Université Paris 7-Denis Diderot
Maîtrise d'Informatique 1994-95

Projet de Compilation

Réalisation d'un compilateur

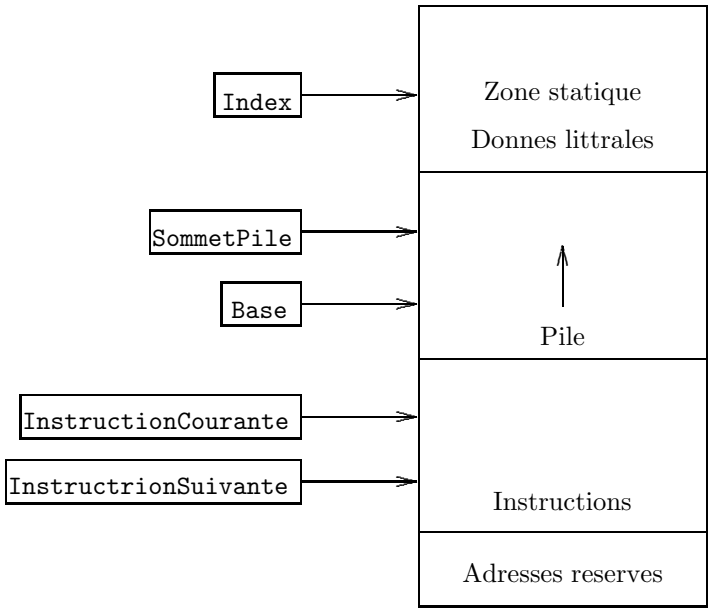
Le but du projet est de construire un compilateur d'un petit langage de programmation évolué, ressemblant à un sous-ensemble de Pascal, dans un langage intermédiaire, style assembleur à trois adresses. Pour des fins de vérifications, on utilisera un interpréteur de cet assembleur qui donnera un résultat exécutable.

1 Le langage intermédiaire

Le langage intermédiaire est celui d'une machine virtuelle décrite ci-dessous.

La mémoire

Une configuration typique de la mémoire est présentée dans la figure suivante.



L'organisation de la mémoire pourra être vue comme suit :
- une zone statique, pouvant éventuellement contenir des valeurs littérales, c'est-à-dire ici des constantes entières ou booléennes.
- une pile,
- une zone dans laquelle sont rangées les instructions du programme,
- une zone d'adresses réservées.

D'autre part, on dispose des registres suivants :
- un compteur de programme `InstructionCourante`, pointant sur l'instruction en cours,
- un compteur `InstructionSuivante` pointant sur la prochaine instruction à effectuer,
- un registre `SommetPile` pointant sur le sommet de la pile,
- un registre `Base` qui contient l'adresse de la base de l'enregistrement d'activation courant,
- un registre d'index `Index`.

Les instructions et les modes d'adressage

Les modes d'adressage sont les suivants :

Nom	Syntaxe	Signification
Immédiat	#A	A est interprété littéralement
Direct	A	A est l'adresse de l'opérande
Indirect	@A	A contient l'adresse de l'opérande
Indexé	Index(A)	l'adresse de l'opérande s'obtient en ajoutant le déplacement A au contenu du registre d'index. A est interpreté littéralement

Les instructions sont données dans le tableau qui suit. Ce sont des instructions à une, deux, ou trois adresses. La signification est donnée de façon informelle : on ne tient pas compte des modes d'adressage employés dans la colonne "Signification". A, B et C sont le plus souvent interprétés comme s'ils étaient donnés en mode indirect. La signification réelle doit bien entendu tenir compte des modes d'adressage. Certains modes sont évidemment interdits ; par exemple, une destination n'est jamais donnée en adressage immédiat.

Les accents ne sont mis ici que pour la lisibilité ; on pourra ne pas s'en préoccuper.

Nom	Arguments	Opération réalisée
LIRE	A	Lit une valeur sur l'entrée standard et la charge dans A
ÉCRIRE	A	Écrit A sur la sortie standard
TRANSFÉRER	A B	Transfère A dans B
EMPILER	A	Empile A
DÉPILER	A	Dépile A
INCRÉMENTER	A	Incrémente A
DECRÉMENTER	A	Décrémente A
ADDITIONNER	A B C	C reçoit A+B
SOUSTRAIRE	A B C	C reçoit A−B
MULTIPLIER	A B C	C reçoit A∗B
DIVISER	A B C	C reçoit le quotient de la division entière de A par B
PRENDRE_LE_RESTE	A B C	C reçoit le reste de la division entière de A par B
CHANGER_DE_SIGNE	A B	B reçoit l'opposé de A
ET	A B C	C reçoit le "et" logique, bit à bit, de A et B
OU	A B C	C reçoit le "ou" logique, bit à bit, de A et B
NON	A B	B reçoit le "non" logique, bit à bit, de A
TEST_SI_INFÉRIEUR	A B C	C reçoit 1 ou 0 suivant que A est inférieur ou non à B
TEST_SI_NUL	A B	B reçoit 1 ou 0 suivant que A est nul ou non
BRANCHEMENT	A	Branchement inconditionnel à l'adresse spécifiée par A
BRANCHEMENT_SI_VRAI	A B	Branchement à l'adresse spécifiée par A si le booléen désigné pas B est vrai
APPELER	A	Appelle la fonction désignée par A
RETOURNER		Retour d'appel
ARRÊTER		Arrêt
PAS_D_OPÉRATION		Ne fait rien

2 Le langage à compiler

Le langage à compiler est du pseudo-Pascal. Les seuls types sont les types entier et booléen, et l'on ne construit pas de types nouveaux, hors les tableaux. Il n'y a pas de procédures, mais, il y a des fonctions, qui peuvent être appelées récursivement, les paramètres étant passés par valeur.

La syntaxe de ce langage est décrite par la grammaire suivante :

$<$ *programme* $>$ $\longrightarrow$ **début** $<$ *liste de déclarations* $><$ *liste de fonctions* $>$
$<$ *liste d'instructions* $>$ **fin**

$<$ *liste de déclarations* $>$ $\longrightarrow$ ε | $<$ *déclaration* $>$ **;** $<$ *liste de déclarations* $>$

$<$ *déclaration* $>$ $\longrightarrow$ **entier** $<$ *identificateur* $>$ | **booléen** $<$ *identificateur* $>$
| **tableau d'entiers** $<$ *identificateur* $>$ [$<$ *entier* $>$]
| **tableau de booléens** $<$ *identificateur* $>$ [$<$ *entier* $>$]

$<$ *liste de fonctions* $>$ $\longrightarrow$ ε | $<$ *fonction* $>$ **;** $<$ *liste de fonctions* $>$

$<$ *fonction* $>$ $\longrightarrow$ $<$ *en-tête* $><$ *corps* $>$

$<$ *en-tête* $>$ $\longrightarrow$ **fonction** $<$ *type* $><$ *identificateur* $>$ **(** $<$ *suite de paramètres* $>$ **)**

$<$ *corps* $>$ $\longrightarrow$ **début** $<$ *liste de déclarations* $><$ *liste d'instructions* $>$ **fin**

$<$ *type* $>$ $\longrightarrow$ **vide** | **entière** | **booléenne**

$<$ *suite de paramètres* $>$ $\longrightarrow$ ε | $<$ *liste de paramètres* $>$

$<$ *liste de paramètres* $>$ $\longrightarrow$ $<$ *paramètre* $>$ | $<$ *paramètre* $>$ **,** $<$ *liste de paramètres* $>$

$<$ *paramètre* $>$ $\longrightarrow$ **entier** $<$ *identificateur* $>$ | **booléen** $<$ *identificateur* $>$

$<$ *bloc d'intructions* $>$ $\longrightarrow$ **début** $<$ *liste d'instructions* $>$ **fin**

$<$ *liste d'instructions* $>$ $\longrightarrow$ ε | $<$ *instruction* $>$ **;** $<$ *liste d'instructions* $>$

$<$ *instruction* $>$ $\longrightarrow$ $<$ *identificateur* $>$:= $<$ *expression* $>$
| $<$ *identificateur* $>$ [$<$ *expression arithmétique* $>$] := $<$ *expression* $>$
| **si** $<$ *expression logique* $>$ **alors** $<$ *bloc d'instructions* $>$
sinon $<$ *bloc d'instructions* $>$
| **tant que** $<$ *expression logique* $>$ **faire** $<$ *bloc d'instructions* $>$
| **arrêt** | **retour**
| **écrire** $<$ *liste d'arguments* $>$ | **lire** $<$ *identificateur* $>$

$<$ *identificateur* $>$ $\longrightarrow$ $<$ *lettre* $>$ | $<$ *lettre* $><$ *identificateur* $>$

$<$ *lettre* $>$ $\longrightarrow$ **a** | ... | **z**

$<$ *expression* $>$ $\longrightarrow$ $<$ *expression arithmétique* $>$ | $<$ *expression logique* $>$
| $<$ *identificateur* $>$ **(** $<$ *suite d'arguments* $>$ **)**

$<$ *expression arithmétique* $>$ $\longrightarrow$ $<$ *expression arithmétique* $>$ + $<$ *terme* $>$
| $<$ *expression arithmétique* $>$ − $<$ *terme* $>$ | $<$ *terme* $>$

$<$ *terme* $>$ $\longrightarrow$ $<$ *terme* $>$ $\star$ $<$ *facteur* $>$ | $<$ *terme* $>$ **div** $<$ *facteur* $>$ | $<$ *facteur* $>$

$<$ *facteur* $>$ $\longrightarrow$ $<$ *identificateur* $>$ | $<$ *constante numérique* $>$
| **(** $<$ *expression arithmétique* $>$ **)**
| $<$ *identificateur* $>$ [$<$ *expression arithmétique* $>$]

$<$ *constante numérique* $>$ $\longrightarrow$ $<$ *entier signé* $>$

$<$ *entier signé* $>$ $\longrightarrow$ + $<$ *entier* $>$ | − $<$ *entier* $>$ | $<$ *entier* $>$

$<$ *entier* $>$ $\longrightarrow$ $<$ *chiffre* $>$ | $<$ *chiffre* $><$ *entier* $>$

$<$ *chiffre* $>$ $\longrightarrow$ **0** | ... | **9**

$<$ *suite d'arguments* $>$ $\longrightarrow$ ε | $<$ *liste d'arguments* $>$

$<$ *liste d'arguments* $>$ $\longrightarrow$ $<$ *argument* $>$ | $<$ *argument* $>$ **,** $<$ *liste d'arguments* $>$

$<$ *argument* $>$ $\longrightarrow$ $<$ *expression* $>$

$<$ *expression logique* $>$ $\longrightarrow$ $<$ *expression logique* $>$ $\vee$ $<$ *terme logique* $>$ | $<$ *terme logique* $>$

$< \textit{terme logique} >$ $\longrightarrow$ $< \textit{terme logique} > \wedge < \textit{facteur logique} > \mid < \textit{facteur logique} >$
$< \textit{facteur logique} >$ $\longrightarrow$ $< \textit{atome} > \mid$ ($< \textit{expression logique} >$) $\mid$ ($< \textit{expression logique} >$)
$\mid$ ($< \textit{expression arithmétique} >< \textit{comparaison} >$
$< \textit{expression arithmétique} >$)
$< \textit{atome} >$ $\longrightarrow$ $< \textit{identificateur} > \mid \; < \textit{identificateur} > \mid < \textit{constante logique} >$
$< \textit{constante logique} >$ $\longrightarrow$ **vrai** | **faux**
$< \textit{comparaison} >$ $\longrightarrow$ $< \mid > \mid = \mid <= \mid >= \mid <>$

Les commentaires sont les textes entre{ et}. Les blancs sont interdits au milieu d'un mot clé, ignorés partout ailleurs. Les accents ne sont mis ici que pour la lisibilité ; on pourra ne pas s'en préoccuper.

3 Travail demandé

Le travail demandé peut être effectué par binômes. Il se conjugue en trois temps. Néanmoins, il est conseillé de réfléchir dès le départ à l'ensemble des tâches qu'il faudra effectuer, et de prévoir en conséquence les possibilités d'extensions qui seront nécessaires. Tout n'est pas entièrement dit dans ce sujet. Il n'est pas interdit de faire preuve d'imagination.

3.1 premier temps : pour le 12 décembre 94

3.1.1 l'interpréteur

Il s'agit d'écrire un petit interpréteur pour le langage intermédiaire.

3.1.2 l'analyseur lexical

Il s'agit d'écrire un analyseur lexical pour le petit sous-ensemble du langage à compiler défini par la grammaire suivante :

$< \textit{suite d'intructions} >$ $\longrightarrow$ **début** $< \textit{liste d'instructions} >$ **fin**
$< \textit{liste d'instructions} >$ $\longrightarrow$ $\varepsilon \mid < \textit{instruction} >$ **;** $< \textit{liste d'instructions} >$
$< \textit{instruction} >$ $\longrightarrow$ $< \textit{identificateur} > := < \textit{expression} >$
$< \textit{identificateur} >$ $\longrightarrow$ $< \textit{lettre} > \mid < \textit{lettre} >< \textit{identificateur} >$
$< \textit{expression} >$ $\longrightarrow$ $< \textit{expression} > + < \textit{terme} > \mid < \textit{expression} > - < \textit{terme} > \mid < \textit{terme} >$
$< \textit{terme} >$ $\longrightarrow$ $< \textit{terme} > \star < \textit{facteur} > \mid < \textit{terme} >$ **div** $< \textit{facteur} > \mid < \textit{facteur} >$
$< \textit{facteur} >$ $\longrightarrow$ $< \textit{identificateur} > \mid < \textit{entier} >$
$< \textit{entier} >$ $\longrightarrow$ $< \textit{chiffre} > \mid < \textit{chiffre} >< \textit{entier} >$
$< \textit{lettre} >$ $\longrightarrow$ **a** | ... | **z**
$< \textit{chiffre} >$ $\longrightarrow$ **0** | ... | **9**

Le programme supprimera les blancs et les commentaires. Il fournira une suite de lexèmes à partir d'un texte de programme.

3.2 deuxième temps : pour le 9 janvier 95

Dans cette partie, on se concentre sur l'analyseur syntaxique. On se restreint à un sous-ensemble du langage à compiler sans fonctions et sans constructions de types, défini par la grammaire suivante :

$< \textit{programme} >$ $\longrightarrow$ **début** $< \textit{liste de déclarations} >< \textit{liste d'instructions} >$ **fin**

$< \textit{liste de déclarations} >$ $\longrightarrow$ $\varepsilon \mid < \textit{déclaration} >$ **;** $< \textit{liste de déclarations} >$

$< \textit{déclaration} >$ $\longrightarrow$ **entier** $< \textit{identificateur} > \mid$ **booléen** $< \textit{identificateur} >$

$< \textit{bloc d'intructions} >$ $\longrightarrow$ **début** $< \textit{liste d'instructions} >$ **fin**

$< \textit{liste d'instructions} >$ $\longrightarrow$ $\varepsilon \mid < \textit{instruction} >$ **;** $< \textit{liste d'instructions} >$

$< \textit{instruction} >$ $\longrightarrow$ $< \textit{identificateur} > := < \textit{expression} >$
$\mid$ **si** $< \textit{expression logique} >$ **alors** $< \textit{bloc d'instructions} >$
sinon $< \textit{bloc d'instructions} >$
$\mid$ **tant que** $< \textit{expression logique} >$ **faire** $< \textit{bloc d'instructions} >$
$\mid$ **arrêt**

$< \textit{identificateur} >$ $\longrightarrow$ $< \textit{lettre} > \mid < \textit{lettre} >< \textit{identificateur} >$

$< \textit{lettre} >$ $\longrightarrow$ **a** $\mid \ldots \mid$ **z**

$< \textit{expression} >$ $\longrightarrow$ $< \textit{expression arithmétique} > \mid < \textit{expression logique} >$

$< \textit{expression arithmétique} >$ $\longrightarrow$ $< \textit{expression arithmétique} > + < \textit{terme} >$
$\mid < \textit{expression arithmétique} > - < \textit{terme} > \mid < \textit{terme} >$

$< \textit{terme} >$ $\longrightarrow$ $< \textit{terme} > \star < \textit{facteur} > \mid < \textit{terme} >$ **div** $< \textit{facteur} > \mid < \textit{facteur} >$

$< \textit{facteur} >$ $\longrightarrow$ $< \textit{identificateur} > \mid < \textit{constante numérique} >$
$\mid$ **(** $< \textit{expression arithmétique} >$ **)**

$< \textit{constante numérique} >$ $\longrightarrow$ $< \textit{entier signé} >$

$< \textit{entier signé} >$ $\longrightarrow$ $+ < \textit{entier} > \mid - < \textit{entier} > \mid < \textit{entier} >$

$< \textit{entier} >$ $\longrightarrow$ $< \textit{chiffre} > \mid < \textit{chiffre} >< \textit{entier} >$

$< \textit{chiffre} >$ $\longrightarrow$ **0** $\mid \ldots \mid$ **9**

$< \textit{expression logique} >$ $\longrightarrow$ $< \textit{expression logique} > \vee < \textit{terme logique} > \mid < \textit{terme logique} >$

$< \textit{terme logique} >$ $\longrightarrow$ $< \textit{terme logique} > \wedge < \textit{facteur logique} > \mid < \textit{facteur logique} >$

$< \textit{facteur logique} >$ $\longrightarrow$ $< \textit{atome} > \mid$ **(** $< \textit{expression logique} >$ **)** $\mid$ **(** $< \textit{expression logique} >$ **)**
$\mid$ **(** $< \textit{expression arithmétique} >< \textit{comparaison} >$
$< \textit{expression arithmétique} >$ **)**

$< \textit{atome} >$ $\longrightarrow$ $< \textit{identificateur} > \mid \ < \textit{identificateur} > \mid < \textit{constante logique} >$

$< \textit{constante logique} >$ $\longrightarrow$ **vrai** $\mid$ **faux**

$< \textit{comparaison} >$ $\longrightarrow$ $< \mid > \mid = \mid <= \mid >= \mid <>$

On demande d'écrire un compilateur complet. On pourra utiliser ici Lex pour construire l'analyseur lexical.

3.3 troisième temps : pour le 27 janvier 95

Dans cette partie, on demande bien sûr l'écriture du compilateur complet, avec des jeux d'essais convaincants. Tous les modules et leur assemblage auront étés testés sur un nombre suffisant d'exemples significatifs.

On pourra utiliser ici Yacc pour construire l'analyseur syntaxique.
On pourra enfin envisager des extensions à ce projet, dans la mesure où le reste du travail demandé aura été correctement effectué.