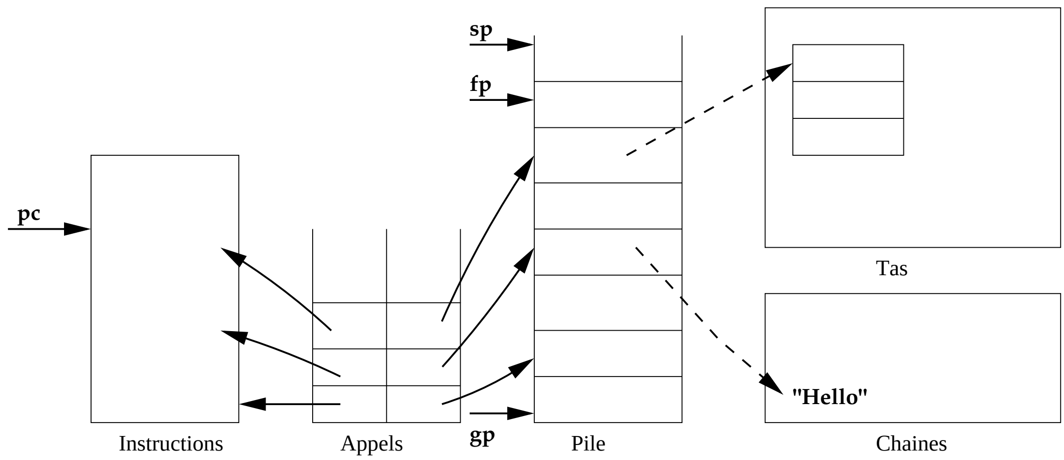
Machine virtuelle pour le projet de compilation

1 Description

1.1 Organisation de la machine

Il s'agit d'une machine à pile (par opposition à une machine à registres), composée d'une pile d'exécution, d'une pile d'appels, d'un code, de deux tas et de quatre registres.



La pile d'exécution contient des *valeurs*, qui peuvent être des entiers, des réels ou des adresses.

Les deux tas contiennent respectivement des chaînes de caractères et des blocs structurés, tous deux repérés par leurs adresses. Un bloc structuré contient un certain nombre de valeurs (de la même nature que celles se trouvant sur la pile).

Une adresse peut être de quatre natures différentes : vers le code, vers la pile, vers un bloc structuré dans le tas ou vers une chaîne.

Trois registres permettent d'accéder à différentes parties de la pile :
- Le registre *sp* (stack pointer) repère le sommet courant de la pile, il pointe sur la première case libre de la pile.
- Le registre *fp* (frame pointer) repère l'adresse de base des variables locales.
- Le registre *gp* contient l'adresse de base des variables globales.

La machine comporte un registre *pc* qui pointe sur l'instruction courante du code à exécuter.

La pile d'appels permet de sauvegarder les appels : elle contient des couples de pointeurs, l'un sauvegardant le registre d'instructions *pc* et l'autre le registre *fp*.

1.2 Instructions

Les instructions sont désignées par un nom et peuvent prendre un ou deux arguments. Ceux-ci peuvent être :
- des constantes entières,

- des constantes réelles,
- des chaînes de caractères entre guillemets avec les mêmes conventions que le langage C pour ce qui concerne les caractères spéciaux `\"`, `\n`, `\\`,
- une étiquette symbolique désignant un emplacement dans le code.

1.2.1 Terminologie

On utilisera les conventions suivantes :
- Empiler une valeur x signifie stocker x à l'emplacement $\mathsf{P}[sp]$ et incrémenter sp de 1.
- Empiler n fois une valeur x signifie itérer n fois l'opération précédente.
- Dépiler n valeurs revient à décrémenter de n la valeur de sp.
- Le sommet de la pile représente la dernière valeur stockée dans la pile soit $\mathsf{P}[sp-1]$, le sous-somment de la pile représente l'avant dernière valeur stockée dans la pile soit $\mathsf{P}[sp-2]$.

1.2.2 Opérations de base

Les opérations arithmétiques ou flottantes s'effectuent entre le sommet et le sous-sommet de la pile, les deux arguments sont dépilés et le résultat est empilé à la place. Le résultat des opérations de comparaison est un entier qui vaut 0 ou 1. L'entier 0 représente la valeur booléenne "faux" tandis que 1 représente la valeur booléenne "vrai".

Opérations sur les entiers

Instruction	Description
ADD	dépile n puis m qui doivent être entiers et empile le résultat $m+n$
SUB	dépile n puis m qui doivent être entiers et empile le résultat $m-n$
MUL	dépile n puis m qui doivent être entiers et empile le résultat $m \times n$
DIV	dépile n puis m qui doivent être entiers et empile le résultat m/n
MOD	dépile n puis m qui doivent être entiers et empile le résultat $m \bmod n$
NOT	dépile n qui doit être un entier et empile le résultat de $n = 0$

Opérations sur les réels

Instruction	Description
FADD	dépile n puis m qui doivent être réels et empile le résultat $m+n$
FSUB	dépile n puis m qui doivent être réels et empile le résultat $m-n$
FMUL	dépile n puis m qui doivent être réels et empile le résultat $m \times n$
FDIV	dépile n puis m qui doivent être réels et empile le résultat m/n
FCOS	dépile n qui doit être réel et empilet $\cos(n)$
FSIN	dépile n qui doit être réel et empilet $\sin(n)$
FSQRT	dépile n qui doit être réel et empilet $\sqrt{n}$

Opérations sur les adresses

Instruction	Description
PADD	dépile n qui doit être un entier puis a qui doit être une adresse et empile l'adresse $a + n$
ISADDR	dépile une valeur v et empile 1 si v est une adresse (sur le tas ou la pile) et 0 sinon

Opérations sur les chaînes

Instruction	Description
CONCAT	dépile n puis m qui doivent être des adresses de chaîne, empile l'adresse d'une chaîne égale à la concaténation de la chaîne d'adresse n et de celle d'adresse m.

Opérations sur le tas

Instruction	Argument	Description
ALLOC	n entier	alloue sur le tas un bloc structuré de taille n et empile l'adresse correspondante
ALLOCN		dépile un entier n, alloue sur le tas un bloc structuré de taille n et empile l'adresse correspondante
FREE		dépile une adresse a et libère le bloc structuré alloué à l'adresse a

Égalité

Le test d'égalité teste que les deux objets (entiers, réels, chaînes de caractères ou adresses) sur la pile sont égaux,

Instruction	Description
EQUAL	dépile n puis m et empile le résultat de $n = m$ lorsque n et m sont de même type, et 0 sinon

Comparaisons

Les opérations de comparaison comparent deux objets de même type. Ces opérations sont restreintes aux objets de type entier, réels et chaînes de caractères.

Instruction	Description
INF	dépile n puis m qui doivent être de même type et empile le résultat $m < n$
INFEQ	dépile n puis m qui doivent être de même type et empile le résultat $m \leq n$
SUP	dépile n puis m qui doivent être de même type et empile le résultat $m > n$
SUPEQ	dépile n puis m qui doivent être de même type et empile le résultat $m \geq n$

Conversions
Différentes instructions permettent de convertir une chaîne de caractères en entier ou réel et réciproquement.

Instruction	Description
ATOI	dépile l'adresse d'une chaîne et empile sa conversion en nombre entier, échoue si la chaîne ne représente pas un entier.
ATOF	dépile l'adresse d'une chaîne et empile sa conversion en nombre réel, échoue si la chaîne ne représente pas un réel.
ITOF	dépile un entier et empile sa conversion en nombre réel.
FTOI	dépile un réel et empile l'entier représentant sa partie entière (obtenue en supprimant les décimales).
STRI	dépile un entier et empile l'adresse d'une chaîne représentant cet entier
STRF	dépile un réel et empile l'adresse d'une chaîne représentant ce réel

1.2.3 Manipuler des données

Si x désigne une adresse dans la pile alors $x[n]$ désigne une adresse située n cases au-dessus.

Empiler

Instruction	Argument	Description
PUSHI	n entier	empile n
PUSHN	n entier	empile n fois la valeur entière 0
PUSHF	n réel	empile n
PUSHS	n chaîne	stocke n dans la zone des chaînes et empile l'adresse
PUSHG	n entier	empile la valeur située en $gp[n]$
PUSHL	n entier	empile la valeur située en $fp[n]$
PUSHSP		empile la valeur du registre sp
PUSHFP		empile la valeur du registre fp
PUSHGP		empile la valeur du registre gp
LOAD	n entier	dépile une adresse a et empile la valeur dans la pile ou le tas située en $a[n]$
LOADN		dépile un entier n, une adresse a et empile la valeur dans la pile ou le tas située en $a[n]$
DUP	n entier	duplique et empile les n valeurs en sommet de pile
DUPN		dépile un entier n, puis duplique et empile les n valeurs en sommet de pile

Dépiler

Instruction	Argument	Description
POP	n entier	dépile n valeurs dans la pile
POPN		dépile un entier n puis dépile n valeurs dans la pile

Stocker

Instruction	Argument	Description
STOREL	n entier	dépile une valeur et la stocke dans la pile en fp[n]
STOREG	n entier	dépile une valeur et la stocke dans la pile en gp[n]
STORE	n entier	dépile une valeur v et une adresse a, stocke v à l'adresse $a[n]$ dans la pile ou le tas
STOREN		dépile une valeur v, un entier n et une adresse a, stocke v à l'adresse $a[n]$ dans la pile ou le tas

Divers

Instruction	Argument(s)	Description
CHECK	n, p entiers	vérifie que le sommet de la pile est un entier i tel que $n \leq i \leq p$, sinon déclenche une erreur
SWAP		dépile n puis m et rempile n puis m

1.3 Générateurs de nombres pseudo-aléatoires

Instruction	Description
RANDOM	dépile n qui doit être un entier strictement positif et empile un entier x tiré aléatoirement tel que $0 \leq x < n$
FRANDOM	dépile n qui doit être un réel et empile un réel x tiré aléatoirement tel que $0 \leq x < n$, si $n > 0$; $n < x \leq 0$, si $n < 0$; et $x = 0$ si $n = 0$

1.3.1 Entrées-Sorties

Les instructions suivantes permettent de gérer les entrées-sorties.

Instruction	Description
WRITEI	dépile un entier et l'imprime sur la sortie standard
WRITEF	dépile un réel et l'imprime sur la sortie standard
WRITES	dépile l'adresse d'une chaîne et imprime la chaîne correspondante sur la sortie standard
READ	lit une chaîne de caractères au clavier, terminée par un retour-chariot, stocke la chaîne (sans le retour-chariot) et empile l'adresse.

1.3.2 Primitives graphiques

Les instructions suivantes permettent de réaliser des dessins en utilisant la version graphique de la machine virtuelle. L'origine de la fenêtre graphique est en haut à gauche.

Instruction	Description
DRAWPOINT	dépile m puis n qui doivent être entiers et dessine un point de coordonnées (n, m)
DRAWLINE	dépile q, p, m puis n qui doivent être entiers et dessine un segment entre les points de coordonnées (n, m) et (p, q)
DRAWCIRCLE	dépile p, m puis n qui doivent être entiers et dessine un cercle de centre (n, m) et de rayon p
DRAWRECT	dépile h, w, m puis n qui doivent être entiers et dessine un rectangle de coin supérieur gauche (n, m), de largeur w et de hauteur h
FILLRECT	comme DRAWRECT mais dessine un rectangle plein
OPENDRAWINGAREA	Dépile h puis w qui doivent être entiers et ouvre une nouvelle fenêtre graphique de largeur w et de hauteur h
CLEARDRAWINGAREA	Efface la sortie graphique et réinitialise la couleur courante à “Noir”
GETMOUSE	empile les coordonnées entières x puis y de la souris ; les coordonnées (0,0) correspondent au coin en haut à gauche de la fenêtre graphique
SETCOLOR	dépile b, g puis r qui doivent être entiers et change la couleur courante suivant la valeur RGB définie par les trois entiers compris entre 0 et 65535
REFRESH	raffraichît la fenêtre graphique, i.e. rend visibles les opérations graphiques effectuées depuis le dernier raffraichissement

1.3.3 Le labyrinthe

La machine virtuelle contient un labyrinthe dans lequel il est possible de se déplacer et d'interagir. Le labyrinthe est formé d'un ensemble rectangulaire de 100×50 pièces (que l'on peut repérer par les coordonnées (0,0) à (99,49)) et de murs éventuels entre les pièces. Chaque pièce contient un entier, qui peut être lu ou modifié. Au démarrage de la machine, on se trouve toujours dans la même pièce (à savoir la pièce (0,0)), toujours orienté de la même façon (à savoir regardant vers la pièce (0,1)) et les pièces contiennent des valeurs aléatoires comprises entre 0 et 255. Les cinq opérations relatives au labyrinthe sont les suivantes :

Instruction	Description
GET	empile la valeur contenue dans la pièce où l'on se trouve
SET	dépile une valeur entière et l'affecte à la pièce où l'on se trouve
MOVE	tente un déplacement vers la pièce se trouvant en face de soi ; empile 1 (vrai) si le déplacement était possible, 0 (faux) sinon
TURNLEFT	effectue un quart de tour vers la gauche
TURNRIGHT	effectue un quart de tour vers la droite

Note : le labyrinthe doit être activé explicitement avec l'option `-maze` de la machine virtuelle.

1.3.4 Opérations de contrôle

Après l'exécution d'une instruction le registre *pc* est habituellement incrémenté de 1. Les opérations de contrôle permettent de modifier ce comportement par défaut.

Modification du registre *pc*

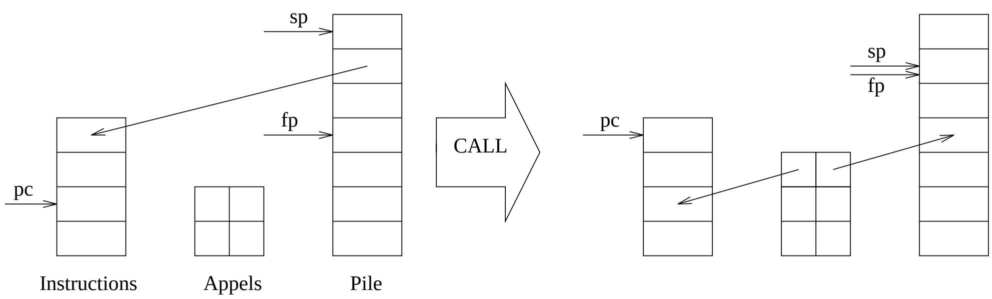
Instruction	Argument	Description
JUMP	*label* étiquette	affecte au registre *pc* l'adresse dans le programme correspondant à *label* qui peut être un entier ou une valeur symbolique.
JZ	*label* étiquette	dépile une valeur, si elle est nulle affecte au registre *pc* l'adresse dans le programme correspondant à *label* sinon incrémente *pc* de 1.
PUSHA	*label* étiquette	empile l'adresse dans le code correspondant à l'étiquette *label*

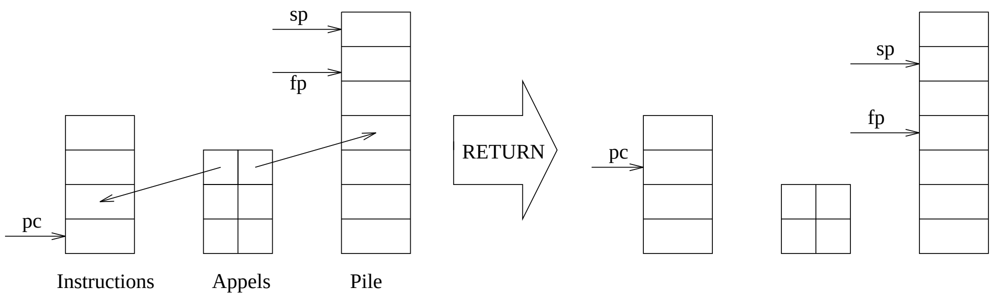
Procédure

Lors de l'appel de procédure il est nécessaire de sauvegarder les registre d'instructions et de variables locales qui seront restaurées au moment du retour.

Instruction	Description
CALL	dépile une adresse de code a, sauvegarde *pc* et *fp* dans la pile des appels, affecte à *fp* la valeur courante de *sp* et à *pc* la valeur a.
RETURN	affecte à *sp* la valeur courante de *fp*, restaure de la pile des appels les valeurs de *fp* et *pc* et incrémente *pc* de 1 pour se retrouver à l'instruction suivant celle d'appel.

Le comportement des instructions CALL et RETURN est décrit sur les schémas suivants où seuls les registres modifiés ont été dessinés :





1.3.5 Initialisation et fin

Dans l'état initial, le registre *pc* pointe sur la première instruction du programme. La pile des appels et la pile d'exécution sont vides. Les registre *gp* et *sp* pointent sur la base de la pile d'exécution tandis que *fp* n'est pas défini. Le registre *fp* doit être initialisé par l'instruction START qui ne peut être utilisée qu'une seule fois. Les instructions suivantes servent à stopper la machine à la fin du programme ou en cas d'erreur.

Instruction	Argument	Description
START		Affecte la valeur de *sp* à *fp*
NOP		ne fait rien.
ERR	x chaîne	déclenche une erreur d'instruction avec le message x.
STOP		arrête l'exécution du programme

2 Réalisation

Une réalisation de la machine virtuelle est fournie, sous la forme de deux programmes `vm` et `gvm`.

2.1 Syntaxe concrète

Les programmes pour la machine virtuelle obéissent à la syntaxe décrite ci-dessous.

2.1.1 Conventions lexicales

Espaces, tabulations et retour-chariots constituent les blancs. Les commentaires débutent par `//` et se poursuivent jusqu'à la fin de la ligne. Les identificateurs obéissent à l'expression régulière $\langle ident \rangle$ suivante :

$$
\begin{array}{lcl}
\langle digit \rangle & ::= & \texttt{0}\text{–}\texttt{9} \\
\langle alpha \rangle & ::= & \texttt{a}\text{–}\texttt{z} \mid \texttt{A}\text{–}\texttt{Z} \\
\langle ident \rangle & ::= & (\langle alpha \rangle \mid \texttt{\_})\ (\langle alpha \rangle \mid \langle digit \rangle \mid \texttt{\_} \mid \texttt{'})\star
\end{array}
$$

Les constantes entières et réelles sont définies par les expressions régulières suivantes :

$$
\begin{array}{lcl}
\langle integer \rangle & ::= & \texttt{-}\,?\ \langle digit \rangle + \\
\langle float \rangle & ::= & \texttt{-}\,?\ \langle digit \rangle +\ (\texttt{.}\ \langle digit \rangle \star)\,?\ ((\texttt{e} \mid \texttt{E})\ (\texttt{+} \mid \texttt{-})\,?\ \langle digit \rangle +)\,?
\end{array}
$$

avec la convention qu'une constante est réelle seulement si elle n'est pas aussi une constante entière (autrement dit, une constante réelle comporte au moins un point décimal ou un exposant).

Les chaînes de caractères sont délimitées par le caractère `"`, et peuvent contenir ce même caractère seulement s'il est précédé du caractère `\`. En d'autres termes, les chaînes obéissent à l'expression régulière suivante :

$$
\langle string \rangle \quad ::= \quad \texttt{"}\ ([\hat{}\ \texttt{"}] \mid \texttt{\textbackslash}\ \texttt{"})\star\ \texttt{"}
$$

Tous les identificateurs constituant des instructions (voir la syntaxe ci-dessous) sont réservés et peuvent être écrits dans une casse quelconque.

2.1.2 Syntaxe

Les programmes obéissent à la syntaxe donnée figure 1.

```
⟨code⟩        ::= ⟨instr⟩*
⟨instr⟩       ::= ⟨ident⟩ :
                | ⟨instr_atom⟩
                | ⟨instr_int⟩ ⟨integer⟩
                | pushf ⟨float⟩
                | (pushs | err) ⟨string⟩
                | check ⟨integer⟩ , ⟨integer⟩
                | (jump | jz | pusha) ⟨ident⟩
⟨instr_atom⟩  ::= add | sub | mul | div | mod | not | inf | infeq | sup
                | supeq | fadd | fsub | fmul | fdiv | fcos | fsin
                | concat | equal | atoi | atof
                | itof | ftoi | stri | strf | padd | isaddr
                | pushsp | pushfp | pushgp | loadn | storen | swap
                | writei | writef | writes | read | call | return
                | drawpoint | drawline | drawcircle | drawrect | fillrect
                | cleardrawingarea | opendrawingarea | setcolor | refresh
                | start | nop | stop | allocn | free | dupn | popn
                | get | set | move | turnleft | turnright
⟨instr_int⟩   ::= pushi | pushn | pushg | pushl | load
                | dup | pop | storel | storeg | alloc
```

FIG. 1 – Syntaxe des programmes

2.2 Utilisation

2.2.1 Machine en mode texte vm

Cette machine est destinée à un usage non-interactif, pour obtenir le résultat d'une exécution. Elle s'utilise ainsi

```
vm [options] [fichier.vm]
```

Lorsque le fichier est omis, le code est lu sur l'entrée standard. Les options sont les suivantes :

`-dump`
: affiche quelques informations sur la machine à la fin de l'exécution (valeurs des registres, sommet de la pile, etc.)

`-silent`
: exécution silencieuse : rien n'est affiché pendant l'exécution

`-count`
: affiche le nombre d'instructions exécutées

`-ssize` *entier*

  fixe la taille de la pile (10000 par défaut)

`-csize` *entier*

  fixe la taille de la pile d'appels (100 par défaut)

`-maze`

  active le labyrinthe

2.2.2 Interface graphique `gvm`

Cette machine est destinée à un usage interactif, pour visualiser l'exécution d'un programme. L'exécution peut se faire pas à pas, ou être lancée jusqu'à l'arrêt du programme, avec mise à jour des différents composants de la machine (animation) ou non (exécution non interactive). Elle s'utilise ainsi

```
gvm [options] [fichier.vm]
```

Les seules options sont `-ssize` et `-csize`, similaires à celles de la machine `vm`.

Seuls les éléments de la pile en dessous de SP sont visualisés (SP a donc pour valeur l'indice de la première ligne vide). La ligne colorée en vert correspond à FP. Les valeurs de SP, FP et GP et PC sont affichées en bas à droite. Une instruction READ déclenche l'ouverture d'une boîte de dialogue pour y saisir une chaîne de caractères, et une instruction d'affichage (WRITEI, WRITEF ou WRITES) a pour effet d'écrire dans la barre située en bas de l'interface.

2.3 Messages d'erreurs

Les différentes erreurs d'exécution possibles sont les suivantes :

`Illegal Operand`

  déclenchée lorsque la ou les valeur(s) sur la pile ne sont pas de la nature attendue

`Segmentation Fault`

  déclenchée pour tout accès à une zone illégale du code, de la pile ou de l'un des deux tas

`Stack Overflow`

  déclenchée pour toute tentative d'addition au sommet d'une pile pleine (pile d'exécution ou pile d'appels)

`Division By Zero`

  déclenchée en cas de division (entière) par zéro

`Error "`*message*`"`

  déclenchée lorsque l'instruction `err` est exécutée

`Anomaly`

  cette erreur-là ne doit normalement jamais se produire ; si c'est le cas, merci de le signaler aux enseignants, en joignant autant que possible le programme l'ayant déclenchée.