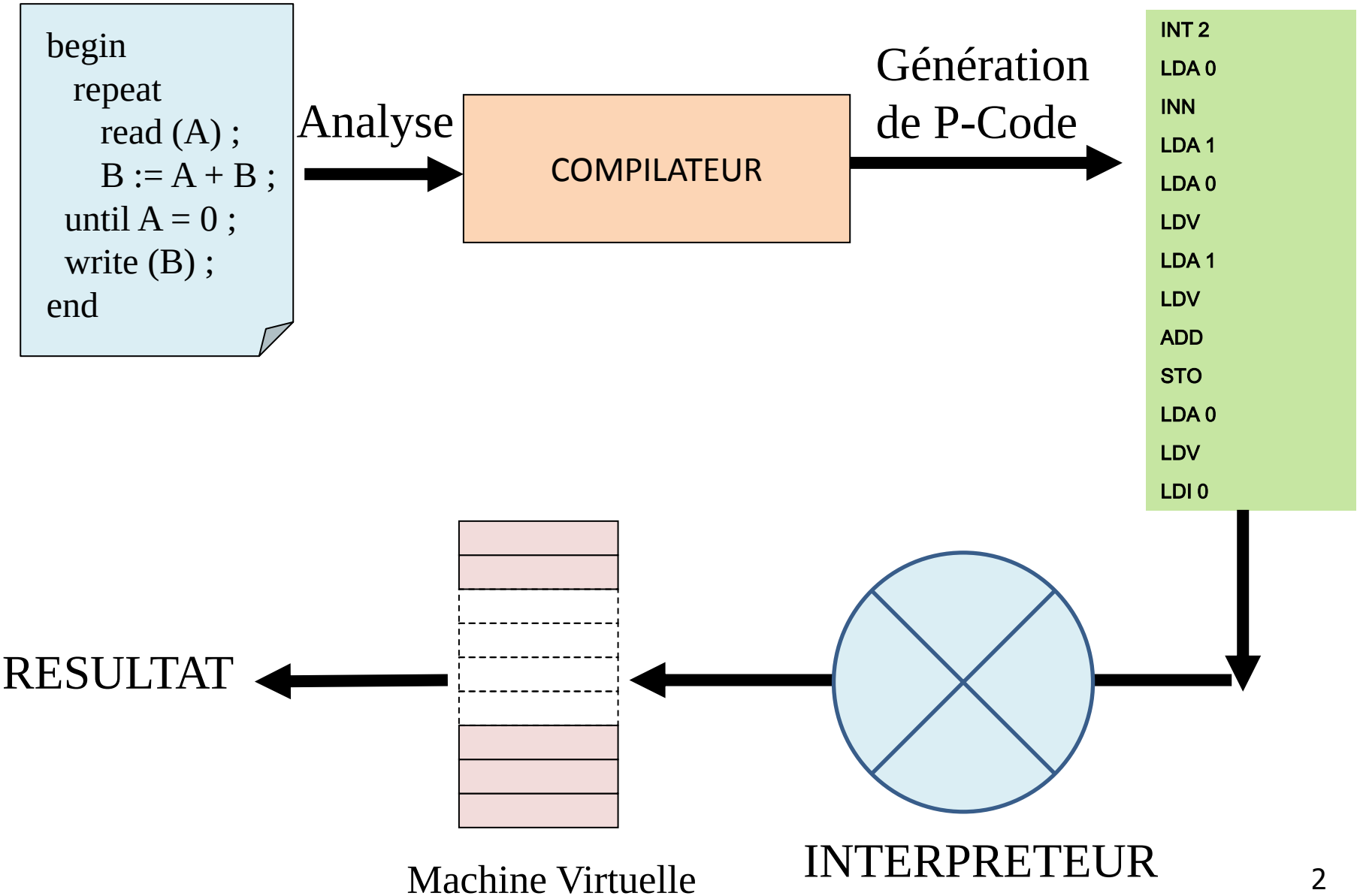
Chap. 4 : Le langage P-Code

Principe des traitements sémantiques



Principe des règles sémantiques

- Règle AFFEC

AFFEC ::= ID := EXPR

LE RESULTAT DE EXPR AU SOMMET DE LA PILE
STOCKER LE SOMMET DE LA PILE A L'ADRESSE MÉMOIRE DU ID

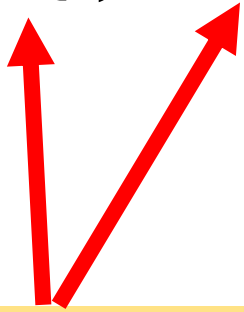
CHARGER L'ADRESSE MEMOIRE DU
ID AU SOMMET DE LA PILE:
RECUPERER ADR DE ID
EMPILER ADR

ACTIONS SEMANTIQUES

Principe des règles sémantiques

- Règle VARS

VARS::= var ID { , ID } ; | ε



RESERVATION D'UNE PLACE MEMOIRE
Actions:
INCREMENTER LE COMPTEUR DE MEMOIRE
stocker l'adresse réservée dans la table des symboles

Jeu d'instructions du P-Code simplifié

ADD	additionne le sous-sommet de pile et le sommet, laisse le résultat au sommet (idem pour SUB, MUL, DIV)
EQL	laisse 1 au sommet de pile si sous-sommet = sommet, 0 sinon (idem pour NEQ, GTR, LSS, GEQ, LEQ)
PRN	imprime le sommet, dépile
INN	lit un entier, le stocke à l'adresse trouvée au sommet de pile, dépile
INT c	incrémente de la constante c le pointeur de pile (la constante c peut être négative)
LDI v	empile la valeur v
LDA a	empile l'adresse a
LDV	remplace le sommet par la valeur trouvée à l'adresse indiquée par le sommet (déréférence)
STO	stocke la valeur au sommet à l'adresse indiquée par le sous-sommet, dépile 2 fois
BRN i	branchement inconditionnel à l'instruction i
BZE i	branchement à l'instruction i si le sommet = 0, dépile
HLT	halte

Exemple d'exécution du code généré sur la machine virtuelle

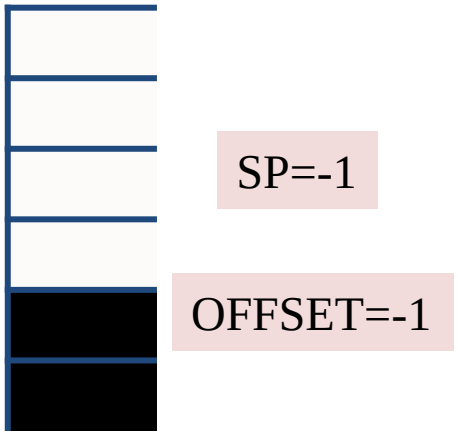
```
program exemple;
const tata=12;
var x;
begin
        x:=tata+15;
        write(x)
end
```

Analyse → COMPILATEUR → Génération de P-Code →

```
INT     1
LDA     0
LDI     12
STO
LDA     1
LDA     0
LDV
LDI     15
ADD
STO
LDA     1
LDV
PRN
HLT
```

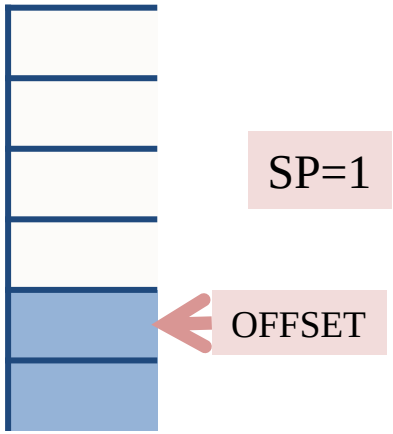
INT	1
LDA	0
LDI	12
STO	
LDA	1
LDA	0
LDV	
LDI	15
ADD	
STO	
LDA	1
LDV	
PRN	
HLT	

PC=-1



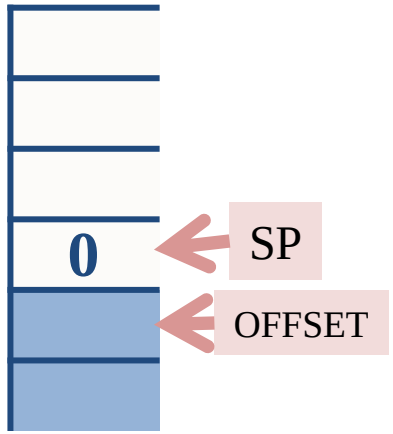
INT	1
LDA	0
LDI	12
STO	
LDA	1
LDA	0
LDV	
LDI	15
ADD	
STO	
LDA	1
LDV	
PRN	
HLT	

PC



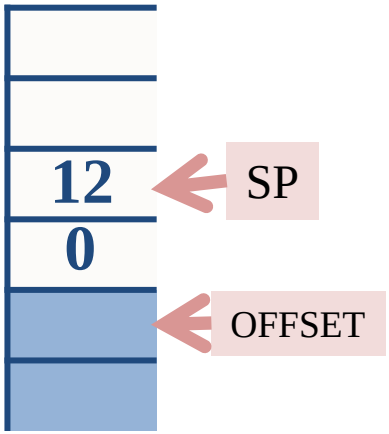
INT	1
LDA	0
LDI	12
STO	
LDA	1
LDA	0
LDV	
LDI	15
ADD	
STO	
LDA	1
LDV	
PRN	
HLT	

PC



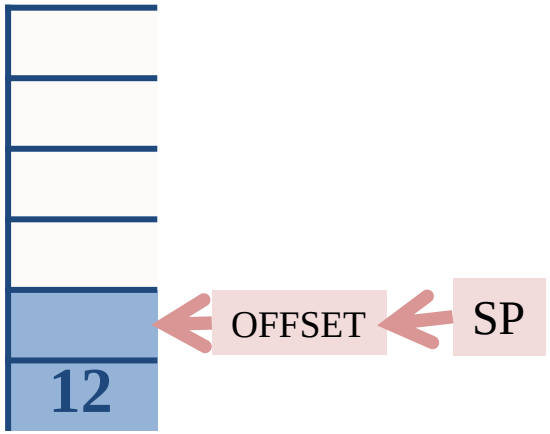
```
INT      1
LDA      0
LDI     12
STO
LDA      1
LDA      0
LDV
LDI     15
ADD
STO
LDA      1
LDV
PRN
HLT
```

PC



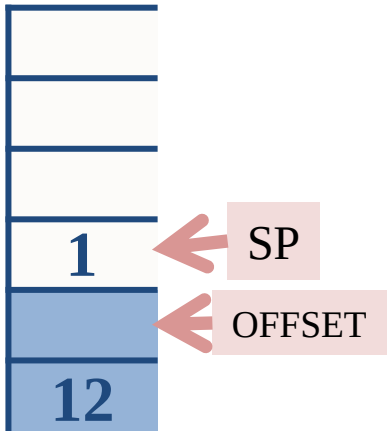

```
INT      1
LDA      0
LDI     12
STO
LDA      1
LDA      0
LDV
LDI     15
ADD
STO
LDA      1
LDV
PRN
HLT
```

PC



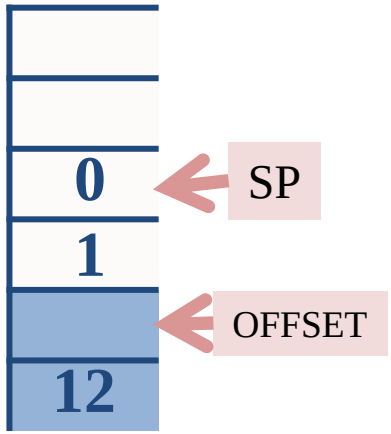

```
INT      1
LDA      0
LDI     12
STO
LDA      1
LDA      0
LDV
LDI     15
ADD
STO
LDA      1
LDV
PRN
HLT
```

PC



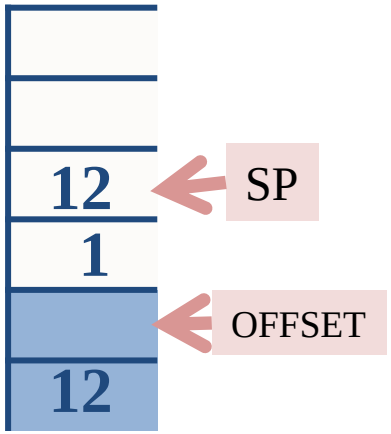
```
INT     1
LDA     0
LDI     12
STO
LDA     1
LDA     0
LDV
LDI     15
ADD
STO
LDA     1
LDV
PRN
HLT
```

PC



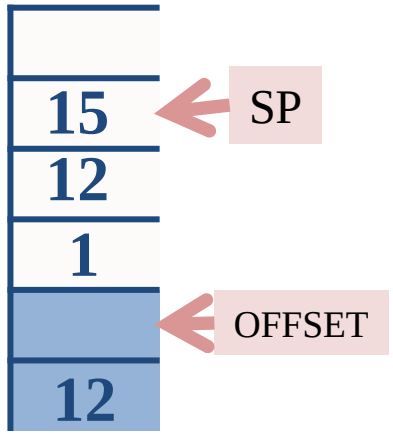

```
INT     1
LDA     0
LDI     12
STO
LDA     1
LDA     0
LDV
LDI     15
ADD
STO
LDA     1
LDV
PRN
HLT
```

PC



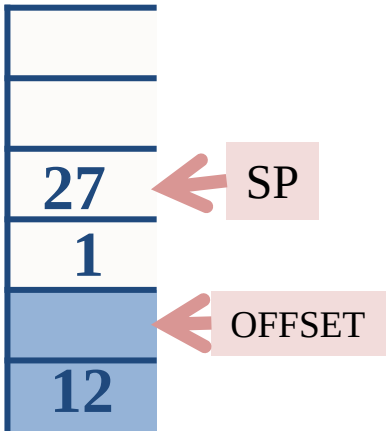

```
INT     1
LDA     0
LDI     12
STO
LDA     1
LDA     0
LDV
LDI     15
ADD
STO
LDA     1
LDV
PRN
HLT
```

PC



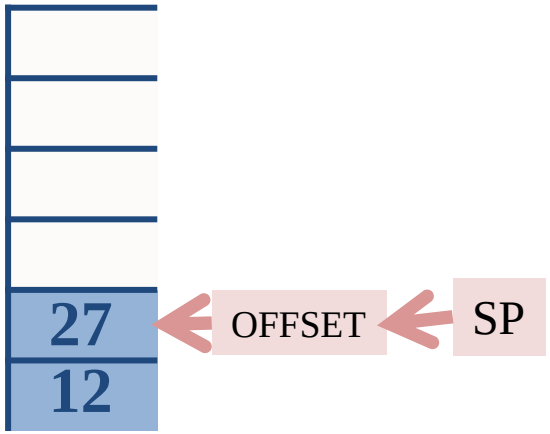
```
INT     1
LDA     0
LDI     12
STO
LDA     1
LDA     0
LDV
LDI     15
ADD
STO
LDA     1
LDV
PRN
HLT
```

PC



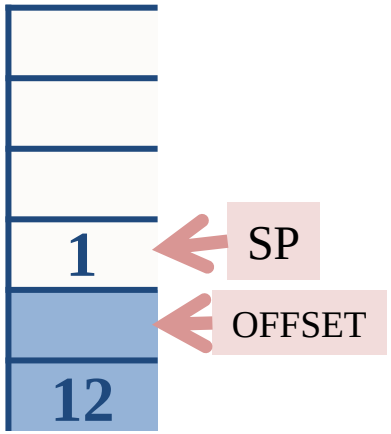

```
INT     1
LDA     0
LDI     12
STO
LDA     1
LDA     0
LDV
LDI     15
ADD
STO
LDA     1
LDV
PRN
HLT
```

PC



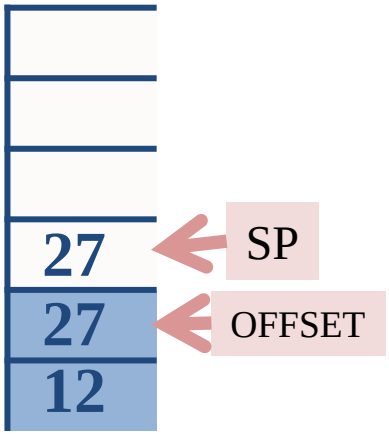

```
INT     1
LDA     0
LDI     12
STO
LDA     1
LDA     0
LDV
LDI     15
ADD
STO
LDA     1
LDV
PRN
HLT
```

PC



```
INT     1
LDA     0
LDI     12
STO
LDA     1
LDA     0
LDV
LDI     15
ADD
STO
LDA     1
LDV
PRN
HLT
```

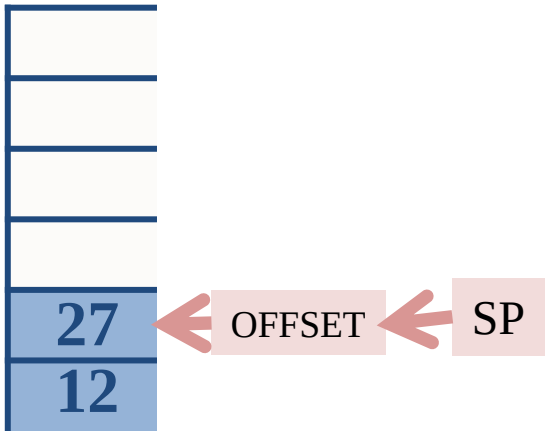
PC




```
INT     1
LDA     0
LDI     12
STO
LDA     1
LDA     0
LDV
LDI     15
ADD
STO
LDA     1
LDV
PRN
HLT
```

PC

> 27




```
INT     1
LDA     0
LDI     12
STO
LDA     1
LDA     0
LDV
LDI     15
ADD
STO
LDA     1
LDV
PRN
HLT
```

PC

Exercice 1

- Donner le P-Code généré pour le programme suivant :

```
program test;
const tata=13;
var x, y;
begin
x:=10;
y:=x+tata;
end.
```

Exercice 2

- Donner le P-Code généré pour le programme suivant :

```
program test;
const tata=13; titi=3;
var x, y;
begin
x:=10;
y:=x+tata-titi;
write(x, tata+titi);
end.
```

Exercice 3

- Donner le P-Code généré pour le programme suivant :

```
program test;
const tata=13; titi=3;
var x, y;
begin
x:=10;
y:=x+tata*titi;
end.
```