Fonctionnement des ordinateurs

chapitre VIII : programmation bas niveau

Prof. Xavier Gandibleux

Université de Nantes
Département Informatique – UFR Sciences et Techniques

Année académique 2019-2020

Programmation

Instructions depuis le niveau du processeur
Assembleur, langage assembleur
Langage machine $\leftrightarrow$ langage assembleur $\leftrightarrow$ langage C
De l'algorithme au 0/1

programmation

Instructions depuis le niveau du processeur

Programmation

Selon la distance entre le langage de programmation et le processeur :

Langages :

- haut niveau (génériques)
  Fortran, Cobol, Lisp, C, C++, Ada, Java, Python, Julia, etc.
  ↳ compilateur et éditeur de liens (cas d'un langage compilé)
- bas niveau (spécifiques à chaque processeur)
  différents niveaux de représentation sous une forme lisible par un humain d'un code exécutable par un processeur :
  - langage assembleur
    niveau 0 : mnémoniques, directives, étiquettes, pseudo-instructions, etc.
    ↳ assembleur
  - langage machine
    niveau -1 : mnémoniques
    niveau -2 : codes machine

Processeur :

- 0 et 1 (code exécutable)

Programmation

Assembleur

Programme informatique chargé de produire un fichier binaire à partir d'un fichier source (programme) :

- traduire la représentation symbolique des instructions en leur code binaire
- allouer une adresse à chaque symbole (nom de variables, étiquette) utilisé par le programme

Désassembleur

Programme informatique chargé de passer du fichier binaire au code source assembleur correspondant

Programmation

Langage assembleur

Exemples de facilités usuellement offertes :

- `* = $C000`
- `LABEL1 LDA #$04`
- `BNE LABEL2`
- `STORE = $0800`
- `ICI = *`
- `; un commentaire`

Programmation

Phases de l'assemblage

Travail en 2 passes

1.
   - représentations symboliques des codes d'instructions $\rightarrow$ binaire
   - noms des opérantes $\rightarrow$ adresses mémoire
   - construire la table des symboles (symbole $\leftrightarrow$ valeur, adresse)
2. fixer les références non encore fixées

Exemple 1 :

```
*=$C000
            LDX #0
Label1      TXA
            STA $0400,X
            LDA #1
            STA $D800,X
            INX
            BNE Label1
            RTS
.END
```

```
C000                LDX #0          A2 00
C002    LABEL1      TXA             8A
C003                STA $0400,X     9D 00 04
C006                LDA #1          A9 01
C008                STA $D800,X     9D 00 D8
C00B                INX             E8
C00C                BNE LABEL1      D0 F4
C00E                RTS             60
C00F    .END
```

Programmation

Exemple 2 (do-while) :

Langage machine 6502 :

```
   ⋮
0100   LDX #$03
0102   NOP
0103   DEX
0104   NOP
0105   BNE $FC
0107   NOP
0108   BRK
   ⋮
```

Langage assembleur 6502 :

```
* = $0100
val          = $03
             LDX val
             NOP
do           DEX
             NOP
while        BNE do
             NOP
             BRK
```

Langage C :

```
int main(void)
{
  int x=3;
  do
  {
    x--;
  } while(x>0);
}
```

Programmation

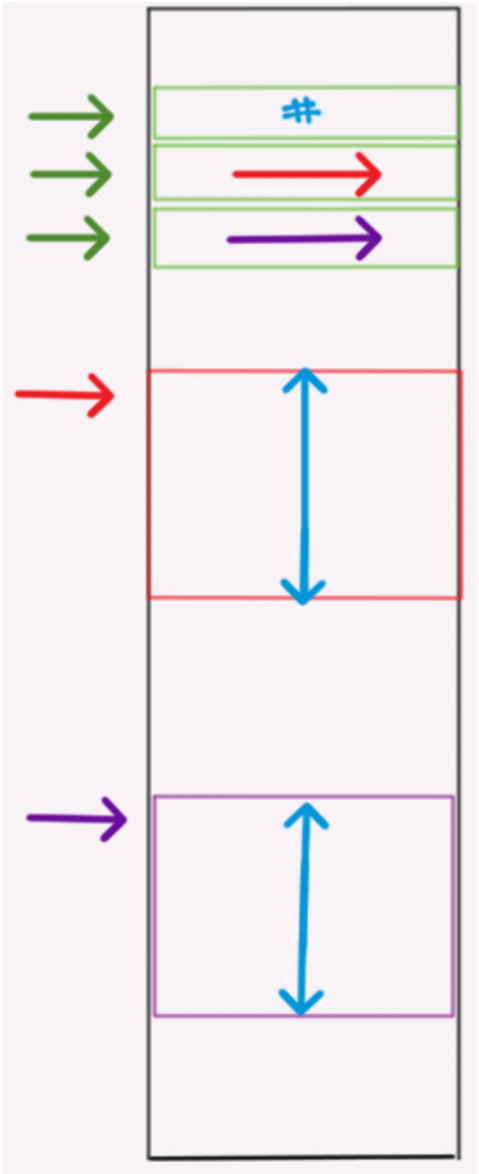
Exemple 3 (relocate) :

Langage assembleur 6502 :

```
long = $50
from = $51
to   = $53
     LDY #$00
next LDA (FROM),Y
     STA (TO),Y
     INY
     CPY LONG
     BNE NEXT
.END

en 0050:
03 00 01 00 02

en 0100:
02 07 04
```



Programmation

Exemple 4 (pointer) :

Langage C :

```
#include <stdio.h>
int main (void)
{
  int i, j, *p;
  i = 3;
  p = &i;
  j = *p;
  printf("j = %d\n",j);
}
```

Langage assembleur 6502 :

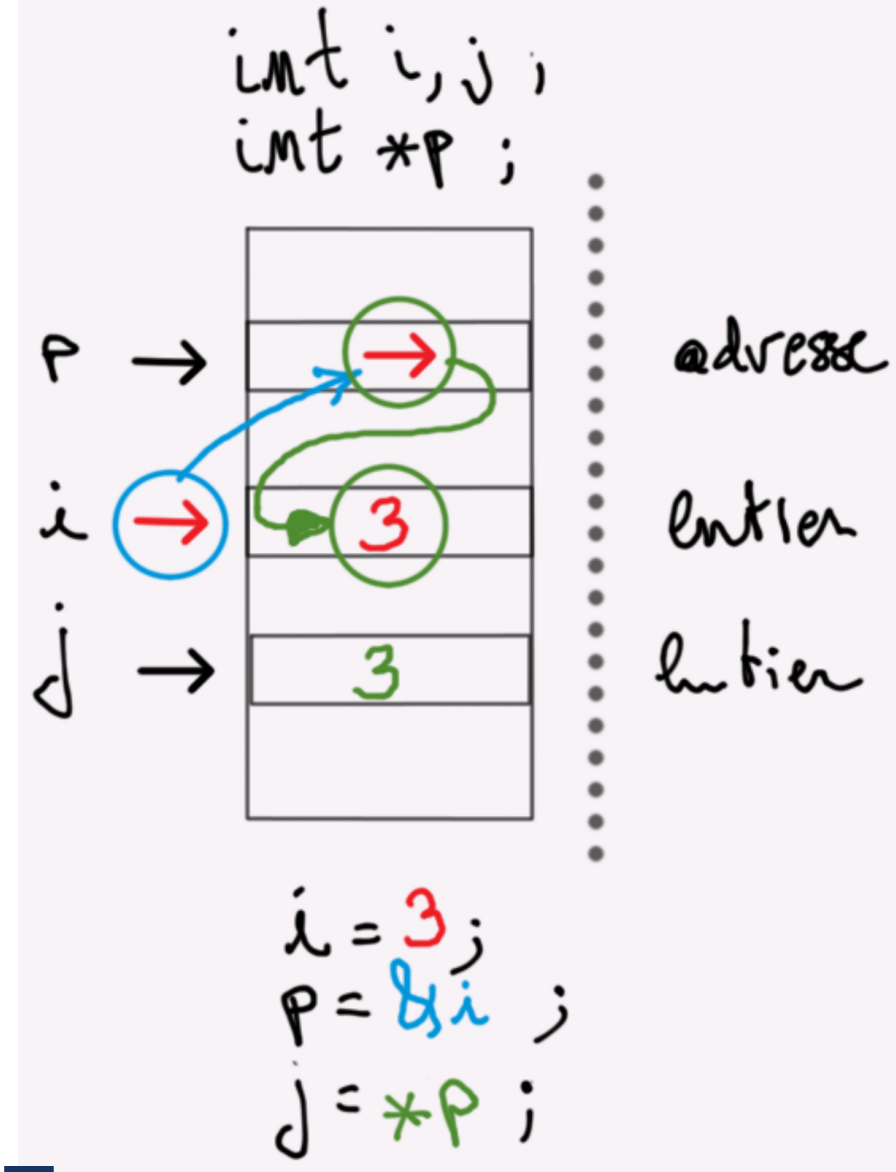
```
adr_p   = $50 ; var_ptrInt
adr_i   = $53 ; var_int
adr_j   = $58 ; var_int

     LDA #$03
     STA adr_i

     LDA #adr_i
     STA adr_p

     LDY #$00
     LDA (adr_p),Y
     STA adr_j
.END

:0050  00 00 00 00 00 00 00 00
:0058  00 00 00 00 00 00 00 00
```



Programmation

Instructions depuis le niveau du processeur : illustration

```
A ← 3
  ⋮
```
Langage de description d'algorithmes

```
A = 3 ;
  ⋮
```
Langage Programmation haut niveau (C)

```
varA   EQU  3
debut  LDX  varA
         ⋮
```
Langage assembleur

```
0100  LDX #$3
0102  ...
```
Langage machine (mnémonique)

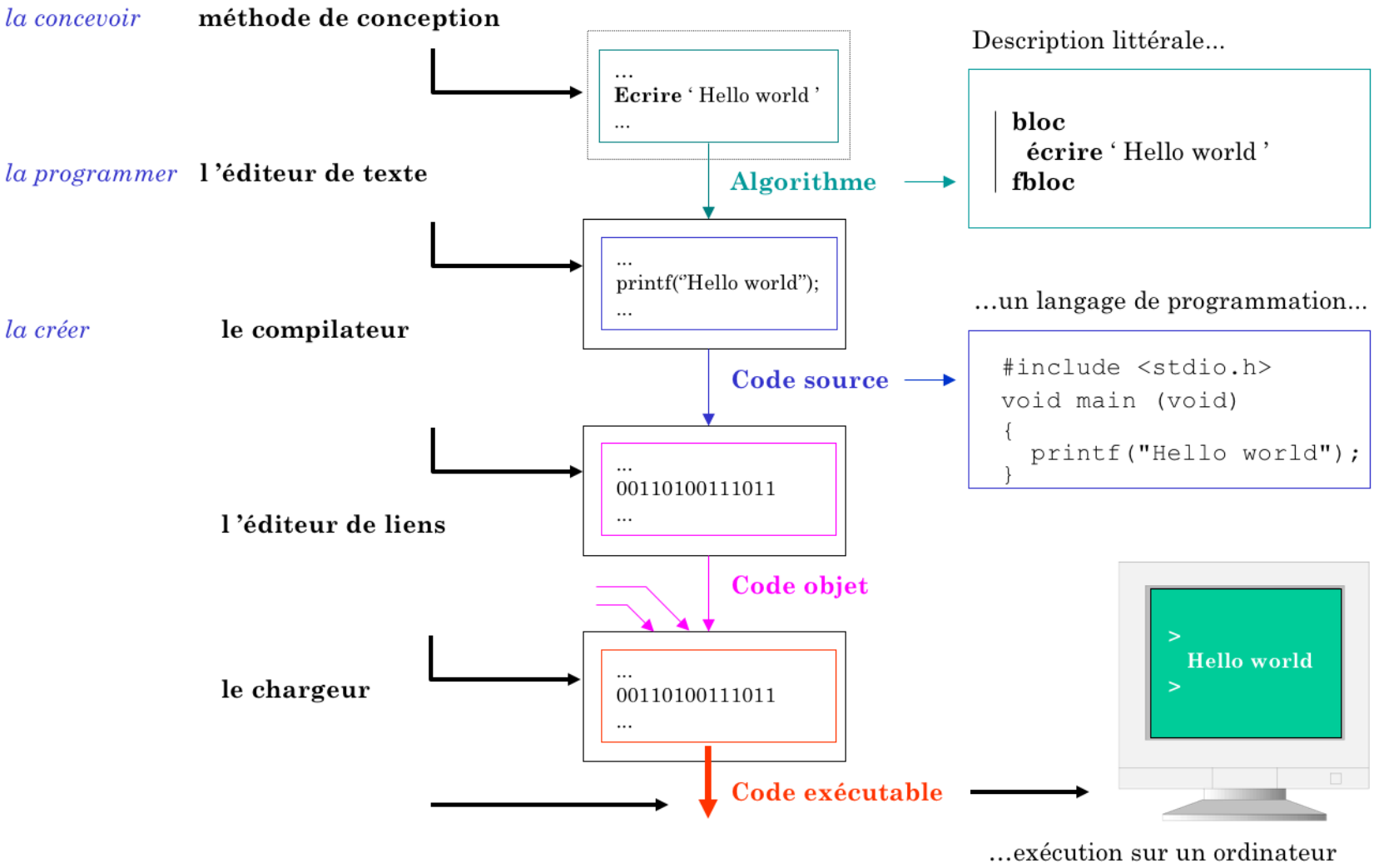
```
0100  A2 03
0102  ...
```
Langage machine (op. code)

```
0100  1010 0010
0101  0000 0011
0102  ...
```
Micro instructions (binaire)

Programmation

Passer d'un langage de haut niveau à son exécution

(cas d'un langage compilé) : `gcc -o hello hello.c`



Fonctionnement de l'ordinateur

That's all folks for now, but...

Fonctionnement de l'ordinateur

Approfondir sur le volet matériel du fonctionnement de l'ordinateur :

- Les mécanismes de piles
- Les mécanismes d'appel à une routine
- Les mécanismes d'interruptions
- Etc.

Approfondir sur des sujets inscrits dans la continuité immédiate :

- Les architectures matérielles
- La programmation en assembleur
- La programmation multicoeur
- La programmation embarquée
- La programmation temps-réel
- Etc.