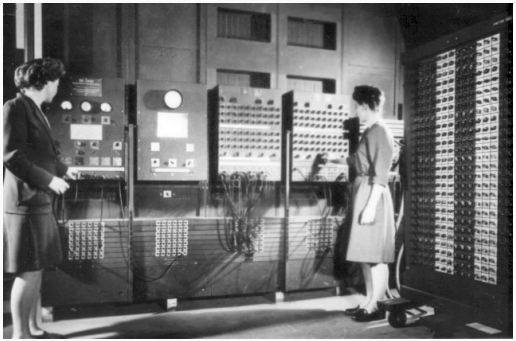
# [CI2] Cours 4: Traduction de programmes

Daniela Petrișan
Université de Paris, IRIF

# Il y a 75 ans, bien avant Java et Python



Deux programmeuses de l'ENIAC,
le premier ordinateur entièrement électronique

# De nos jours

La compilation permet de traduire les structures de contrôle du langage source de haut niveau en séquences d'instructions de bas niveau (instructions en langage machine).

Langages de haut niveau

Langage machine

# En Java

- le code source : le fichier `.java` qui représente le programme écrit en syntaxe java
- le bytecode : la version binaire du code que toutes les JVM peuvent charger et exécuter (on l'obtient en utilisant la commande `javac`)
- le code natif : le langage machine spécifique à la plate-forme actuelle (OS, CPU) qui représente le programme sous une forme que le CPU peut exécuter directement

# Le langage d'assemblage

L'assembleur autorise le programmeur à écrire les instructions du micro-processeur sous forme symbolique : il est plus facile de manipuler les noms de registres plutôt que les adresses mémoire écrites en hexadécimal.

```java
class Fibonacci{
  public static void main(String[] a){
    int u=0, v=1, w, k=12;
    for(int i=0; i<k; i++){
      w=u+v;
      u=v;
      v=w;
    }
    System.out.println("fibo("+k+")="+v);
  }
}
```

```
.ORIG x0000
AND R0, R0, #0     ;R0 <- 0
AND R1, R1, #0     ;R1 <- 0
AND R2, R2, #0     ;R2 <- 0
ADD R0, R0, #12    ;R0 <- 12
ADD R2, R2, #1     ;R2 <- 1
ADD R3, R1, R2     ;R3 <- R1 + R2 (inst#5)
ADD R1, R2, #0     ;R1 <- R2
ADD R2, R3, #0     ;R2 <- R3
ADD R0, R0, #-1    ;R0–
BRp #-5            ;(inst#9)BRanchement conditionnel :
                   ;si R0>0 alors on branche
                   ; vers instruction -5

.END
```

# Traductions de programmes

Nous n'étudierons pas de langage d'assemblage ici !

Nous allons néanmoins traduire des programmes Java (plus ou moins) complexes en un autre programme Java, mais dont l'exécution ressemble en grandes lignes à celle d'un programme en langage machine.

Nous utiliserons notamment une variable locale `ic` ou `instructionCourrante` pour modéliser le compteur ordinal, qui contient le numéro de l'instruction en cours d'exécution.

# Structure générale de la traduction d'un programme

```
class Traduction{
  public static void main(String[] a){
    int ic=0;                    //compteur d'instructions
    int[] mem=new int[100000];   //mémoire à plat
    boolean fin = false; //test pour sortie
    while(!fin) {
      switch(ic) {
        case 0: instruction atomique; ic++; break;
        case 1: instruction atomique; ic+=...; break;
                // dans les instructions atomiques,
        ...     // il est interdit de modifier ic.

        case n: fin = true;
      }
    }
  }
}
```

# L'analyse du programme Fibonacci

```
class Fibonacci{
  public static void main(String[] a){
    int u=0, v=1, w, k=12;
    for(int i=0; i<k; i++){
      w=u+v;
      u=v;
      v=w;
    }
    System.out.println("fibo("+k+")="+v);
  }
}
```

```
class Fibonacci2{
  public static void main(String[] a){
    int u=0;      // ic = 0
    int v=1;      // ic = 1
    int w;
    int k=12;     // ic = 2
    for(int i=0; // ic = 3
        i<k;      // ic = 4; saut conditionnel
        i++) {    // ic = 8
      w=u+v;      // ic = 5
      u=v;        // ic = 6
      v=w;        // ic = 7
        }         // ic = 9; saut inconditionne
                  //         vers 4
     System.out.println("fibo("+k+")="+v);
                  // ic =10
  }  // ic = 11; sortie
}
```

# Vers une traduction : déconstruire la boucle

```
class Fibonacci2{
  public static void main(String[] a){
    int u=0;      // ic = 0
    int v=1;      // ic = 1
    int w;
    int k=12;     // ic = 2
    for(int i=0; // ic = 3
        i<k;      // ic = 4; saut conditionnel
        i++) {    // ic = 8
      w=u+v;      // ic = 5
      u=v;        // ic = 6
      v=w;        // ic = 7
        }         // ic = 9; saut inconditionnel
                  //         vers 4
     System.out.println("fibo("+k+")="+v);
                  // ic =10
  }  // ic = 11; sortie
```

```
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
```

# Vers une traduction : déconstruire la boucle

```
class Fibonacci2{
  public static void main(String[] a){
    int u=0;      // ic = 0
    int v=1;      // ic = 1
    int w;
    int k=12;     // ic = 2
    for(int i=0; // ic = 3
        i<k;      // ic = 4; saut conditionnel
        i++) {    // ic = 8
      w=u+v;      // ic = 5
      u=v;        // ic = 6
      v=w;        // ic = 7
        }         // ic = 9; saut inconditionnel
                  //         vers 4
     System.out.println("fibo("+k+")="+v);
                  // ic =10
  }  // ic = 11; sortie
```

```
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut inconditionn
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
```

```
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```
class FibonacciTrad{
  public static void main(String[] a){
    int ic=0;    //compteur d'instructions
    int[] mem=new int[100000];  //mémoire à plat
    boolean fin = false;        //test pour sortie
    while(!fin) {
       switch(ic){
       case 0:
       case 1:
       case 2:
       case 3:
       case 4:
       case 5:
       case 6:
       case 7:
       case 8:
       case 9:
       case 10:
       case 11:
} } } }
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
  public static void main(String[] a){
    int ic=0;    //compteur d'instructions
    int[] mem=new int[100000];  //mémoire à plat
    boolean fin = false;        //test pour sortie
    while(!fin) {
      switch(ic){
      case 0: mem[0]=0; ic++; break;
      case 1:
      case 2:
      case 3:
      case 4:
      case 5:
      case 6:
      case 7:
      case 8:
      case 9:
      case 10:
      case 11:
} } } }
```

```
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```
class FibonacciTrad{
  public static void main(String[] a){
    int ic=0;    //compteur d'instructions
    int[] mem=new int[100000];  //mémoire à plat
    boolean fin = false;        //test pour sortie
    while(!fin) {
      switch(ic){
      case 0: mem[0]=0; ic++; break;
      case 1: mem[1]=1; ic++; break;
      case 2:
      case 3:
      case 4:
      case 5:
      case 6:
      case 7:
      case 8:
      case 9:
      case 10:
      case 11:
} } } }
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
  public static void main(String[] a){
    int ic=0;    //compteur d'instructions
    int[] mem=new int[100000];  //mémoire à plat
    boolean fin = false;        //test pour sortie
    while(!fin) {
      switch(ic){
        case 0: mem[0]=0; ic++; break;
        case 1: mem[1]=1; ic++; break;
        case 2: mem[3]=12; ic++; break;
        case 3:
        case 4:
        case 5:
        case 6:
        case 7:
        case 8:
        case 9:
        case 10:
        case 11:
} } } }
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
  public static void main(String[] a){
    int ic=0;    //compteur d'instructions
    int[] mem=new int[100000];  //mémoire à plat
    boolean fin = false;        //test pour sortie
    while(!fin) {
      switch(ic){
        case 0: mem[0]=0; ic++; break;
        case 1: mem[1]=1; ic++; break;
        case 2: mem[3]=12; ic++; break;
        case 3: mem[4]=0; ic++; break;
        case 4:
        case 5:
        case 6:
        case 7:
        case 8:
        case 9:
        case 10:
        case 11:
} } } }
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
 public static void main(String[] a){
   int ic=0;    //compteur d'instructions
   int[] mem=new int[100000];  //mémoire à plat
   boolean fin = false;        //test pour sortie
   while(!fin) {
      switch(ic){
      case 0: mem[0]=0; ic++; break;
      case 1: mem[1]=1; ic++; break;
      case 2: mem[3]=12; ic++; break;
      case 3: mem[4]=0; ic++; break;
      case 4: if(!(mem[4]<mem[3])) {ic=10;}
              else ic++; break;
      case 5:
      case 6:
      case 7:
      case 8:
      case 9:
      case 10:
      case 11:
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
 public static void main(String[] a){
   int ic=0;    //compteur d'instructions
   int[] mem=new int[100000];  //mémoire à plat
   boolean fin = false;        //test pour sortie
   while(!fin) {
      switch(ic){
      case 0: mem[0]=0; ic++; break;
      case 1: mem[1]=1; ic++; break;
      case 2: mem[3]=12; ic++; break;
      case 3: mem[4]=0; ic++; break;
      case 4: if(!(mem[4]<mem[3])) {ic=10;}
              else ic++; break;
case 5: mem[2]=mem[0]+mem[1]; ic++; break;
      case 6:
      case 7:
      case 8:
      case 9:
      case 10:
      case 11:
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
 public static void main(String[] a){
   int ic=0;    //compteur d'instructions
   int[] mem=new int[100000];  //mémoire à plat
   boolean fin = false;        //test pour sortie
   while(!fin) {
      switch(ic){
      case 0: mem[0]=0; ic++; break;
      case 1: mem[1]=1; ic++; break;
      case 2: mem[3]=12; ic++; break;
      case 3: mem[4]=0; ic++; break;
      case 4: if(!(mem[4]<mem[3])) {ic=10;}
              else ic++; break;
      case 5: mem[2]=mem[0]+mem[1]; ic++; break;
      case 6: mem[0]=mem[1]; ic++ ;break;
      case 7:
      case 8:
      case 9:
      case 10:
      case 11:
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
 public static void main(String[] a){
   int ic=0;    //compteur d'instructions
   int[] mem=new int[100000];  //mémoire à plat
   boolean fin = false;        //test pour sortie
   while(!fin) {
      switch(ic){
      case 0: mem[0]=0; ic++; break;
      case 1: mem[1]=1; ic++; break;
      case 2: mem[3]=12; ic++; break;
      case 3: mem[4]=0; ic++; break;
      case 4: if(!(mem[4]<mem[3])) {ic=10;}
              else ic++; break;
      case 5: mem[2]=mem[0]+mem[1]; ic++; break;
      case 6: mem[0]=mem[1]; ic++ ;break;
      case 7: mem[1]=mem[2]; ic++; break;
      case 8:
      case 9:
      case 10:
      case 11:
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
 public static void main(String[] a){
   int ic=0;    //compteur d'instructions
   int[] mem=new int[100000];  //mémoire à plat
   boolean fin = false;        //test pour sortie
   while(!fin) {
      switch(ic){
      case 0: mem[0]=0; ic++; break;
      case 1: mem[1]=1; ic++; break;
      case 2: mem[3]=12; ic++; break;
      case 3: mem[4]=0; ic++; break;
      case 4: if(!(mem[4]<mem[3])) {ic=10;}
              else ic++; break;
      case 5: mem[2]=mem[0]+mem[1]; ic++; break;
      case 6: mem[0]=mem[1]; ic++ ;break;
      case 7: mem[1]=mem[2]; ic++; break;
      case 8: mem[4]++; ic++; break;
      case 9:
      case 10:
      case 11:
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
 public static void main(String[] a){
   int ic=0;    //compteur d'instructions
   int[] mem=new int[100000];  //mémoire à plat
   boolean fin = false;        //test pour sortie
   while(!fin) {
      switch(ic){
      case 0: mem[0]=0; ic++; break;
      case 1: mem[1]=1; ic++; break;
      case 2: mem[3]=12; ic++; break;
      case 3: mem[4]=0; ic++; break;
      case 4: if(!(mem[4]<mem[3])) {ic=10;}
              else ic++; break;
      case 5: mem[2]=mem[0]+mem[1]; ic++; break;
      case 6: mem[0]=mem[1]; ic++ ;break;
      case 7: mem[1]=mem[2]; ic++; break;
      case 8: mem[4]++; ic++; break;
      case 9: ic=4; break; //saut inconditionnel
      case 10:
      case 11:
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
 public static void main(String[] a){
   int ic=0;    //compteur d'instructions
   int[] mem=new int[100000];  //mémoire à plat
   boolean fin = false;        //test pour sortie
   while(!fin) {
      switch(ic){
      case 0: mem[0]=0; ic++; break;
      case 1: mem[1]=1; ic++; break;
      case 2: mem[3]=12; ic++; break;
      case 3: mem[4]=0; ic++; break;
      case 4: if(!(mem[4]<mem[3])) {ic=10;}
              else ic++; break;
case 5: mem[2]=mem[0]+mem[1]; ic++; break;
      case 6: mem[0]=mem[1]; ic++ ;break;
      case 7: mem[1]=mem[2]; ic++; break;
      case 8: mem[4]++; ic++; break;
      case 9: ic=4; break; //saut inconditionnel
      case 10: S.o.p("fibo("+mem[3]+")="+mem[1]);
      case 11:
```

```java
class FibonacciBoucleDeconstruite{
  public static void main(String[] a){
    int u=0;  // ic = 0: u stocké à mem[0]
    int v=1;  // ic = 1: v stocké à mem[1]
    int w;    // w stocké à mem[2]
    int k=12; // ic = 2: k stocké à mem[3]
    int i=0;  // ic = 3: i stocké à mem[4]
    while(true)
      if(i<k){ // ic = 4: saut cond. vers 10
        w=u+v; // ic = 5
        u=v;   // ic = 6
        v=w;   // ic = 7
        i++;   // ic = 8
      } else break;  //ic = 9 saut incond. vers 4
      System.out.println("fibo("+k+")="+v);
                  // ic = 10
  }  // ic = 11 : sortie
}
```

```java
class FibonacciTrad{
 public static void main(String[] a){
   int ic=0;    //compteur d'instructions
   int[] mem=new int[100000];  //mémoire à plat
   boolean fin = false;        //test pour sortie
   while(!fin) {
      switch(ic){
      case 0: mem[0]=0; ic++; break;
      case 1: mem[1]=1; ic++; break;
      case 2: mem[3]=12; ic++; break;
      case 3: mem[4]=0; ic++; break;
      case 4: if(!(mem[4]<mem[3])) {ic=10;}
              else ic++; break;
case 5: mem[2]=mem[0]+mem[1]; ic++; break;
      case 6: mem[0]=mem[1]; ic++ ;break;
      case 7: mem[1]=mem[2]; ic++; break;
      case 8: mem[4]++; ic++; break;
      case 9: ic=4; break; //saut inconditionnel
      case 10: S.o.p("fibo("+mem[3]+")="+mem[1]);
      case 11: fin = true;
```