Machines Virtuelles – MV6
CM 1

Peter Habermehl
Transparents avec Michele Pagani et autres

Université Paris Cité
UFR Informatique
IRIF

Peter.Habermehl@irif.fr

1 février 2024

Organisation

Intervenant : Peter Habermehl

Horaire: Jeudi 8h30 - 10h30

Page du cours : Moodle : Machines Virtuelles

et surtout

`https://www.irif.fr/~haberm/cours/mv`

Évaluation :

$$\text{session1} = 0.5 * \text{CC} + 0.5 * \text{exam1}$$

$$\text{session2} = \max(0.5 * \text{CC} + 0.5 * \text{exam2}, 0.2 * \text{CC} + 0.8 * \text{exam2})$$

Bibliographie

Cours atypique, pas de livre “tout-en-un”

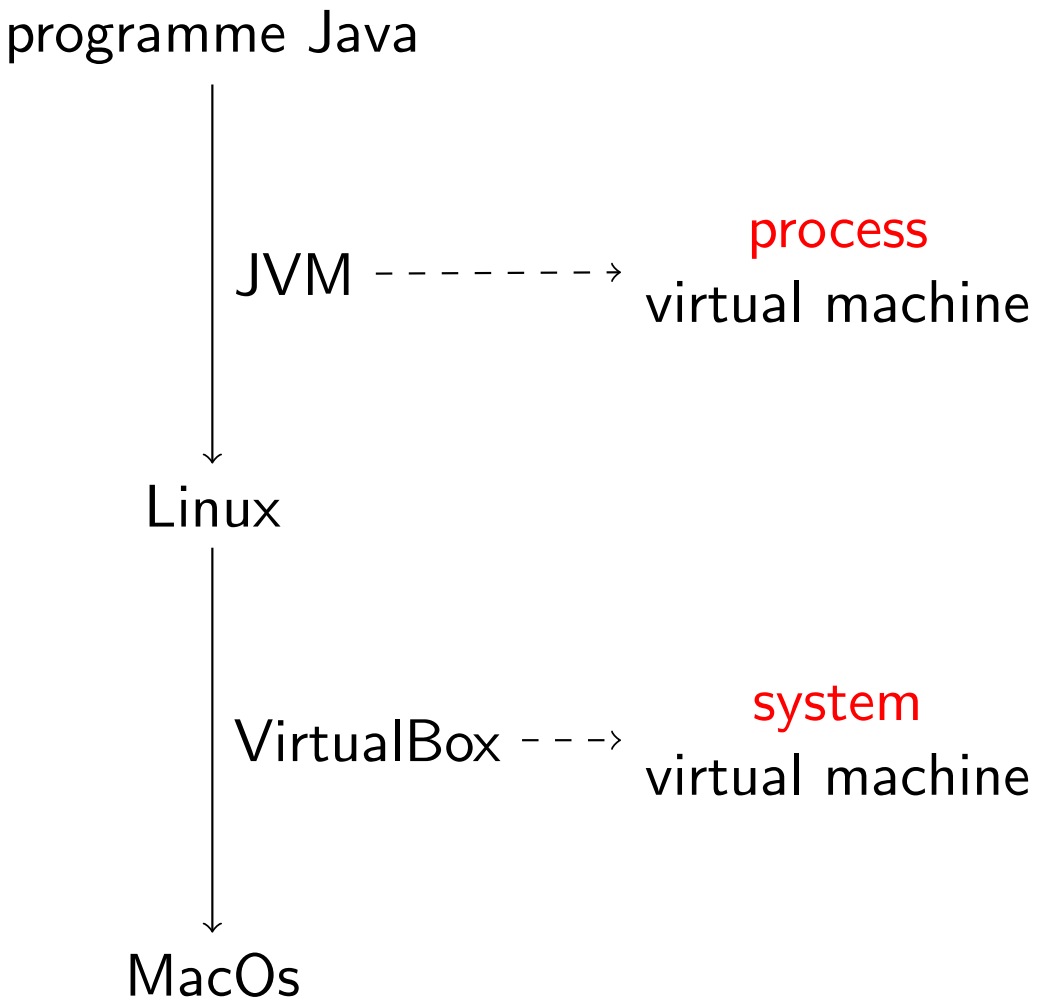
- Virtual Machines : Versatile Platforms for Systems and Processes
  (Smith, Nair, 2005)
- Développement d'applications avec Objective Caml
  (Chailloux, Manoury, Pagano, 2000)[1]
- Caml Virtual Machine – File and data format
  (Clerc, 2007)[2]
- Caml Virtual Machine – Instruction set
  (Clerc, 2010)[3]
- The Java Virtual Machine Specification
  (Lindholm, Yellin, Bracha, Buckley, Smith 2023)[4]

---

[1] https://caml.inria.fr/pub/docs/oreilly-book/html/index.html
[2] http://cadmium.x9c.fr/distrib/caml-formats.pdf
[3] http://cadmium.x9c.fr/distrib/caml-instructions.pdf
[4] https://docs.oracle.com/javase/specs/jvms/se21/jvms21.pdf

Machine Virtuelle : C'est Quoi ?

Un exemple très proche de nous



- portabilité
- nouvelles fonctionnalités (GC)
- optimisation
- sécurité

- sécurité (sandboxing)
- émulation plusieurs machines
- plusieurs OS

Exemples : il y en a partout !

Virtualisation

Émulation

Cloud computing

Langages de haut niveau

Définition informelle

Machine virtuelle

L'implémentation d'une machine comme un programme prenant un programme et émulant son execution.

Hôte. Machine sur laquelle tourne la MV.

Invité. Machine émulée.

Une machine dématérialisée, sans existence physique : ni silicium, ni engrenage, mais un programme qui exécute un programme !

Qu'est-ce qu'on gagne ?

Choix du jeu d'instructions. On n'est plus lié au jeu d'instructions du processeur : émulation, code-octet...

Choix des structures. On peut introduire dans la machine des mécanismes inexistant sur l'hôte : ramasse-miette, typage, contrats, permissions...

Contrôle de l'execution. La MV peut observer le programme avant de l'évaluer, sauver et restaurer son état : débogueur, virtualisation, "sandboxing" (bac à sable).

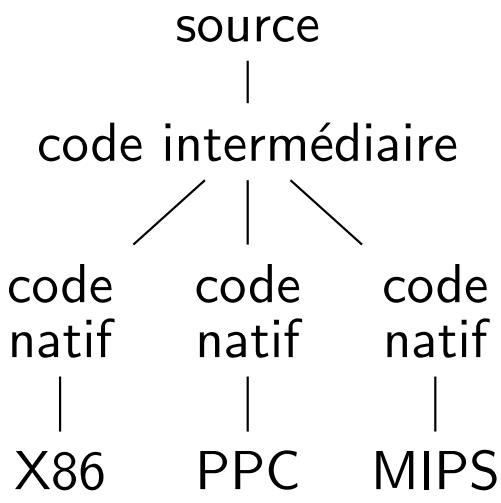
Raisonnement sur les programmes. On peut s'abstraire des détails de l'électronique : un cadre formel et universel pour comprendre, i.e. prouver des propriétés sur l'évaluation.

La MV comme donnée. Comme tout programme, la MV elle-même peut être téléchargée, mise à jour...

Des programmes portables

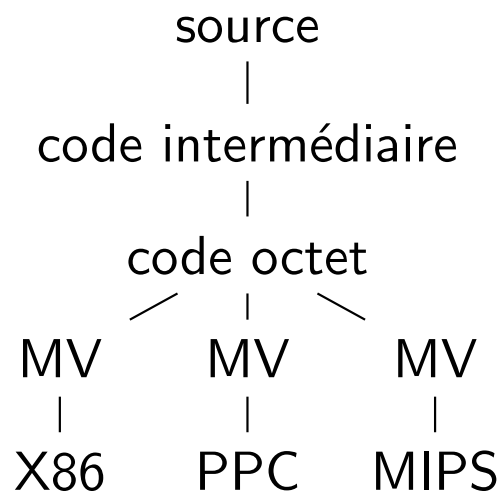
Sans machine virtuelle

- un compilateur classique génère code objet/natif pour architecture physique donnée (x86, PPC, MIPS)
- *n* architectures à prendre en charge
- *n* executables à distribuer



Avec machine virtuelle

- Ocamlc/Javac génèrent du code-octet pour une MV (Ocamlrun, JVM), qui l'interprète "traduit" en code natif
- Un seul executable distribué
- *n* portages de la MV



Définition formelle

Une machine est un couple $(S, \texttt{exec})$ avec:

- S un ensemble d'états: mémoire, registres,...
- $\texttt{exec}$ une fonction de transition

$$\texttt{exec} : S \mapsto S$$

(par ex. execution d'une instruction).

Une machine virtuelle (MV) est composée de:

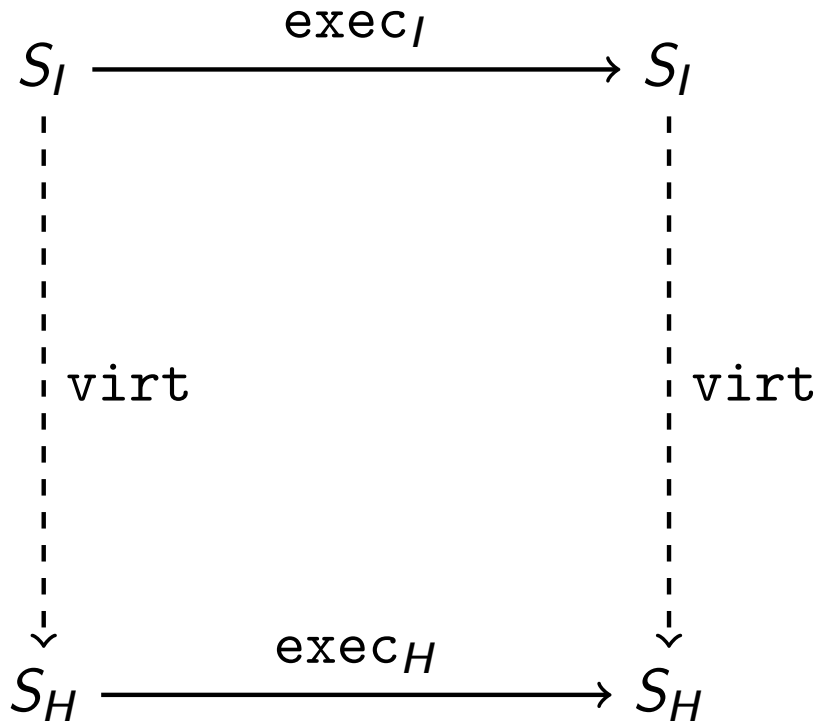
- deux machines $(S_H, \texttt{exec}_H)$ et $(S_I, \texttt{exec}_I)$,
- une fonction de virtualisation

$$\texttt{virt} : S_I \mapsto S_H$$

associant à chaque état de l'invité un état de l'hôte.

Définition formelle

Une MV doit vérifier une propriété d'émulation:



Compromis interprétation/compilation

interprétation: exécution au fur et à mesure de l'analyse
pas de pré-traitement code source

compilation: traduction code source en "langage machine"
(instructions processeur)

Avec machine virtuelle. Compilation du code source en un langage machine, puis interprétation par une MV

Remarque. La distinction n'est pas si nette:

- même les interprètes travaillent sur une forme pré-traitée du code source (par ex. arbre de syntaxe abstraite)
- les "langages interprétés" (Python, Javascript) sont souvent à MV
- les instructions processeur sont compilées en un langage de plus bas niveau (microcode)

Dans ce cours

On apprendra à concevoir et implémenter des machines virtuelles:

- coder/décoder des instructions en code-octet (assembler/désassembler)
- comprendre les machines à (a)-pile
- savoir compiler des expressions vers du code-octet
- traiter les appels de fonctions et de méthodes

Deux études de cas:

- OCamlrun, la machine virtuelle de OCaml
- JVM, la machine virtuelle de Java

Spoiler alert

L'étude des machines virtuelles n'est qu'une excuse pour introduire la compilation dans un cadre simple.

Une machine à a-pile

Modèles de machines

Modèles concrets :

- machine à pile (e.g. Postscript, LISP)
- machines à registres (RAM, RASP, pointeurs)

Une machine à pile avec quelques registres: machine à a-pile

Modèles plus abstraits :

- machines de Turing
- systèmes de réécriture de termes

Machines à a-pile

Un état de la machine est constitué de:

- une pile S
- deux registres:

  A: accumulateur

  PC: program counter

  pointeur vers la prochaine instruction
- un programme, cet à-dire tableau d'instructions C

Jeu d'instructions constituant C:

Push	empile le contenu de A sur S
Pop	dépile la tête de S et l'écrit dans A
Consti n	remplace le contenu de A par n
Addi	dépile un mot n de S, remplace A par $A + n$
Ori	dépile un mot n de S, remplace A par 0 si $A = n = 0$, par 1 sinon
Eqi	dépile un mot n de S, remplace A par 1 si $n = A$, 0 sinon

Example 1

Calculer l'execution de `Consti 3; Push; Consti 1; Addi` sur $A =$ quelque chose et $S = \epsilon$.

	PC	A	S
configuration initiale $\Rightarrow$	0	...	ϵ
	1	3	ϵ
	2	3	3
	3	1	3
configuration finale $\Rightarrow$	4	4	ϵ

Example 2

Calculer l'execution de

`Consti 3; Push; Consti 2; Push; Consti 1; Push; Addi; Addi; Addi`

sur $A =$ quelque chose et $S = \epsilon$.

Example 2

Calculer l'execution de

`Consti 3; Push; Consti 2; Push; Consti 1; Push; Addi; Addi; Addi`

sur $A =$ quelque chose et $S = \epsilon$.

PC	A	S
0	...	ϵ
1	3	ϵ
2	3	3
3	2	3
4	2	2, 3
5	1	2, 3
6	1	1, 2, 3
7	2	2, 3
8	4	3
9	7	

Exercice 1

Trouver un programme qui calcule le quintuple d'un entier, c'est-à-dire tel que l'execution sur

$$A = n \text{ et } S = \epsilon$$

termine avec:

$$A = 5n \text{ et } S = \epsilon$$

Exercice 1

Trouver un programme qui calcule le quintuple d'un entier, c'est-à-dire tel que l'execution sur

$$A = n \text{ et } S = \epsilon$$

termine avec:

$$A = 5n \text{ et } S = \epsilon$$

Solution

```
Push; Push; Push; Push; Addi; Addi; Addi; Addi
```

Exercice 1

Trouver un programme qui calcule le quintuple d'un entier, c'est-à-dire tel que l'execution sur

$$A = n \text{ et } S = \epsilon$$

termine avec:

$$A = 5n \text{ et } S = \epsilon$$

Solution

```
Push; Push; Push; Push; Addi; Addi; Addi; Addi
```

$\Longrightarrow$ et comment faire si on vaudrait calculer 32n ?

Exercice 1

Trouver un programme qui calcule le quintuple d'un entier, c'est-à-dire tel que l'execution sur

$$A = n \text{ et } S = \epsilon$$

termine avec:

$$A = 5n \text{ et } S = \epsilon$$

Solution

```
Push; Push; Push; Push; Addi; Addi; Addi; Addi
```

$\Longrightarrow$ et comment faire si on vaudrait calculer 32n ?
en dix instructions. . .

Exercice 1

Trouver un programme qui calcule le quintuple d'un entier, c'est-à-dire tel que l'execution sur

$$A = n \text{ et } S = \epsilon$$

termine avec:

$$A = 5n \text{ et } S = \epsilon$$

Solution

```
Push; Push; Push; Push; Addi; Addi; Addi; Addi
```

$\Longrightarrow$ et comment faire si on vaudrait calculer 32n ?
en dix instructions. . .

Solution

```
Push; Addi; Push; Addi; Push; Addi; Push; Addi; Push;
Addi
```

Exercice 2

Trouver un programme qui calcule la négation,
c'est-à-dire tel que l'exécution sur

$$A = n \text{ et } S = \epsilon$$

termine avec:

$$\begin{cases} A = 1 \text{ et } S = \epsilon & \text{si } n = 0, \\ A = 0 \text{ et } S = \epsilon & \text{sinon.} \end{cases}$$

Exercice 2

Trouver un programme qui calcule la négation,
c'est-à-dire tel que l'exécution sur

$$A = n \text{ et } S = \epsilon$$

termine avec:

$$\begin{cases} A = 1 \text{ et } S = \epsilon & \text{si } n = 0, \\ A = 0 \text{ et } S = \epsilon & \text{sinon.} \end{cases}$$

Solution

```
Push; Consti 0; Eqi
```

Exercice 3

Trouver un programme qui calcule l'implication $m \to n$, c'est-à-dire tel que l'exécution sur

$$A = m \text{ et } S = n$$

termine avec:

$$\begin{cases} A = 0 \text{ et } S = \epsilon & \text{si } m \neq 0 \text{ et } n = 0, \\ A = 1 \text{ et } S = \epsilon & \text{sinon.} \end{cases}$$

Exercice 3

Trouver un programme qui calcule l'implication $m \to n$, c'est-à-dire tel que l'exécution sur

$$A = m \text{ et } S = n$$

termine avec:

$$\begin{cases} A = 0 \text{ et } S = \epsilon & \text{si } m \neq 0 \text{ et } n = 0, \\ A = 1 \text{ et } S = \epsilon & \text{sinon.} \end{cases}$$

Solution

$m \to n$ est équivalent à $\neg m \vee n$, donc:

```
Push; Consti 0; Eqi; Ori
```

Vers le TD !

Est-il possible de calculer la conjunction $n \wedge m$?

Est-il possible de calculer la multiplication $n \times m$?

Vers le TD !

Est-il possible de calculer la conjunction $n \wedge m$? Non.

- $n \wedge m = \neg(\neg n \vee \neg m)$,
- il faut faire négation de n et de m,
- c.-à-d., mettre en tête d'abord n, puis m (sans effacer la valeur de n)
- pas possible avec ce jeu d'instructions d'échanger l'ordre des éléments dans la pile (par exemple, passer de $S = n, m$ à $S = m, n$).

Est-il possible de calculer la multiplication $n \times m$?

Vers le TD !

Est-il possible de calculer la conjunction $n \wedge m$? Non.

- $n \wedge m = \neg(\neg n \vee \neg m)$,
- il faut faire négation de n et de m,
- c.-à-d., mettre en tête d'abord n, puis m (sans effacer la valeur de n)
- pas possible avec ce jeu d'instructions d'échanger l'ordre des éléments dans la pile (par exemple, passer de $S = n, m$ à $S = m, n$).

Est-il possible de calculer la multiplication $n \times m$? Non.

- $n \times m = m + \cdots + m$ (n fois),
- répéter une suite d'operations un nombre de fois dépendant du contenu de l'état de la machine,
- pas possible avec ce jeu d'instructions de faire des sauts conditionnels.

Vers le TD !

Est-il possible de calculer la conjunction $n \wedge m$? Non.

- $n \wedge m = \neg(\neg n \vee \neg m)$,
- il faut faire négation de n et de m,
- c.-à-d., mettre en tête d'abord n, puis m (sans effacer la valeur de n)
- pas possible avec ce jeu d'instructions d'échanger l'ordre des éléments dans la pile (par exemple, passer de $S = n, m$ à $S = m, n$).

Est-il possible de calculer la multiplication $n \times m$? Non.

- $n \times m = m + \cdots + m$ (n fois),
- répéter une suite d'operations un nombre de fois dépendant du contenu de l'état de la machine,
- pas possible avec ce jeu d'instructions de faire des sauts conditionnels.

On éteindra le jeu d'instructions pour permettre tout cela, dans le sujet de TD !