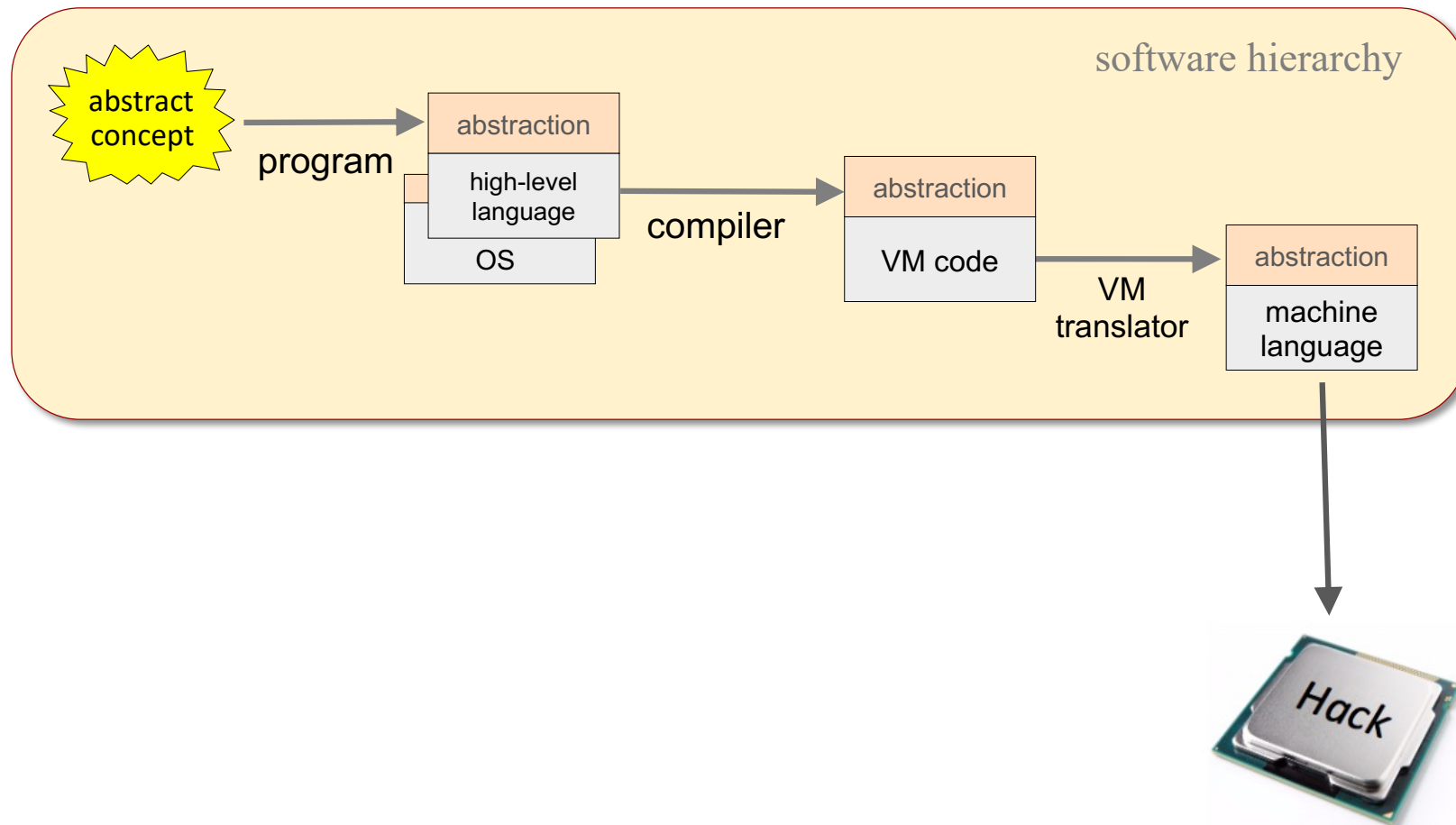
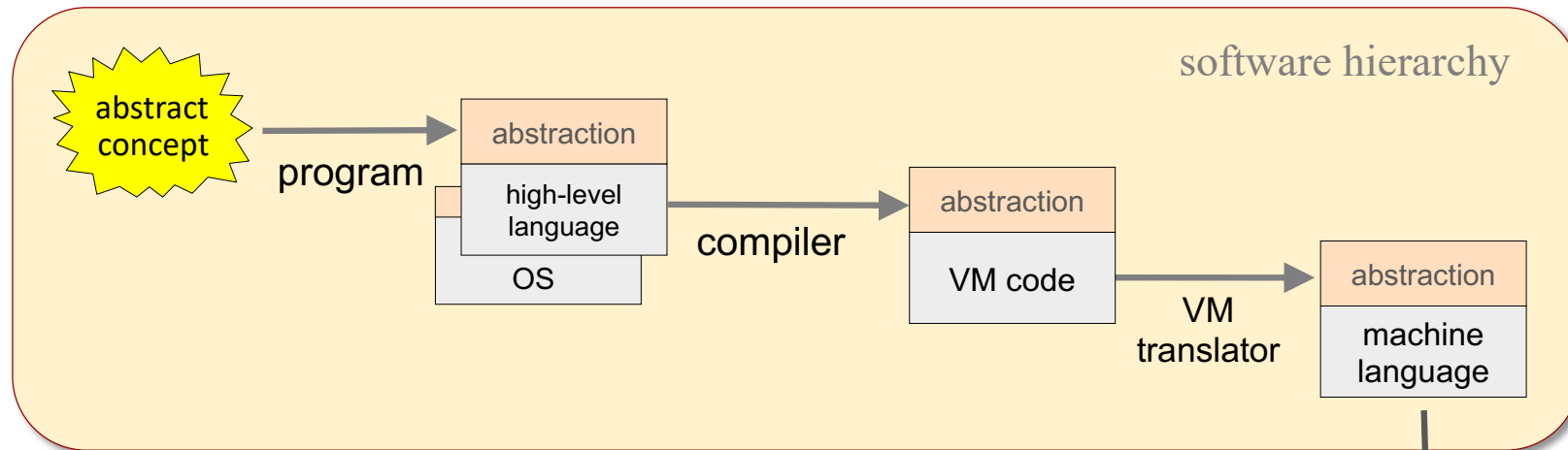


2. Machine Virtuelle

Introduction



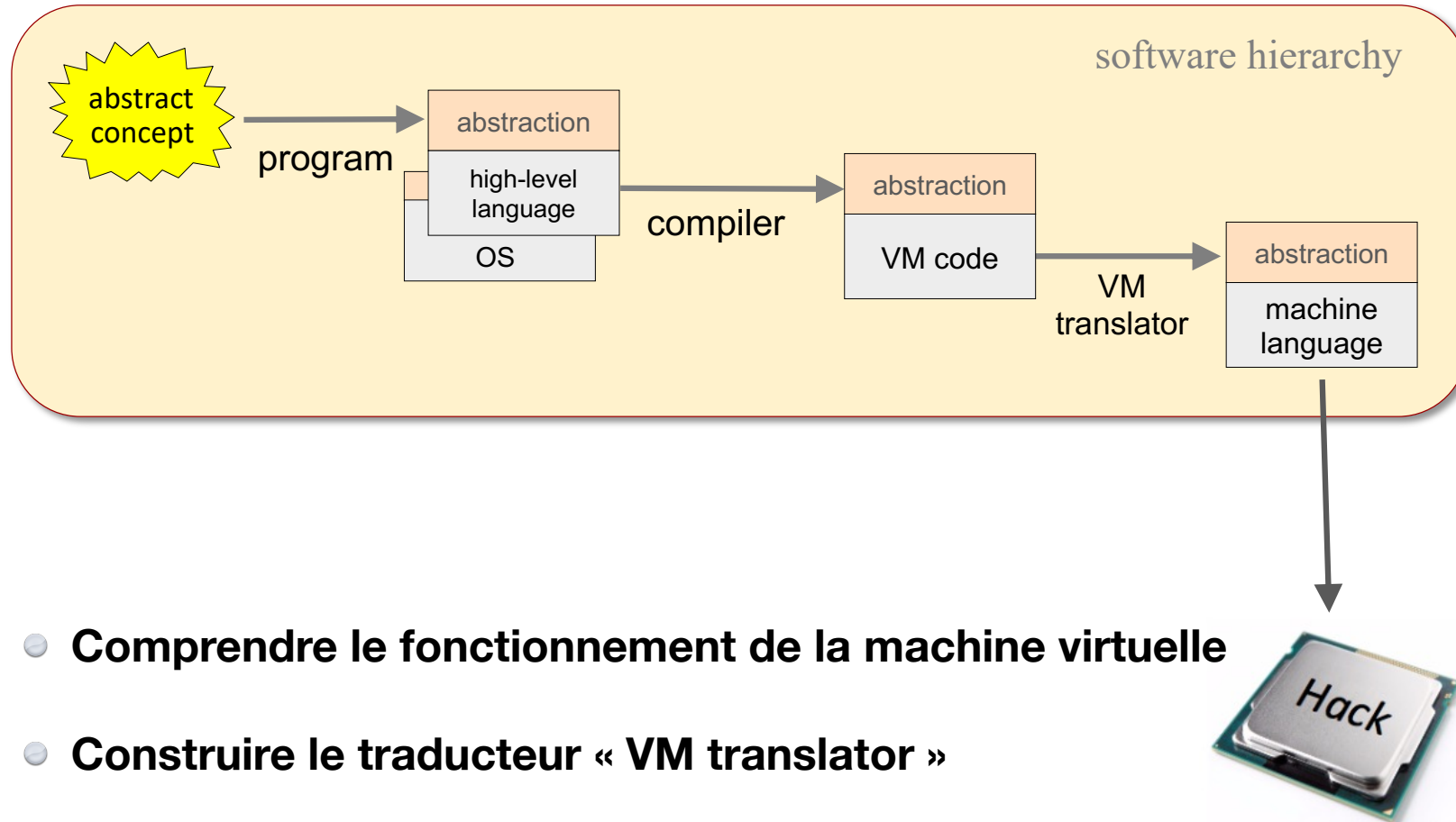
Rôle de la machine virtuelle



- Le « VM Code » généré par le compilateur est conçu pour s'exécuter sur une machine abstraite à base de pile.
- Le traducteur « VM translator » traduit ce code en langage assembleur.



Plan du cours



La machine à pile Bob - exercice 6 TP1

Question 6 Nous considérons le langage Bob opérant juste sur une pile et une RAM. Un programme Bob est composé d'une suite élémentaire d'opérations respectant les règles suivantes :

- *val* avec *val* un entier : *val* est empilé
- *+* : additionne les deux éléments en haut de la pile en les dépilant puis empile le résultat.
- *=* : teste l'égalité des deux éléments en haut de la pile en les dépilant puis empile le résultat.
- (*label*) avec *label* un mots commençant par une lettre défini une étiquette sur l'instruction suivante
- *label* empile l'étiquette
- *< -* : lire la valeur *val* en haut de la pile et la remplace par $RAM[val]$
- *- >* : lire deux valeurs *val1*, *val2* en haut de la pile en les dépilant et range la valeur *val2* dans la RAM à l'adresse *val1*. A noter que *val1* est la valeur en haut de la pile et *val2* la deuxième.
- *JEQ* : lire deux valeurs *val1*, *val2* en haut de la pile en les dépilant et réalise un saut à l'étiquette *val1* si *val2* vaut 0.
- ajouter les autres opérations qui vous semblent utiles (i.e. des opérateurs de saut et d'autres tests que l'égalité!)

Ecrire un Python qui prend un code Bob et renvoie un code Assembleur. Un prototype du programme (*tp1bob.py*) est disponible sur Celene.

Le langage de la machine virtuelle

Push / pop commands

- `push segment i`
- `pop segment i`

Arithmetic / Logical commands

- `add, sub, neg`
- `eq, gt, lt`
- `and, or, not`

Branching commands

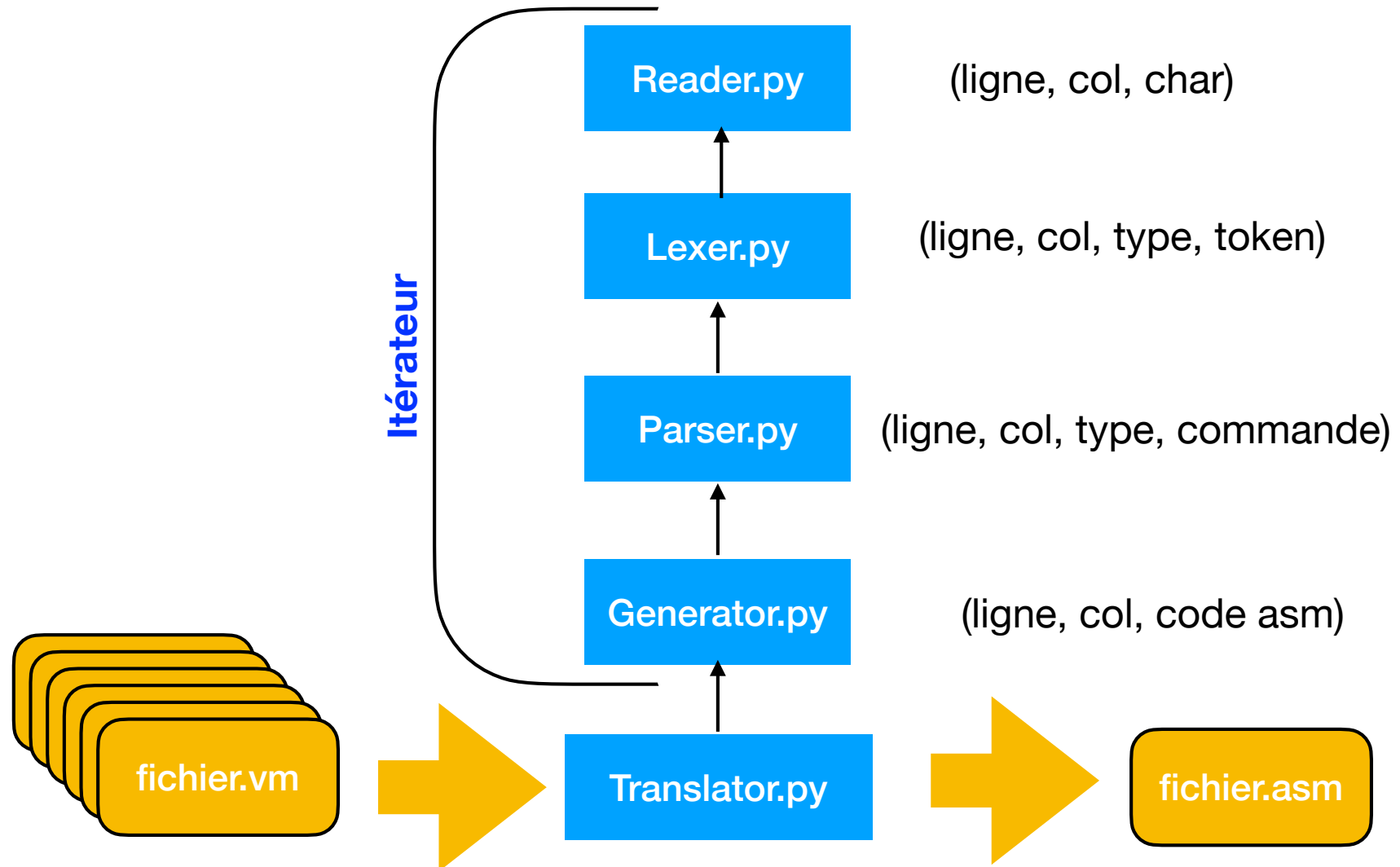
- `label label`
- `goto label`
- `if-goto label`

Function commands

- `Function functionName nVars`
- `Call functionName nArgs`
- `return`

où *i* est un entier positif et *segment* est un mot des mots clés suivant :
local, argument, static, constant, this, that, pointer, temp.

Le traducteur « VM translator »



Démo !

Quoi faire !

- Dès que le dépôt git est prêt, déposer un projet comprenant
 - *un readme.md à la racine*
 - *un répertoire VMTranslator*
 - *dans ce répertoire, un readme.md et le code python*
- Ajouter dans VMTranslator, un répertoire test et préparer des fichiers d'exemples de code vm
- Lire et tester le code *Lecteur.py* et *Lexer.py*
- Ecrire le code de *Parser.py*
- Ecrire le code de *Generator.py* en renvoyant juste le texte le commande en attendant de savoir générer le code asm
- Ecrire le code de *Translator.py*