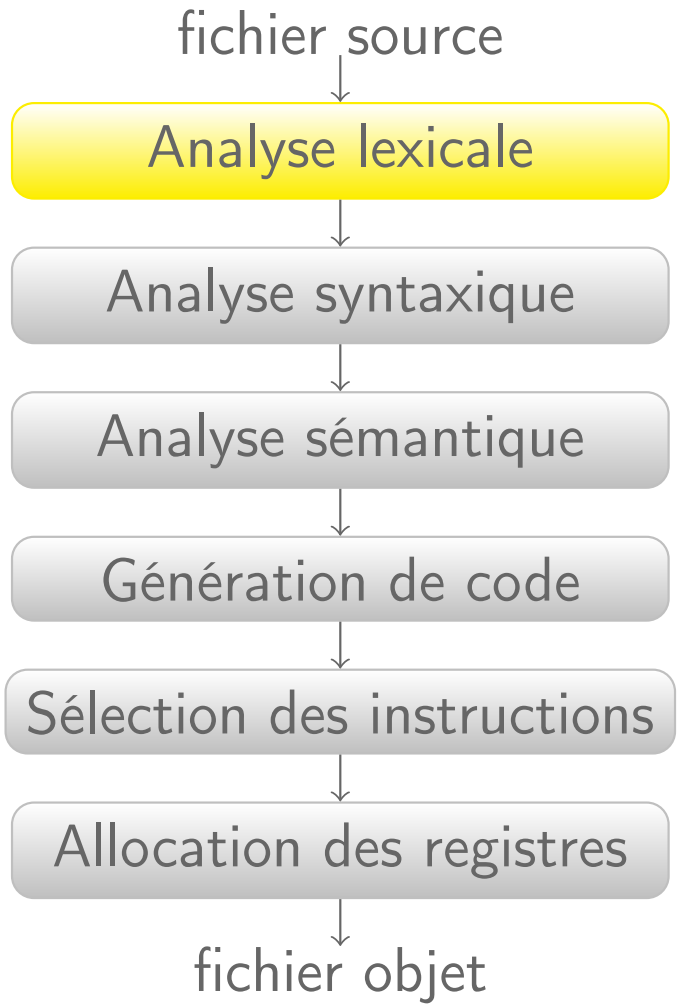
Plan

1. Introduction
2. Les quadruplets
3. Traduction des expressions
4. Traduction des instructions
5. Traduction des fonctions et variables globales
6. Conclusion

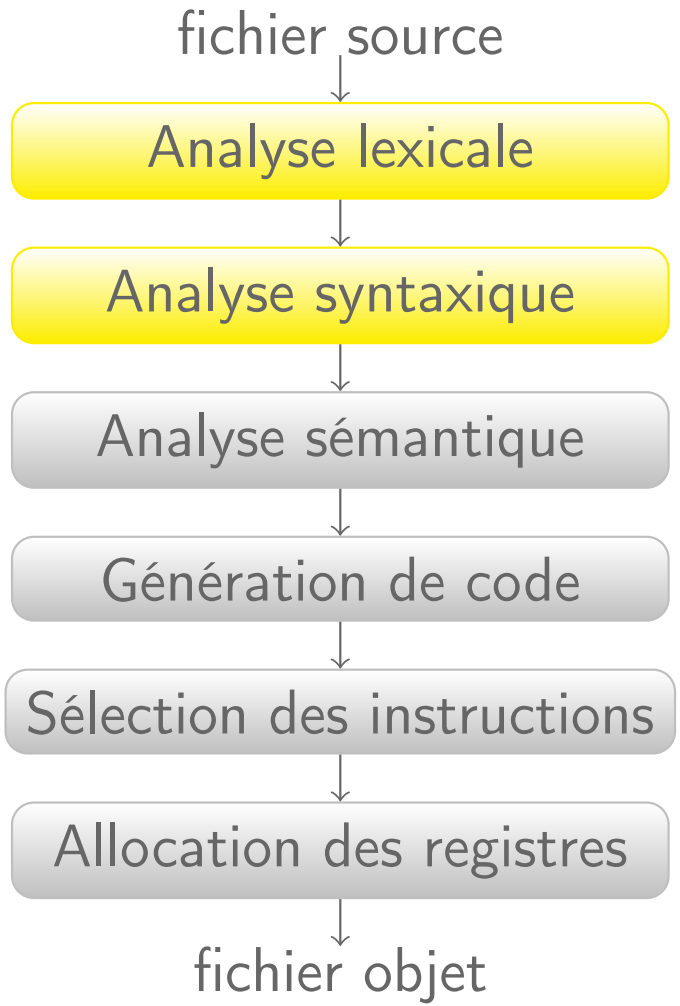
La chaîne de compilation



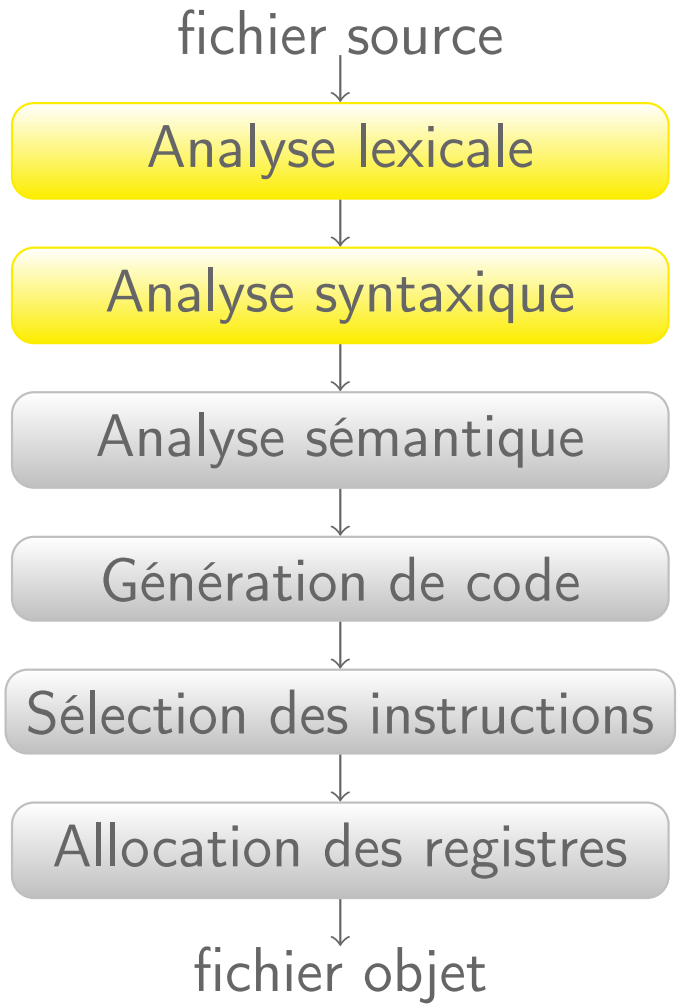
La chaîne de compilation



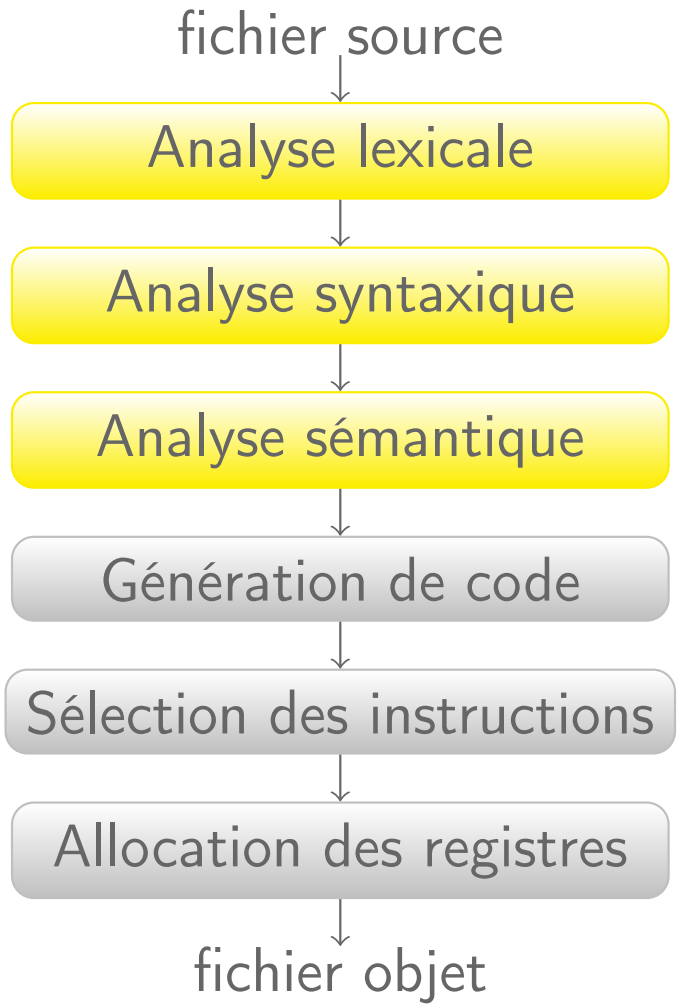
La chaîne de compilation



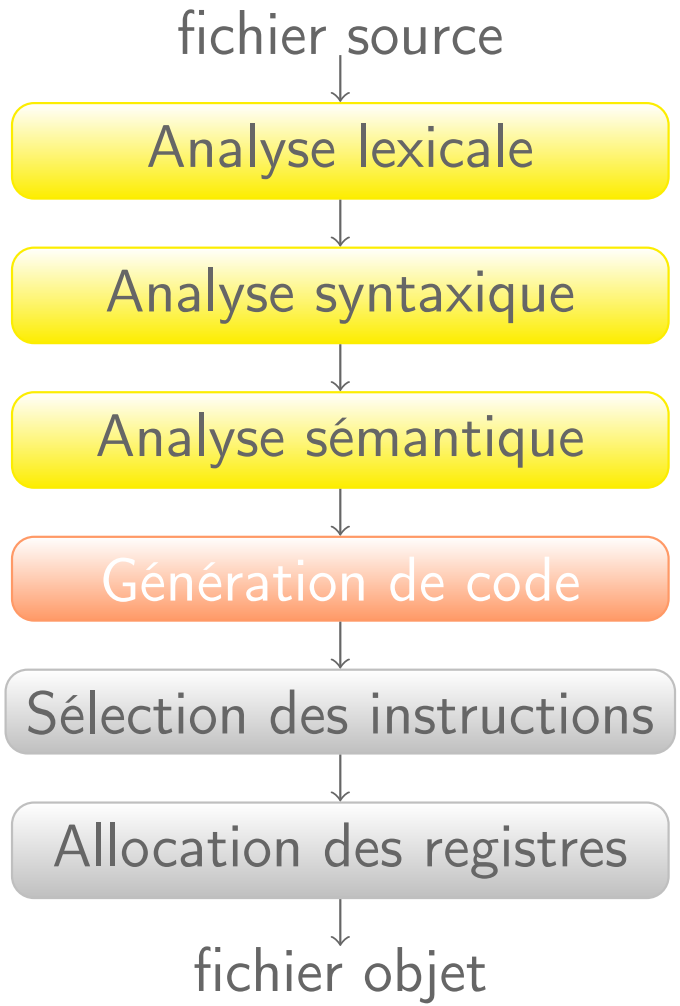
La chaîne de compilation



La chaîne de compilation



La chaîne de compilation



Objectif : traduire

- AST– proche du langage de haut niveau (C)
- quadruplets – proche du langage de base niveau (assembleur)

Contrainte : conserver la sémantique !

Techniquement...

Traducteurs :

- $T_s : S \times \Gamma \mapsto Q^*$
- $T_e : E \times \Gamma \mapsto \mathcal{V} \times Q^*$
- $T_c : E \times \mathcal{L} \times \mathcal{L} \times \Gamma \mapsto Q^*$ – condition

Avec :

- Q – quadruplet
- $\Gamma : ID \rightarrow L$ – environnement (identificateur, position en mémoire)
- L – localisation dans la mémoire
- $\mathcal{L}$– étiquette dans le code

Modèle d'exécution

- microprocesseur = machine à état
  - $\mathcal{S}$ – état de la machine (mémoire + registres)
  - $\mathcal{I}$ – ensemble des instructions
  - $\mathbb{I} : \mathcal{I} \times \mathcal{S} \mapsto \mathcal{S}$ – fonction d'exécution
- état $\mathcal{S} : \mathcal{A} \cup \mathcal{R} \mapsto \mathbb{V}$
  - $\mathbb{V} = \mathbb{Z}_n \cup \mathbb{F}_n$ – mot machines sur n bits
  - $\mathcal{A}$ – ensemble des addresses ($\mathbb{N}_m$ bits)
  - $\mathcal{R}$ – ensemble des registres $\{r_0, r_1, ..., r_{13}, LR, PC, SR\}$

Interprétation des instructions (ARM)

Sémantique opérationnelle :

- opérations sur les registres
  $\mathbb{I}[add\ r_i, r_j, r_k]\ s = s[\ r_i \rightarrow s[r_j] + s[r_k]\ ]$
- opérations spécialisées pour la mémoire
  $\mathbb{I}[ldr\ r_i, [r_j]]\ s = s[\ r_i \rightarrow s[s[r_j]]\ ]$
  $\mathbb{I}[str\ r_i, [r_j]]\ s = s[\ s[r_j] \rightarrow s[r_i]\ ]$
- comparaison
  $\mathbb{I}[cmp\ r_i, r_j]\ s = s[SR \rightarrow s[r_i] \sim s[r_j]\ ]$

Avec $x \sim y$ résultat de la comparaison (EQ, NE, LT, ...).

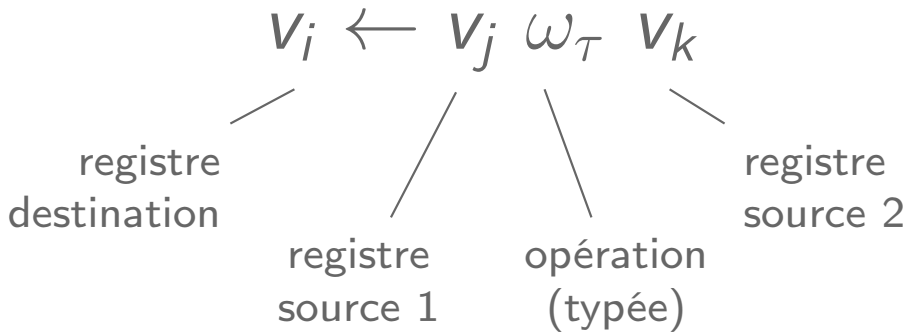
Gestion du flot de contrôle

- exécution du programme avec continuation
  $\mathbb{I}_c\ s = \mathbb{I}[i]s[PC \rightarrow s[PC] + |i|]$ avec $i = s[s[PC]]$
- exécution complète du programme
  $\mathbb{I}^*\ s_0 = s_f = \mathbb{I}_c\ \mathbb{I}_c\ \mathbb{I}_c ...\ \mathbb{I}s_0$ avec $s_f[s_f[PC]] = stop$
- réalisation des branchements
  - inconditionnel
    $\mathbb{I}[b\ a]\ s = s[PC \rightarrow a]$
  - conditionnel
    $$\mathbb{I}[beq\ a]\ s = \begin{cases} s[PC \rightarrow a] & si\ EQ \in s[SR] \\ s & sinon \end{cases}$$
- réalisation de sous-programme
  $\mathbb{I}[bl\ a]\ s = s\ [LR \rightarrow s[PC] + |i|]\ [PC \rightarrow a]$

Plan

1. Introduction
2. Les quadruplets
3. Traduction des expressions
4. Traduction des instructions
5. Traduction des fonctions et variables globales
6. Conclusion

Un quadruplet



- $v_i, v_j, v_k \in \mathcal{V}$ – ensemble infini de registres virtuels
- $\omega \in \Omega_2$ (noté $+$, $-$, ...)
- $\tau \in T$ – opération typée (souvent omise, INT sous-entendu)

Quadruplets irréguliers

- opérateur unaire – $v_i \leftarrow \omega_1\ v_j$ avec $\omega_1 \in \Omega_1$
- lecture mémoire – $v_i \leftarrow M_\tau\,[v_j]$
- écriture en mémoire – $M_\tau\,[v_i] \leftarrow v_j$
- affectation de constante – $v_i \leftarrow k,\ k \in \mathbb{N}_n \cup \mathbb{F}_m$
- étiquettes (pseudo-quadruplet) – *label* l, $l \in \mathcal{L}$
- branchement inconditionnel – *goto* l ou *goto* v_i
- branchement conditionnel – *if* $v_i\ \omega_\tau\ v_j$ *goto* l
  avec $\omega_\tau \in \{=, \neq, <, \leq, >, >, \geq\}$
- conversion – $v_i \leftarrow cast(\tau, v_j)$
- appel de sous-programme – $v_r \leftarrow call\ l(v_i, v_j, ...)$
- retour de sous-programme – *return*

Gestion des variables

Localisation L :

- globale – adresse absolue (ou relogeable)
- locale – relative au pointeur de bloc d'activation d'un sous-programme *FP* (*frame pointer*)

$$L : \mathit{noloc} \mid \mathit{global}(\mathbb{Z}) \mid \mathit{local}(\mathbb{Z})$$

Environnement de compilation Γ :

$$\Gamma : \mathit{ID} \mapsto L$$

$$\gamma_\perp \in \Gamma, \gamma_\perp = \lambda x \ . \ \mathit{noloc}$$

$$\gamma_{\mathit{init}} = \lambda x \ . \ l \ \mathit{if} \ (x, l) \in \mathit{Global} \cup \mathit{Param}, \mathit{noloc} \ \mathit{else}$$

Exemple (1)

```
void memcpy(void *p, void *q,
  int s);
char t[256];

void f(int s) {
  char tp[256];
  if(s != 0)
    memcpy(tp, t, s);
  ...
}
```

$$\gamma_{init} = \{"t" \rightarrow global(k_t),$$
$$"s" \rightarrow local(k_s),$$
$$"tp" \rightarrow local(k_{tp})\}$$

$label\ f:$

$v_0 \leftarrow k_s$

$v_1 \leftarrow SP + v_0$

$c_2 \leftarrow 0$

$if\ v_0 = v_2\ goto\ l_0$

$v_3 \leftarrow k_{tp}$

$v_4 \leftarrow SP + v_3$

$v_6 \leftarrow k_t$

$v_7 \leftarrow v_6$

$call\ memcpy(v_4, v_7, v_0)$

$label\ l_0:$

$...$

Exemple (2)

```
char t[256];

void f() {
  char s;
  int i;
  ...
  s = s + t[i];
  ...

}
```

$$\gamma_{init} = \{"t" \rightarrow global(k_t),$$
$$"s" \rightarrow local(k_s),$$
$$"i" \rightarrow local(k_i)\}$$

$label\ f$:

$$\ldots$$
$$v_1 \leftarrow k_s$$
$$v_2 \leftarrow SP + v_1$$
$$v_3 \leftarrow M_{char}[v_2]$$
$$v_4 \leftarrow k_t$$
$$v_5 \leftarrow k_i$$
$$v_6 \leftarrow SP + v_5$$
$$v_7 \leftarrow M_{int}[v_6]$$
$$v_8 \leftarrow v_4 + v_7$$
$$v_9 \leftarrow M_{char}[v_8]$$
$$v_{10} \leftarrow k_s$$
$$v_{11} \leftarrow SP + v_{10}$$
$$M_{char}[v_{11}] \leftarrow v_9$$
$$\ldots$$

Travailler avec les listes de quadruplets

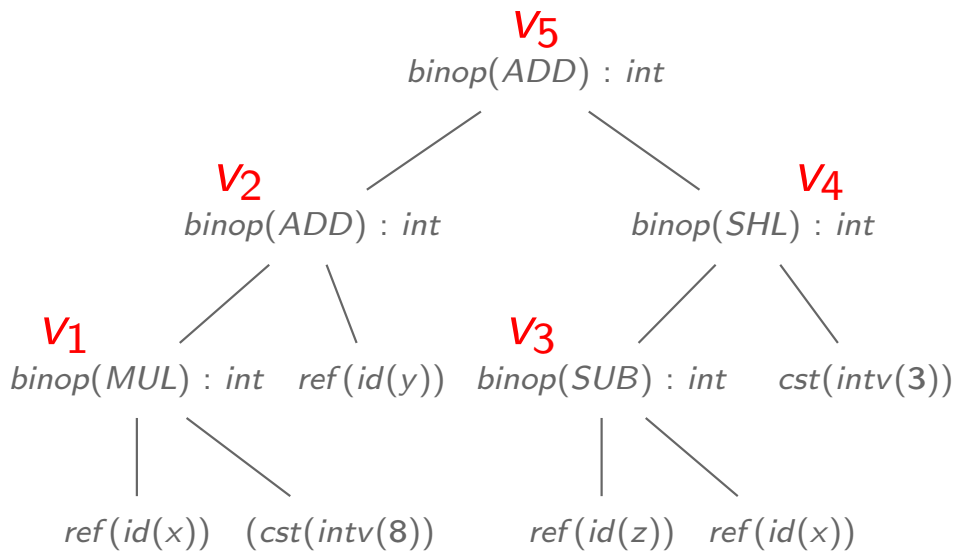
- $(v_1 \leftarrow v_2 \times v_3) \in Quad$ – quadruplet
- $[v_1 \leftarrow v_2 \times v_3, v_4 \leftarrow v_5 + v_6] \in Q^*$ – liste de quadruplets
- $\forall q \in Q, qs \in Q^*, q :: qs$ – construction d'une liste avec ajout en tête
- $\forall q_1, q_2 \in Q^*, q_1 \mathbin{@} q_2$ – construction d'une liste par concaténation
- $\forall qs \in Q^*, let\ (q, t) = qs$ – obtention de la tête et de la queue d'une liste de quadruplet

Plan

1. Introduction
2. Les quadruplets
3. Traduction des expressions
4. Traduction des instructions
5. Traduction des fonctions et variables globales
6. Conclusion

Ordre d'évaluation

```
int x, y, z;
x * 8 + y + ((z - x) >> 3);
```



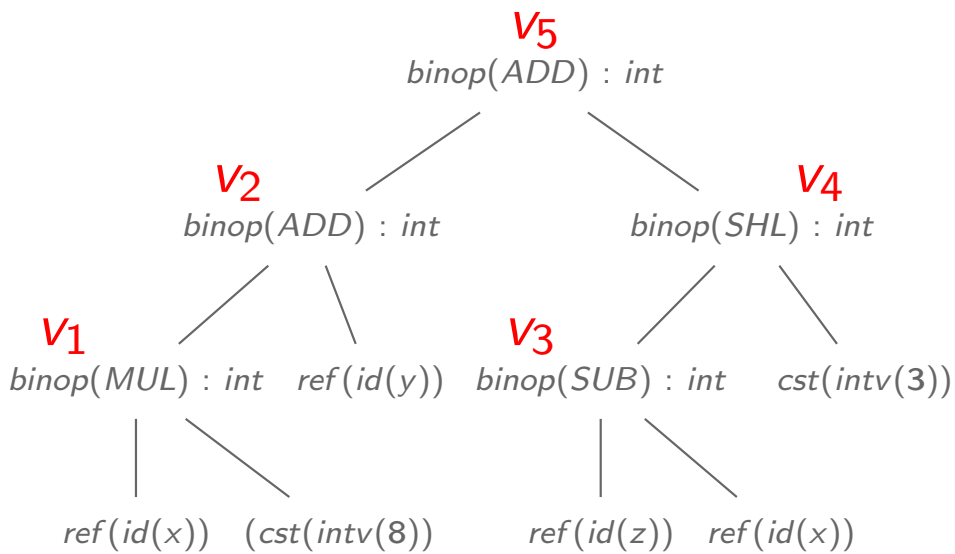
Ordre d'évaluation :

1. $x * 8 \rightarrow v_1$
2. $v_1 + y \rightarrow v_2$
3. $z - x \rightarrow v_3$
4. $v_3 >> 3 \rightarrow v_4$
5. $v_2 + v_4 \rightarrow v_5$

- $\Leftrightarrow$ parcours en profondeur d'abord !
- résultats intermédiaires $\rightarrow$ registres !

Traduction des expressions

```
int x, y, z;
x * 8 + y + ((z - x) >> 3);
```



générateur de registre virtuel :

$$new\_temp : T \mapsto \mathcal{V}$$

traduction des expressions :

$$T_e : E \times \Gamma \mapsto \mathcal{V} \times Q^*$$

$$\begin{aligned} &T_e[\![binop(\omega, e_1, e_2) : \tau]\!]\ \gamma = \\ &\quad let\ v_1, q_1 = T_e[\![e_1]\!]\ \gamma \\ &\quad let\ v_2, q_2 = T_e[\![e_2]\!]\ \gamma \\ &\quad let\ v = new\_temp\ \tau \\ &\quad (v,\ q_1 @ q_2 @ [v \leftarrow v_1\ \omega_\tau\ v_2]) \end{aligned}$$

avec $\tau \in \{int, float\}$.

Autres expressions

$T_e[\![cst(intv(n))]\!]\ \gamma =$
$\quad let\ v = new\_temp\ int$
$\quad (v,\ [v \leftarrow n])$
$T_e[\![cst(floatv(x))]\!]\ \gamma =$
$\quad let\ v = new\_temp\ float$
$\quad (v,\ [v \leftarrow x])$

$T_e[\![ref(r) : \tau]\!]\ \gamma =$
$\quad let\ v_r, q_r = T_r[\![r]\!]\ \gamma$
$\quad let\ v = new\_temp\ \tau$
$\quad (v, q_r @ [v \leftarrow M_\tau\ [v_r]])$

Traduction des références :

$$T_r : R \times \Gamma \mapsto \mathcal{V} \times Q^*$$

Produit l'adresse de la référence dans le registre résultat !

$T_r[\![id(i) : \tau]\!]\ \gamma =$
$\quad let\ v = new\_temp\ ptr(\tau)$

$$\begin{cases} (v, [v \leftarrow SP + k]) & if\ \gamma[i] = local(k) \\ (v, [v \leftarrow k]) & if\ \gamma[i] = global(k) \end{cases}$$

Exercice 1

1. Donner la traduction des opérations unaires.
2. Donner la traduction de l'expression $addr(r)$.
3. Donner la traduction de la référence $at(e)$.

Exercice 1

1. Donner la traduction des opérations unaires.
2. Donner la traduction de l'expression $addr(r)$.
3. Donner la traduction de la référence $at(e)$.

$$
\begin{aligned}
&T_e[\![unop(\omega, e) : \tau]\!]\ \gamma = \\
&\quad let\ v', q' = T_e[\![e]\!]\ \gamma \\
&\quad let\ v = new\_temp\ \tau \\
&\quad (v, q' @ [v \leftarrow \omega\ v'])
\end{aligned}
$$

Exercice 1

1. Donner la traduction des opérations unaires.
2. Donner la traduction de l'expression $addr(r)$.
3. Donner la traduction de la référence $at(e)$.

$$
\begin{aligned}
&T_e[\![unop(\omega, e) : \tau]\!]\ \gamma = \\
&\quad let\ v', q' = T_e[\![e]\!]\ \gamma \\
&\quad let\ v = new\_temp\ \tau \\
&\quad (v, q' @ [v \leftarrow \omega\ v'])
\end{aligned}
$$

$$
\begin{aligned}
&T_r[\![addr(r : \tau) : ptr(\tau)]\!]\ \gamma = \\
&\quad T_r[\![r]\!]\ \gamma
\end{aligned}
$$

Exercice 1

1. Donner la traduction des opérations unaires.
2. Donner la traduction de l'expression $addr(r)$.
3. Donner la traduction de la référence $at(e)$.

$$T_e[\![unop(\omega, e) : \tau]\!]\ \gamma =$$
$$\quad let\ v', q' = T_e[\![e]\!]\ \gamma$$
$$\quad let\ v = new\_temp\ \tau$$
$$\quad (v, q'@[v \leftarrow \omega\ v'])$$

$$T_r[\![addr(r : \tau) : ptr(\tau)]\!]\ \gamma =$$
$$\quad T_r[\![r]\!]\ \gamma$$

$$T_r[\![at(e : ptr(\tau)) : \tau]\!]\ \gamma =$$
$$\quad T_e[\![e]\!]\ \gamma$$

Exercice 2

1. Donner la traduction des opérations binaires (addition, soustraction) entre pointeurs et entiers.
2. Donner la traduction des opérations binaires entre pointeurs.

Exercice 2

1. Donner la traduction des opérations binaires (addition, soustraction) entre pointeurs et entiers.
2. Donner la traduction des opérations binaires entre pointeurs.

$$
\begin{aligned}
&T_e[\![binop(\omega, e_1 : ptr(\tau), e_2 : int) : ptr(\tau)]\!]\ \gamma = \\
&\quad let\ (v_1, q_1) = T_e[\![e_1]\!]\ \gamma \\
&\quad let\ (v_2, q_2) = T_e[\![e_2]\!]\ \gamma \\
&\quad let\ v, v', v'' = new\_tmp\ ptr(\tau), int, int \\
&\quad (v,\ q_1\ @\ q_2\ @ \\
&\quad [v' \leftarrow |\tau|, v'' \leftarrow v_2 \times v', v \leftarrow v_1 + v''])
\end{aligned}
$$

avec $\omega \in \{ADD, SUB\}$

Exercice 2

1. Donner la traduction des opérations binaires (addition, soustraction) entre pointeurs et entiers.
2. Donner la traduction des opérations binaires entre pointeurs.

$$
\begin{aligned}
&T_e[\![binop(\omega, e_1 : ptr(\tau), e_2 : int) : ptr(\tau)]\!]\ \gamma = \\
&\quad let\ (v_1, q_1) = T_e[\![e_1]\!]\ \gamma \\
&\quad let\ (v_2, q_2) = T_e[\![e_2]\!]\ \gamma \\
&\quad let\ v, v', v'' = new\_tmp\ ptr(\tau), int, int \\
&\quad (v,\ q_1\ @\ q_2\ @ \\
&\quad [v' \leftarrow |\tau|, v'' \leftarrow v_2 \times v', v \leftarrow v_1 + v''])
\end{aligned}
$$

avec $\omega \in \{ADD, SUB\}$

$$
\begin{aligned}
&T_e[\![binop(SUB, e_1 : ptr(\tau), e_2 : ptr(\tau)) : int]\!] \\
&\gamma = \\
&\quad let\ (v_1, q_1) = T_e[\![e_1]\!]\ \gamma \\
&\quad let\ (v_2, q_2) = T_e[\![e_2]\!]\ \gamma \\
&\quad let\ v, v', v'' = new\_tmp\ ptr(\tau), int, int \\
&\quad (v,\ q_1\ @\ q_2\ @ \\
&\quad [v' \leftarrow v_1 - v_2, v'' \leftarrow |\tau|, v \leftarrow v'/v''])
\end{aligned}
$$

Exerice 3

On étend les types avec les tableaux (type des éléments, nombre d'éléments) :

$$T : ... \mid \mathit{array}(T, \mathbb{N})$$

On ajoute une référence sur un élément de tableau :

$R : ... \mid \mathit{itemat}(e_1 : \mathit{array}(\tau, n), e_2 : \mathit{int})$

1. Proposez une organisation en mémoire du tableau et déduisez une formule pour obtenir l'adresse d'un élément `t[i]`.
2. Proposez la traduction de $\mathit{itemat}(t, i)$.

Exerice 3

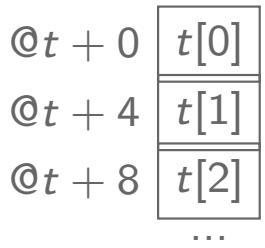
On étend les types avec les tableaux (type des éléments, nombre d'éléments) :

$$T : \ldots \mid \mathit{array}(T, \mathbb{N})$$

On ajoute une référence sur un élément de tableau :

$$R : \ldots \mid \mathit{itemat}(e_1 : \mathit{array}(\tau, n), e_2 : \mathit{int})$$

1. Proposez une organisation en mémoire du tableau et déduisez une formule pour obtenir l'adresse d'un élément t[i].
2. Proposez la traduction de $\mathit{itemat}(t, i)$.



$$\&t[i] = @t + i \times |\tau|$$

Exerice 3

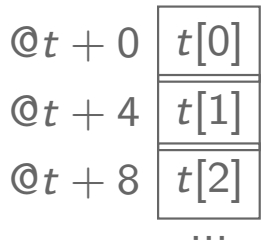
On étend les types avec les tableaux (type des éléments, nombre d'éléments) :

$$T : ... \mid array(T, \mathbb{N})$$

On ajoute une référence sur un élément de tableau :

$$R : ... \mid itemat(e_1 : array(\tau, n), e_2 : int)$$

1. Proposez une organisation en mémoire du tableau et déduisez une formule pour obtenir l'adresse d'un élément t[i].
2. Proposez la traduction de $itemat(t, i)$.



$$\&t[i] = @t + i \times |\tau|$$

$$\begin{aligned}
&T_r[\![itemat(t : array(\tau, n), i : int) : \tau]\!]\ \gamma = \\
&\quad let\ (v_t, q_t) = T_r[\![t]\!]\ \gamma \\
&\quad let\ (v_i, q_i) = T_r[\![i]\!]\ \gamma \\
&\quad let\ v, v', v'' = new_tmp\ ptr(\tau), int, int \\
&\quad (v, q_t\ @\ q_i\ @ \\
&\qquad [v' \leftarrow |\tau|, v'' \leftarrow v_i \times v', v \leftarrow v_t + v''])
\end{aligned}$$

Plan

1. Introduction
2. Les quadruplets
3. Traduction des expressions
4. Traduction des instructions
5. Traduction des fonctions et variables globales
6. Conclusion

Instructions simples

Traduction des instructions :

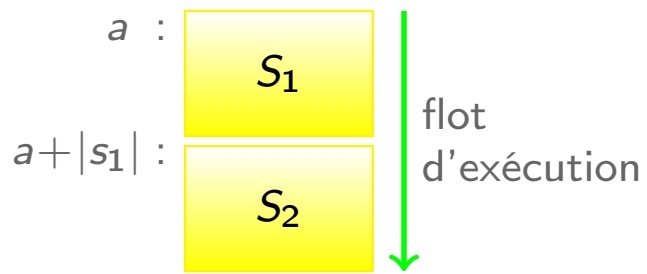
$$T_s : S \times \Gamma \to Q^*$$

Affectation : $set(r : \tau, e : \tau)$

$$T_s[\![set(r : \tau, e : \tau)]\!]\ \gamma =$$
$$let\ (v_r, q_r) = T_r[\![r]\!]\ \gamma$$
$$let\ (v_e, q_e) = T_e[\![e]\!]\ \gamma$$
$$q_r\ @\ q_e\ @\ [M_\tau\ [v_r] \leftarrow v_e]$$

Séquence : $seq(s_1, s_2)$

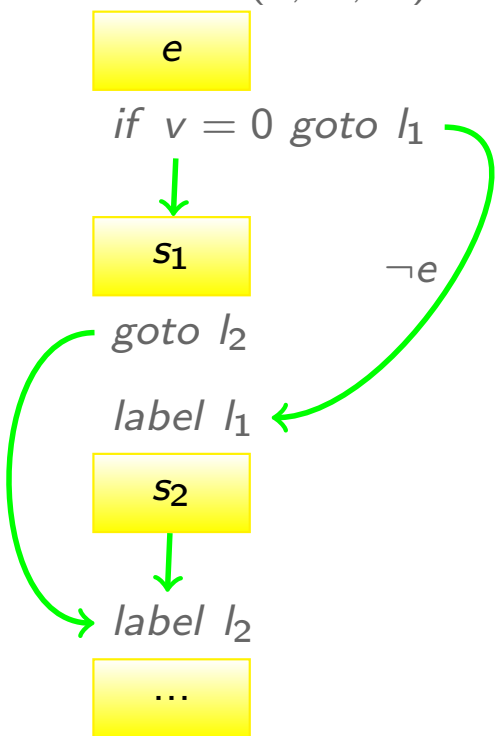
- nativement supportée par la machine
- instruction séquentiellement placées en mémoire



$$T_s[\![seq(s_1, s_2]\!]\ \gamma =$$
$$(T_s[\![s_1]\!]\ \gamma)\ @\ (T_s[\![s_2]\!]\ \gamma)$$

Instructions avec condition

Sélection : $if\,(e, s_1, s_2)$



Génération de label :

$$new\_lab : \mathbb{N} \mapsto \mathcal{L}$$

Génération de la condition e :

$$
\begin{aligned}
&T_e[\![binop(EQ, e_1, e_2)]\!]\ \gamma = \\
&\quad let\ (v_1, q_1) = T_e[\![e_1]\!]\ \gamma \\
&\quad let\ (v_2, q_2) = T_e[\![e_2]\!]\ \gamma \\
&\quad let\ v = new\_temp\ int \\
&\quad let\ l_1, l_2 = new\_lab2 \\
&\quad (v, q_1\ @\ q_2\ @\ [\\
&\qquad if\ v_1 \neq v_2\ goto\ l_1, \\
&\qquad v \leftarrow 1, \\
&\qquad goto\ l_2, \\
&\qquad label\ l_1, \\
&\qquad v \leftarrow 0, \\
&\qquad label\ l_2 \\
&\quad]
\end{aligned}
$$

Traduction des conditions

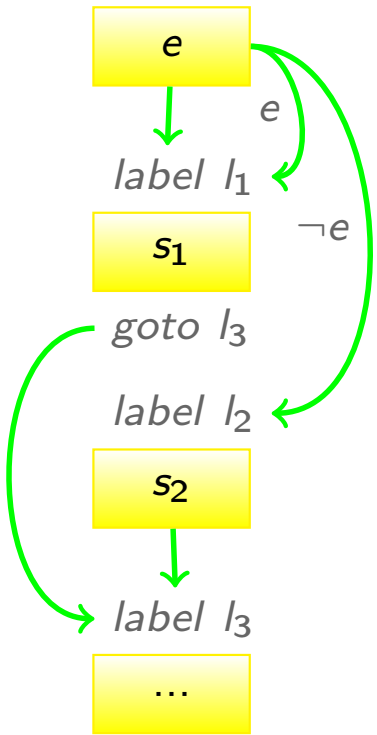
Fonction spécialisées pour la traduction des expressions en condition :

$$T_c : E \times (l_\top : \mathcal{L}) \times (l_\bot : \mathcal{L}) \times \Gamma \mapsto Q^*$$

Exemple $binop(EQ, e_1, e_2)$:

$$\begin{aligned}
&T_c[\![binop(EQ, e_1, e_2)]\!]\ l_\top\ l_\bot\ \gamma = \\
&\quad let\ (v_1, q_1) = T_e[\![e_1]\!]\ \gamma \\
&\quad let\ (v_2, q_2) = T_e[\![e_2]\!]\ \gamma \\
&\quad q_1\ @\ q_2\ @\ [\\
&\qquad if\ v_1 = v_2\ goto\ l_\top \\
&\qquad goto\ l_\bot \\
&\quad]
\end{aligned}$$

Sélection revue et corrigée



$$
\begin{aligned}
&T_s[\![if(e, s_1, s_2)]\!]\ \gamma = \\
&\quad let\ l_1, l_2, l_3 = new\_lab\ 3 \\
&\quad let\ q_e = T_c[\![e]\!]\ l_1\ l_2\ \gamma \\
&\quad let\ q_1 = T_s[\![s_1]\!]\ \gamma \\
&\quad let\ q_2 = T_s[\![s_2]\!]\ \gamma \\
&\qquad q_e \\
&\quad @\ [label\ l_1] \\
&\quad @\ q_1 \\
&\quad @\ [goto\ l_3] \\
&\quad @\ [label\ l_2] \\
&\quad @\ q_2 \\
&\quad @\ [label\ l_3]
\end{aligned}
$$

Exercice 4

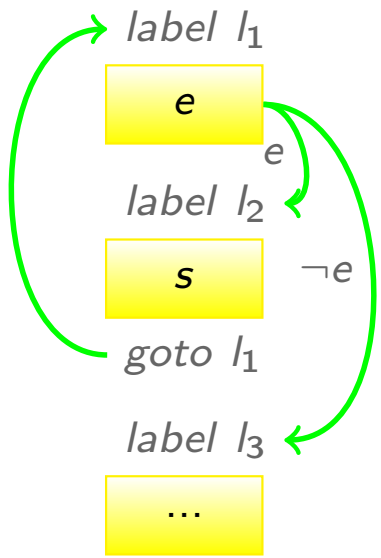
On désire réaliser la traduction de $while(e, s)$:

1. donner le schéma de traduction où apparaissent les branchements et les étiquettes
2. écrire la fonction de traduction T_s pour réaliser la traduction du *while*

Exercice 4

On désire réaliser la traduction de $while(e, s)$:

1. donner le schéma de traduction où apparaissent les branchements et les étiquettes
2. écrire la fonction de traduction T_s pour réaliser la traduction du *while*



Exercice 4

On désire réaliser la traduction de $while(e, s)$:

1. donner le schéma de traduction où apparaissent les branchements et les étiquettes
2. écrire la fonction de traduction T_s pour réaliser la traduction du *while*

$$
\begin{aligned}
&T_s[\![while(e,s)]\!]\ \gamma = \\
&\quad let\ l_1, l_2, l_3 = new\_lab\ 3 \\
&\quad let\ q_e = T_c[\![e]\!]\ 2\ 3\ \gamma \\
&\qquad [label\ l_1] \\
&\quad @\ q_e \\
&\quad @\ [label\ l_2] \\
&\quad @\ q_s \\
&\quad @\ [goto\ l_1] \\
&\quad @\ [label\ l_3]
\end{aligned}
$$

Traduction des conditions simples

Opérateur de comparaison :

$$
\begin{aligned}
&T_c[\![binop(\omega, e_1, e_2]\!]\ \top\ bot\ \gamma = \\
&\quad let\ (v_1, q_1) = T_e[\![e_1]\!]\ \gamma \\
&\quad let\ (v_2, q_2) = T_e[\![e_2]\!]\ \gamma \\
&\quad q1\ @\ q2\ @\ [\\
&\qquad if\ v_1\ \omega\ v_2\ goto\ l_\top \\
&\qquad goto\ l_\bot \\
&\quad]
\end{aligned}
$$

Avec $\omega \in \{EQ, NE, LT, LE, GT, GE\}$.

Autres conditions :

$$
\begin{aligned}
&T_c[\![e]\!]\ l_\top\ l_\bot\ \gamma = \\
&\quad T_c[\![binop(NE, e, cst(intv(0)))]\!] \\
&\qquad\qquad\qquad\qquad l_\top\ l_\bot\ \gamma
\end{aligned}
$$

Optimisation à lucarne :

$$
\begin{aligned}
&if\ v_1\ \omega\ v_2\ goto\ l \\
&goto\ l' \\
&label\ l : \\
&\qquad \Updownarrow \\
&if\ v_1\ \overline{\omega}\ v_2\ goto\ l'
\end{aligned}
$$

Avec $\overline{\omega}$ comparateur inverse de ω.

Évaluation en circuit court

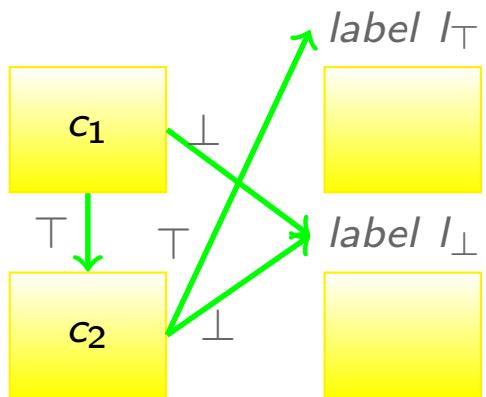
Évaluation du c_1 && c_2 :

- si c_1 s'évalue à faux $\Rightarrow$ condition fausse $\Rightarrow$ branchement sur $l_\perp$
- sinon brancher selon la condition c_2

Évaluation en circuit court

Évaluation du c_1 && c_2 :

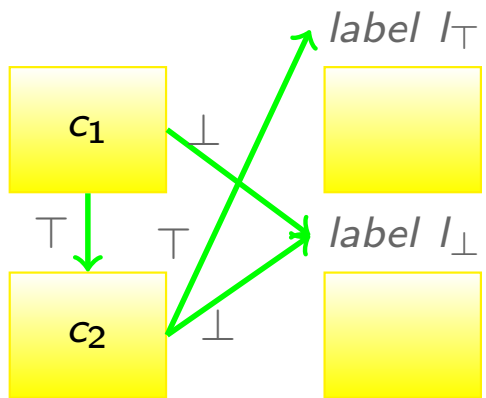
- si c_1 s'évalue à faux $\Rightarrow$ condition fausse $\Rightarrow$ branchement sur $l_\perp$
- sinon brancher selon la condition c_2



Évaluation en circuit court

Évaluation du c_1 && c_2 :

- si c_1 s'évalue à faux $\Rightarrow$ condition fausse $\Rightarrow$ branchement sur $l_\perp$
- sinon brancher selon la condition c_2



$$
\begin{aligned}
&T_c[\![binop(LOG\_AND, c_1, c_2)]\!]\ l_\top\ l_\perp\ \gamma = \\
&\quad let\ l = new\_lab\ 1 \\
&\qquad (T_c[\![c_1]\!]\ l\ l_\perp\ \gamma) \\
&\quad @\ [label\ l] \\
&\quad @\ (T_c[\![c_2]\!]\ l_\top\ l_\perp\ \gamma)
\end{aligned}
$$

Exercice 5

1. Donnez la définition de $T_c[\![unop(NOT, c)]\!]$.
2. Proposez un schéma d'évaluation en circuit court pour $binop(LOG\_OR, c_1, c_2)$.
3. Donnez la définition de $binop(LOG\_OR, c_1, c_2)$.

Exercice 5

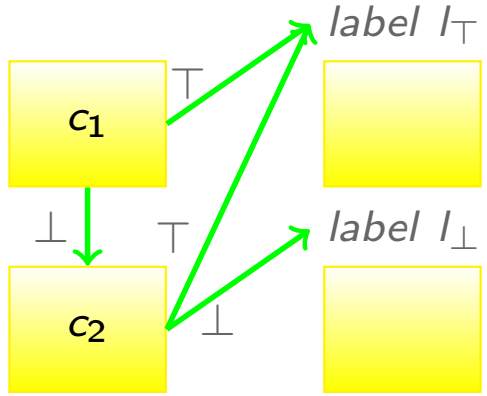
1. Donnez la définition de $T_c[\![unop(NOT, c)]\!]$.
2. Proposez un schéma d'évaluation en circuit court pour $binop(LOG\_OR, c_1, c_2)$.
3. Donnez la définition de $binop(LOG\_OR, c_1, c_2)$.

$$T_c[\![unop(NOT, c)]\!]\ I_\top\ I_\bot\ \gamma = T_c[\![c]\!]\ I_\bot\ I_\top\ \gamma$$

Exercice 5

1. Donnez la définition de $T_c[\![unop(NOT, c)]\!]$.
2. Proposez un schéma d'évaluation en circuit court pour $binop(LOG\_OR, c_1, c_2)$.
3. Donnez la définition de $binop(LOG\_OR, c_1, c_2)$.

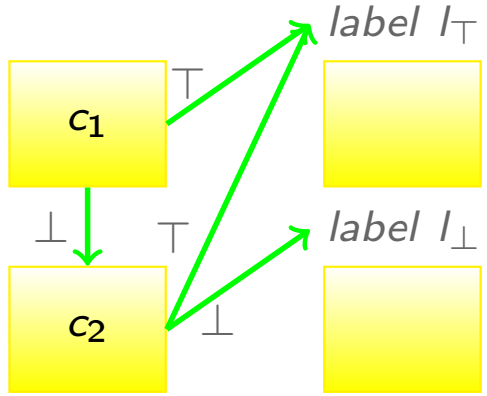
$$T_c[\![unop(NOT, c)]\!]\ l_\top\ l_\bot\ \gamma = T_c[\![c]\!]\ l_\bot\ l_\top\ \gamma$$



Exercice 5

1. Donnez la définition de $T_c[\![unop(NOT, c)]\!]$.
2. Proposez un schéma d'évaluation en circuit court pour $binop(LOG\_OR, c_1, c_2)$.
3. Donnez la définition de $binop(LOG\_OR, c_1, c_2)$.

$$T_c[\![unop(NOT, c)]\!]\ l_\top\ l_\bot\ \gamma = T_c[\![c]\!]\ l_\bot\ l_\top\ \gamma$$



$$T_c[\![binop(LOG\_OR, c_1, c_2]\!]\ l_\top\ l_\bot\ \gamma =$$
$$let\ l = new\_lab\ 1$$
$$(T_c[\![c_1]\!]\ l_\top\ l\ \gamma)$$
$$@\ [label\ l]$$
$$@\ (T_c[\![c_2]\!]\ l_\top\ l_\bot\ \gamma)$$

Plan

1. Introduction
2. Les quadruplets
3. Traduction des expressions
4. Traduction des instructions
5. Traduction des fonctions et variables globales
6. Conclusion

Les variables globales

Caractéristiques d'une variable globale :

- identificateur $\Rightarrow$ .symtab
- type $\Rightarrow$ taille + alignement
- valeur initiale

Stockage dans les sections exécutables :

- variable globale constante $\Rightarrow$ allocation dans .rodata
- variable globale initialisée $\Rightarrow$ allocation dans .data
- variable globale non-initialisée $\Rightarrow$ allocation dans .bss
- variable externe $\Rightarrow$ enregistrement dans .rel (relocation)

Les sous-programmes

Caractéristiques d'une fonction :

- identificateur $\Rightarrow$ .symtab
- paramètres $\Rightarrow$ pile
- variables locales $\Rightarrow$ pile
- corps $\Rightarrow$ traduit en quadruplet $\Rightarrow$ traduit en assembleur $\Rightarrow$ .text

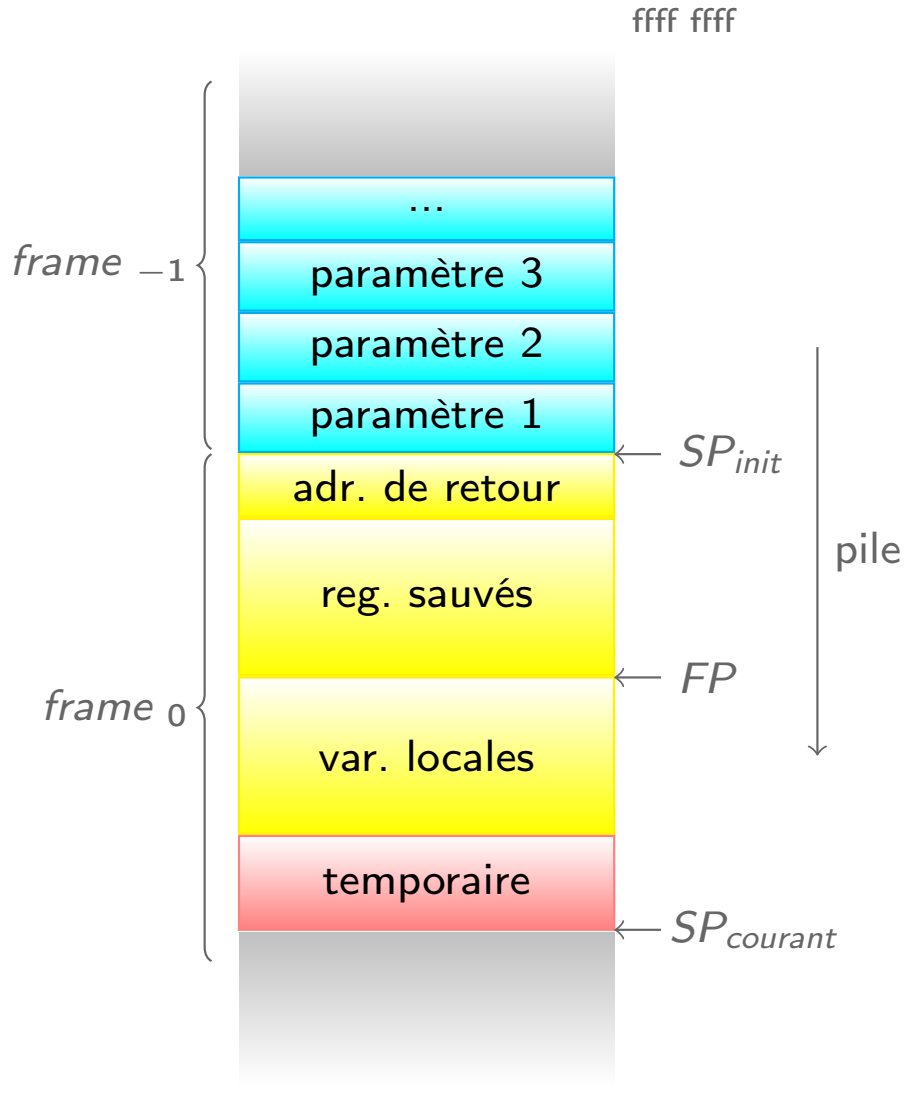
Le bloc d'activation

Paramètres et variables locales :

- identificateur
- type $\Rightarrow$ taille + alignement
- déplacement dans la pile

Bloc d'activation (*frame*) :

- adresse de retour
- sauvegarde des registres
- variables locales
- variables temporaires
- paramètres



Passage des paramètres et de résultat

Médias possibles :

- dans la mémoire globale $\Rightarrow$ ne permet la récursivité
- dans la pile
- dans les registres

Dépendant de l'architecture et du système : défini dans une ABI (*Application Binary Interface*) $\Rightarrow$ compatiblité bibliothèque
Exemple : ARM

- 4 premiers paramètres dans les registres R_0 à R_3
- paramètres suivants dans la pile
- place réservée dans la pile pour les 4 premiers paramètres
- résultat $\leq$ 32-bit dans R_0
- résultat sur 64-bit dans R_0 et R_1
- paramètre sur 64-bit dans paire de registres R_i et R_{i+1}
- SP dans R_{13} et FP dans R_{12}
- les paramètres de pile sont stocké sur une taille multiple de 4

Compilation d'un sous-programme

Données :

- k_{save} – taille adresse de retour + registres sauvés
- k_{local} – taille variables locales
- k_i^p – déplacement du paramètre i par rapport à SP_{init}
- k_i^l – déplacement de la variable locale i par rapport à FP

Adresses locales :

- paramètre i : $FP + k_{save} + k_i^p$
- variable locale i : $FP - k_i^l$

$$
\begin{aligned}
&T_f[\![fun(\tau, i, s)]\!]\ \gamma = \\
&\quad let\ v = new\_temp\ ptr(void) \\
&\quad let\ q = T_s[\![s]\!]\ \gamma \\
&\qquad [push\ r, \forall r \in used(q) \cup \{FP\}] \\
&\quad @\ [FP \leftarrow SP] \\
&\quad @\ [v \leftarrow k_{local}] \\
&\quad @\ [SP \leftarrow SP - v] \\
&\quad @\ q \\
&\quad @\ [SP \leftarrow FP] \\
&\quad @\ [pop\ r, \forall r \in used(q) \cup \{FP\}] \\
&\quad @\ [return]
\end{aligned}
$$

Exercice 6

En remarquant que :

1. l'instruction *return* réalise un branchement en fin de sous-programme.
2. on nomme v_r le pseudo-registre associé au registre de résultat d'un sous-programme.

Donnez la traduction du *return*(e).

Exercice 6

En remarquant que :

1. l'instruction *return* réalise un branchement en fin de sous-programme.
2. on nomme v_r le pseudo-registre associé au registre de résultat d'un sous-programme.

Donnez la traduction du $return(e)$.

$$
\begin{aligned}
&T_f[\![fun(\tau, i, s)]\!]\ \gamma = \\
&\quad let\ v = new\_temp\ ptr(void) \\
&\quad let\ l_{retur} = new\_lab\ 1 \\
&\quad let\ q = T_s[\![s]\!]\ \gamma[\text{"\$return"} \rightarrow l_{return}] \\
&\quad ... \\
&\quad @\ [label\ l_{return}] \\
&\quad @\ [return]
\end{aligned}
$$

Exercice 6

En remarquant que :

1. l'instruction *return* réalise un branchement en fin de sous-programme.
2. on nomme v_r le pseudo-registre associé au registre de résultat d'un sous-programme.

Donnez la traduction du $return(e)$.

$$
\begin{aligned}
&T_f[\![fun(\tau, i, s)]\!]\ \gamma = \\
&\quad let\ v = new\_temp\ ptr(void) \\
&\quad let\ l_{retur} = new\_lab\ 1 \\
&\quad let\ q = T_s[\![s]\!]\ \gamma["\$return" \rightarrow l_{return}] \\
&\quad ... \\
&\quad @\ [label\ l_{return}] \\
&\quad @\ [return]
\end{aligned}
$$

$$
\begin{aligned}
&T_s[\![return(e)]\!]\ \gamma = \\
&\quad let(v, q) = T_e[\![e]\!]\ \gamma \\
&\quad \begin{cases} [\,] & if\ e = null \\ [v_r \leftarrow v] & else \end{cases} \\
&\quad @\ [goto\ \gamma["\$return"]]
\end{aligned}
$$

Execice 7

On ajoute aux instructionx S les constructeurs *break* et *continue* (qui ne prennent pas de paramètre et sont très faciles à typer).
Donner les traductions :

- $T_s[\![break]\!]\ \gamma$
- $T_s[\![continue]\!]\ \gamma$

Execice 7

On ajoute aux instructionx S les constructeurs *break* et *continue* (qui ne prennent pas de paramètre et sont très faciles à typer).
Donner les traductions :

- $T_s[\![break]\!]\ \gamma$
- $T_s[\![continue]\!]\ \gamma$

$$
\begin{aligned}
&T_s[\![while(e,s)]\!]\ \gamma = \\
&\quad let\ l_1, l_2, l_3 = new\_lab\ 3 \\
&\quad let\ q_e = T_c[\![e]\!]\ 2\ 3\ \gamma \\
&\qquad [label\ l_1] \\
&\quad @\ q_e \\
&\quad @\ [label\ l_2] \\
&\quad @\ T_s[\![s]\!]\ \gamma["\$brk" \to l_3, "\$ctn" \to l_1] \\
&\quad @\ [goto\ l_1] \\
&\quad @\ [label\ l_3]
\end{aligned}
$$

Execice 7

On ajoute aux instructionx S les constructeurs *break* et *continue* (qui ne prennent pas de paramètre et sont très faciles à typer).
Donner les traductions :

- $T_s[\![break]\!]\ \gamma$
- $T_s[\![continue]\!]\ \gamma$

$$
\begin{aligned}
&T_s[\![while(e,s)]\!]\ \gamma = \\
&\quad let\ l_1, l_2, l_3 = new\_lab\ 3 \\
&\quad let\ q_e = T_c[\![e]\!]\ 2\ 3\ \gamma \\
&\qquad [label\ l_1] \\
&\quad @\ q_e \\
&\quad @\ [label\ l_2] \\
&\quad @\ T_s[\![s]\!]\ \gamma["\$brk" \to l_3, "\$ctn" \to l_1] \\
&\quad @\ [goto\ l_1] \\
&\quad @\ [label\ l_3]
\end{aligned}
$$

$$
\begin{aligned}
&T_s[\![break]\!]\ \gamma = \\
&\quad [goto\ \gamma["\$brk"]\ ] \\
&T_s[\![continue]\!]\ \gamma = \\
&\quad [goto\ \gamma["\$ctn"]\ ]
\end{aligned}
$$

Plan

1. Introduction
2. Les quadruplets
3. Traduction des expressions
4. Traduction des instructions
5. Traduction des fonctions et variables globales
6. Conclusion

Conclusion

- traduction AST $\Rightarrow$ quadruplets
- focus sur un constructeur à la fois
- définition de fonction de traduction dépendant du contexte
  - T_s pour les instructions
  - T_e pour les expressions produisant une valeur
  - T_c pour les expressions condition
  - T_r pour produire des adresses de référence
- utilisation de l'environnement pour les instructions couplées
  - *while* + *break* + *continue*
  - sous-programme + *return*