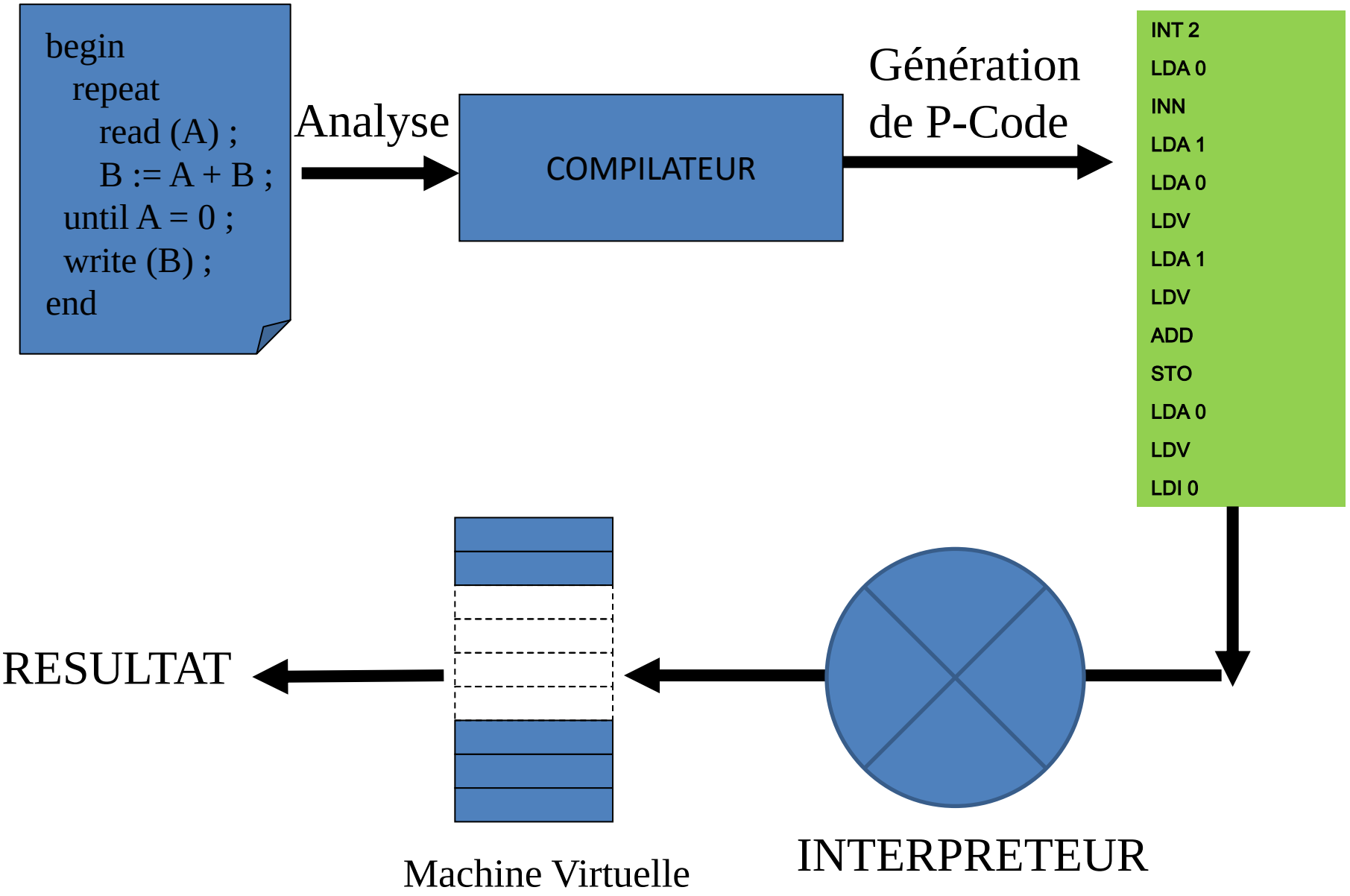
ECRITURE D'UN MINI COMPILATEUR

ANALYSEUR SYNTAXIQUE PRINCIPE

PROGRAM ::= **program** ID **;** BLOCK **.**
BLOCK ::= CONSTS VARS INSTS
CONSTS ::= **const** ID = NUM **;** { ID = NUM **;** } | **ε**
VARS ::= **var** ID { **,** ID } **;** | **ε**
INSTS ::= **begin** INST { **;** INST } **end**
INST ::= INSTS | AFFEC | SI | TANTQUE | ECRIRE | LIRE | ε
AFFEC ::= ID **:=** EXPR
SI ::= **if** COND **then** INST
TANTQUE ::= **while** COND **do** INST
ECRIRE ::= **write (** EXPR { **,** EXPR } **)**
LIRE ::= **read (** ID { **,** ID } **)**
COND ::= EXPR RELOP EXPR
RELOP ::= = | <> | < | > | <= | >=
EXPR ::= TERM { ADDOP TERM }
ADDOP ::= + | -
TERM ::= FACT { MULOP FACT }
MULOP ::= * | /
FACT ::= ID | NUM | **(** EXPR **)**

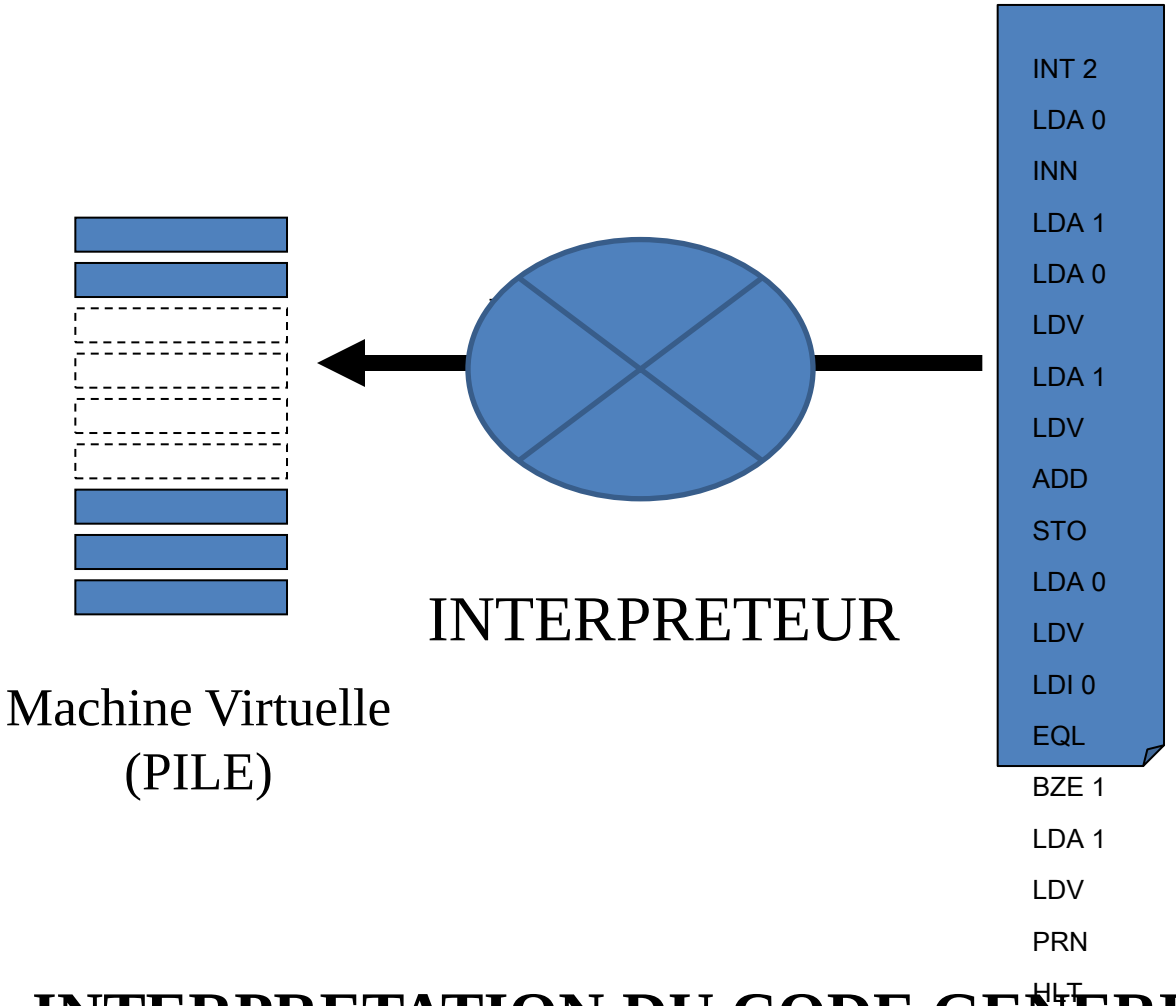
PRINCIPE DE TRAIMENTS SEMANTIQUE


begin
repeat
read (A) ;
B := A + B ;
until A = 0 ;
write (B) ;
end
Analyse
COMPILATEUR
Génération
de P-Code
INT 2
LDA 0
INN
LDA 1
LDA 0
LDV
LDA 1
LDV
ADD
STO
LDA 0
LDV
LDI 0
RESULTAT
Machine Virtuelle
INTERPRETEUR

```
Program exemple;
Const tata=12;titi=23;
Var x, y;
begin
        X:=tata;
        Y:=titi+tata;
        Write(x,y)
end
```

Analyse → COMPILATEUR → Génération de P-Code →

```
INT     3
LDA     0
LDI     12
STO
LDA     1
LDI     23
STO
LDA     2
LDA     0
LDV
STO
LDA     2
LDA     1
LDV
LDA     0
LDV
ADD
STO
LDA     2
LDV
PRN
LDA     3
LDV
PRN
HLT
```



2ième PARTIE: INTERPRETATION DU CODE GENERE

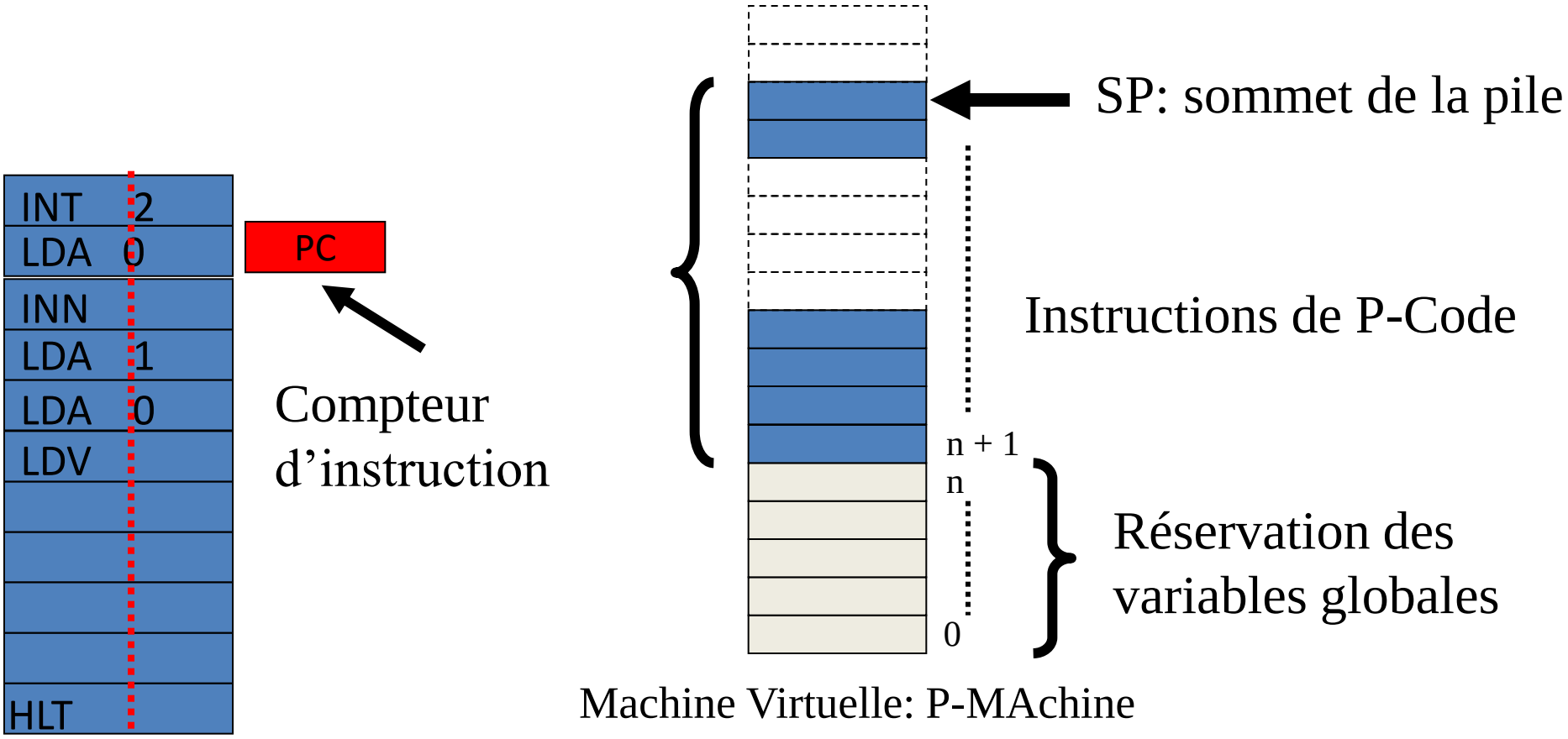
JEU DE CODE MACHINE

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

jeu d'instruction du P-Code simplifié

ENVIRONNEMENT D'EXECUTION ET GENERATION DE CODE

Le P-Code est le langage intermédiaire utilisé pour le Pascal.
Il est associé à la machine abstraite P-Machine composée de:

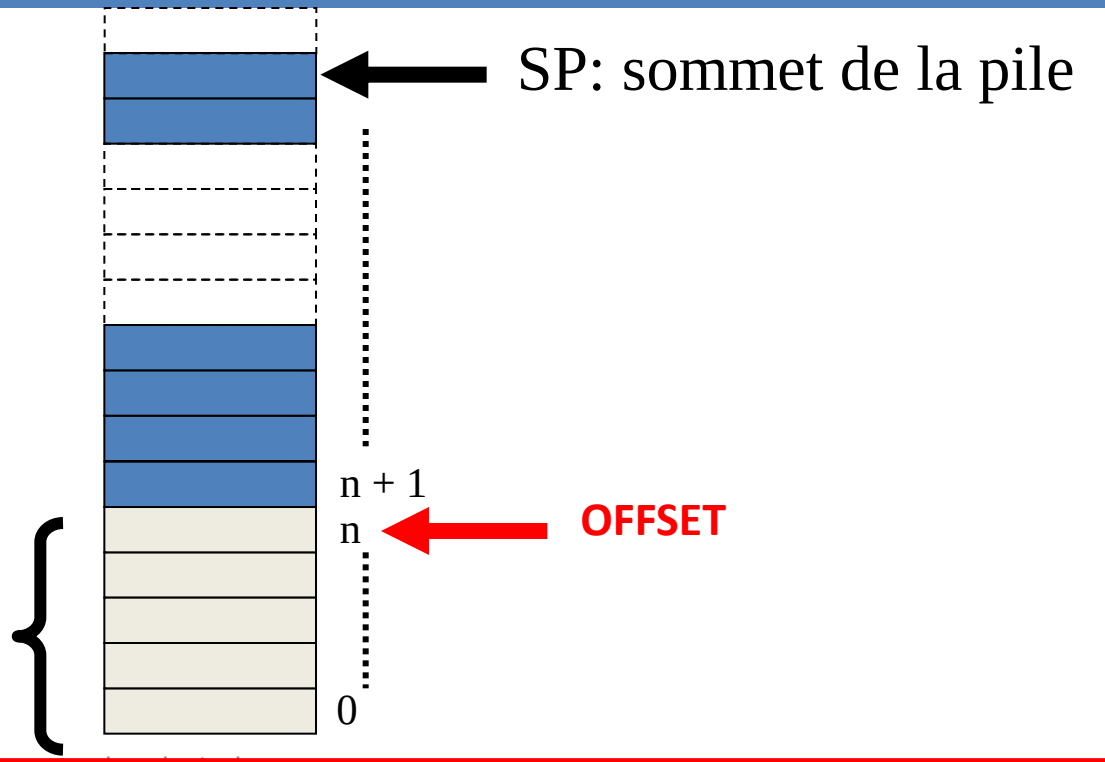


ENVIRONNEMENT

LA MEMOIRE

Table des symboles

```
var
TABLESYM : tableau [TABLEINDEX] de enregistrement
                                        NOM : ALFA ;
                                        CODE : CLASSES ;
                                        TYPSYM : TypeSymbole;
                                        ADRESSE : ENTIER
                                fin ;
OFFSET : ENTIER ;
```



ENVIRONNEMENT

LES DECLARATIONS

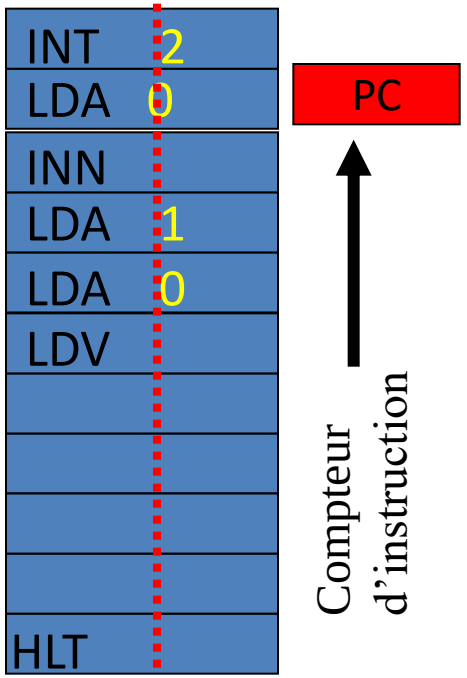
Les structures de données nécessaires lors de l'écriture d'un interprète simplifié pour le P-Code sont :

un tableau PCODE représentant les instructions de P-Code et le compteur associé PC

```
Type MNEMONIQUES = (ADD,SUB,MUL,DIV,EQL,
                    NEQ,GTR,LSS,GEQ,LEQ, PRN,
                    INN,INT,LDI,LDA,LDV,STO,BRN,
                    BZE,HLT) ;
```

```
INSTRUCTION = enregistrement
                    MNE : MNEMONIQUES ;
                    SUITE : entier
              fin
VAR PCODE : tableau [0 .. TAILLECODE] de INSTRUCTION ;
      PC=0 : entier ;
```

```
VAR OFFSET=-1;
```



EXEMPLE D'EXECUTION DU CODE GENERE SUR LA MACHINE VURTUELLE

Exercice 1: Donnez le PCODE généré pour cet exemple

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

Exercice 2: Donnez le PCODE généré pour cet exemple

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

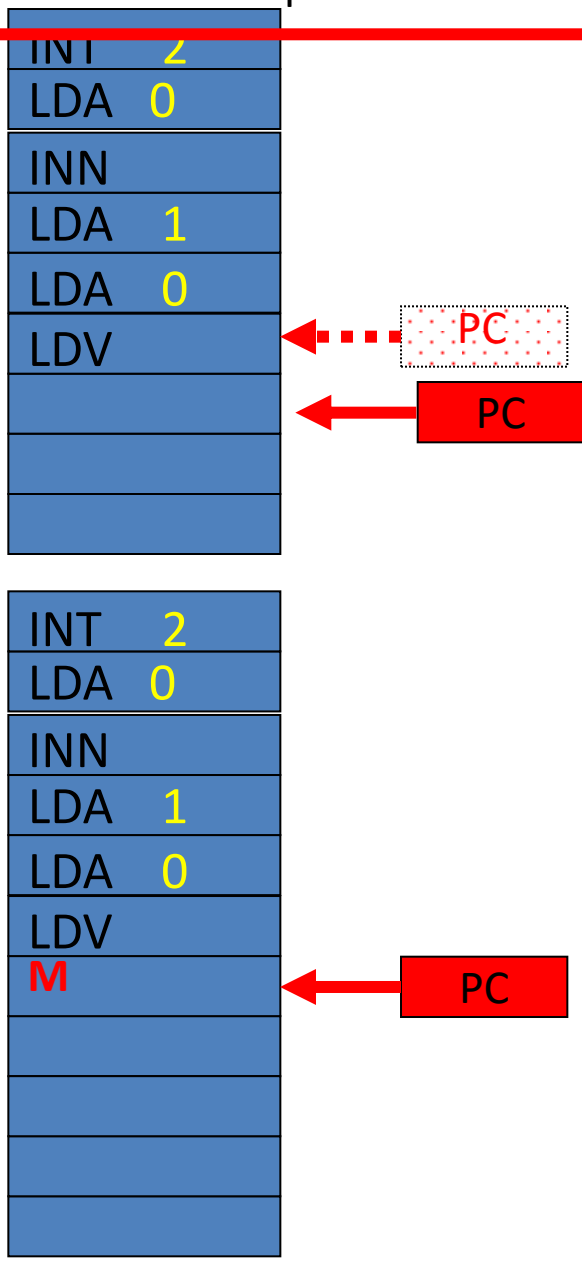
Exercice 3: Donnez le PCODE généré pour cet exemple

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

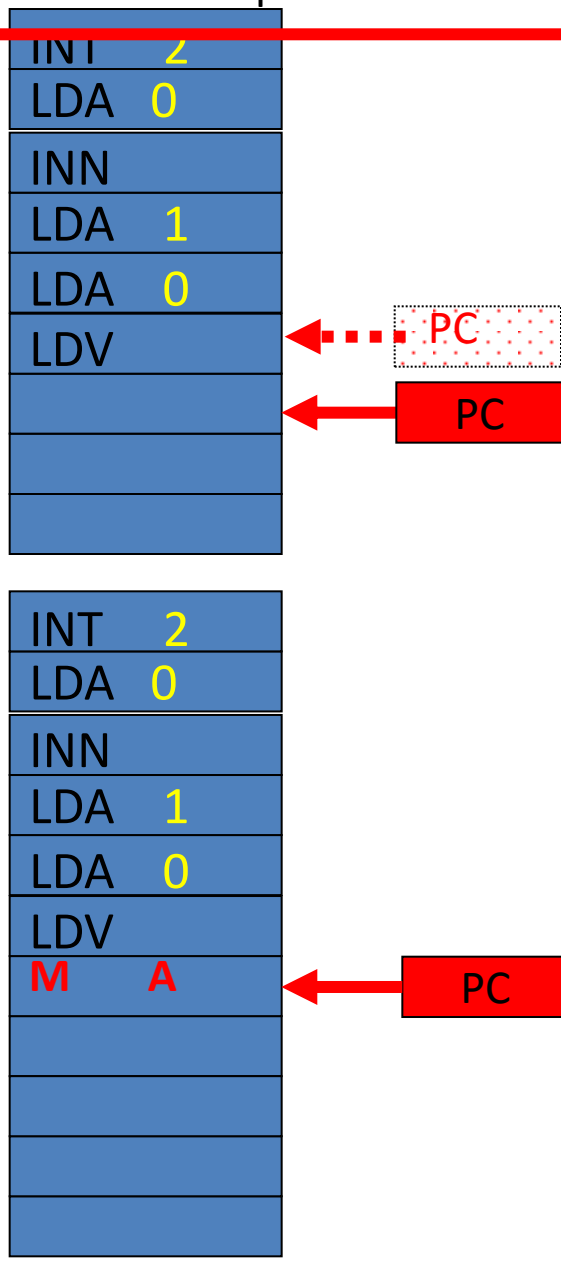
LES FONCTIONS DE GENERATION DE CODE

Procédures de génération de P-Code

```
procedure GENERER1 (M:MNEMONIQUES) ;
debut
        si  PC = TAILLECODE
                alors ERREUR ;
        PC := PC + 1 ;
        PCODE [PC]. MNE := M
fin ;
```



```
procedure GENERER2 (M:MNEMONIQUES ; A:entier) ;
debut
        si  PC = TAILLECODE
                alors ERREUR ;
        PC := PC + 1 ;
        PCODE [PC].MNE := M ;
        PCODE [PC].SUITE := A
fin;
```



A VOS MACHINES
et
BON COURAGE