Introduction à l'architecture des ordinateurs
Codage interne des informations

Michel Salomon

IUT de Belfort-Montbéliard
Département d'informatique

Introduction

Un ordinateur traite des informations de différents types

- Instructions → `mov AH,09h` ⇔ `1011010000001001` ⇔ `B409h` ; etc.
- Nombres → -23 ; 3,1415 ; etc.
- Images → formats `PNG`, `PNM`, `JPG`, `GIF`, etc.
- etc.

Toute information est représentée sous **forme binaire**

- Suite de chiffres binaires → 2 valeurs possibles : **0** ou **1**
- Un chiffre binaire ou ***bi**nary digi**t*** → information élémentaire

Codage / format d'une information

Définit la correspondance entre représentations externe ↔ interne

- Convention / règle à suivre pour représenter une information
- Exemple : représentation de la quantité dix → `dix`, `10`, `X`, etc.

Introduction

En interne, un ordinateur traite deux types d'informations

1. les instructions (`mov AH,09h` $\rightarrow$ 8086 - 16 bits) ;
   - Représentent les opérations effectuées par un ordinateur ;
   - elles sont composées de deux champs :

     1 - le code de l'opération ($B4_{16} = 10110100_2$) ;

     `mov AH,`*val*

     2 - les opérandes de l'opération ($09_{16} = 00001001_2$)

     `mov AH,09h` = `B409h`

   - elles constituent le langage machine ;
   - le langage le plus proche du langage machine est le langage d'assemblage ou Assembleur
2. les données élémentaires

Introduction

En interne, un ordinateur traite deux types d'informations

1. les instructions ;
2. les données élémentaires
   - Ce sont les opérandes et les résultats d'opérations ;
   - elles sont de deux types :

     1 - numériques ;
     - nombres entiers relatifs : -23 ; -1 ; 0 ; 1 ; 12 ; 234 ; etc.
     - nombres fractionnaires : -0,125 ; 3,1415 ; etc.
     - nombres en notation scientifique : $3,1 \times 10^5$; etc.

     2 - non numériques
     - caractères alphanumériques : A, B, . . . , Z, a, b, . . . , z, 0, 1, . . . , 9
     - caractères spéciaux : ?, #, $, etc.

Langage machine - Illustration avec le 8086

Assembleur 16 bits dans MS-DOS - `bonjour.asm`

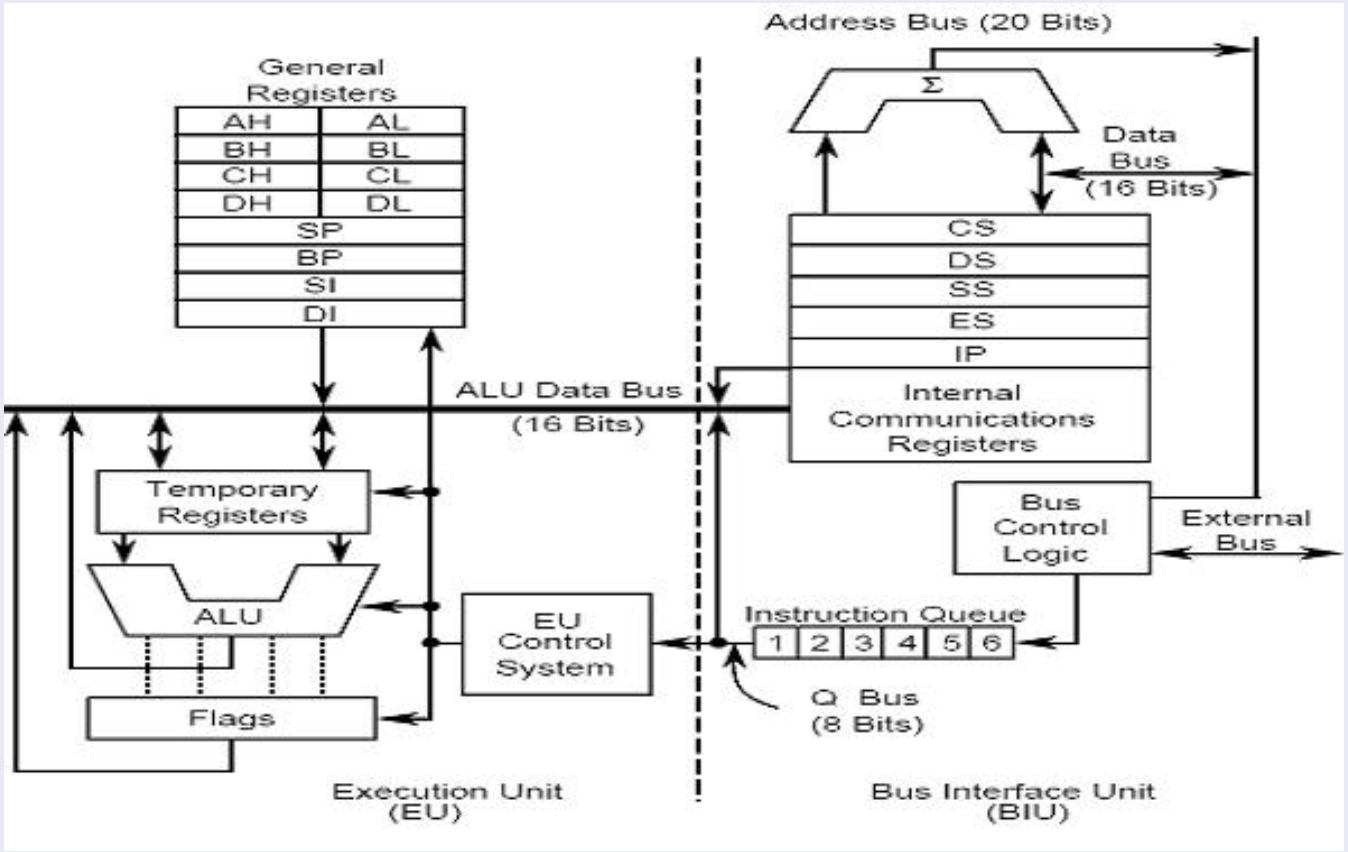
```
bits 16 ; indique qu'on produit du code 16 bits
org 0x100 ; directive de localisation mém. (COM)
section .data
    MessageBonjour: db 10,'Bonjour, monde...',10,'$'
section .text
    mov DX,MessageBonjour
    mov AH,09h
    int 21h
    mov AH,4Ch
    int 21h
```

Caractéristiques du 8086

- Bus d'adresses de 20 bits, mais registres d'adresses de 16 bits
- Registres de travail de 16 bits (`AX`=`AH` `AL`, `BX`, `CX` et `DX`)

Langage machine - Illustration avec le 8086

Structure globale du 8086 - **B**us **I**nterface **U**nit et **E**xecution **U**nit



Langage machine - Illustration avec le 8086

Assembleur 16 bits dans MS-DOS - `bonjour.asm`

```
bits 16 ; indique qu'on produit du code 16 bits
org 0x100 ; directive de localisation mém. (COM)
section .data
      MessageBonjour: db 10,'Bonjour, monde...',10,'$'
section .text
      mov DX,MessageBonjour
      mov AH,09h
      int 21h
      mov AH,4Ch
      int 21h
```

Dump mémoire du programme

```
00000000  BA0C01                    mov dx,0x10c
00000003  B409                      mov ah,0x9
00000005  CD21                      int 0x21
00000007  B44C                      mov ah,0x4c
00000009  CD21                      int 0x21
```

Langage machine - Illustration avec archi. x86 / IA-32

Assembleur 32 bits dans linux - `bonjour.asm`

```
bits 32
section .data
      MessageBonjour: db 10,'Bonjour, monde...',10
      MessageBonjourLong: equ $-MessageBonjour
section .text
global _start
_start: ; Affichage de la chaine de caracteres
      mov EAX,4
      mov EBX,1
      mov ECX,MessageBonjour
      mov EDX,MessageBonjourLong
      int 80h
      ; Terminaison du programme
      mov EAX,1
      mov EBX,0
      int 80h
```

Processus d'assemblage (dans un terminal)

- Compilation → `nasm -f elf bonjour.asm`
- Édition des liens → `ld -s bonjour.o -o bonjour` (??)

Format de données - Illustration avec des images

Un format est une convention (normalisée) de représentation ou stockage d'un type de données tel que texte, image, son, etc.

Codage des images matricielles

- Une image est un ensemble de points lumineux $\rightarrow$ pixel
- Une image est un tableau 2D (matrice) $\rightarrow$ contient les pixels
- Profondeur de codage / couleur $\rightarrow$ nombre de bits par pixel
  - 1 bits $\rightarrow 2^1 = 2$ couleurs (noir et blanc)
  - 8 bits $\rightarrow 2^8 = 256$ couleurs (avec palette) ou niveaux de gris
  - 24 bits $\rightarrow 2^{24} = 16$ millions de couleurs (codage RVB)
- Représentation des couleurs sur écran $\rightarrow$ codage RVB
  - 1 pixel logique correspond à 3 pixels physiques à l'écran
  - chaque pixel d'un triplet physique émet une couleur primaire

Unités de mesure

Le bit (0 ou 1) est la plus petite unité de mesure

Toutes les autres unités sont des regroupements de bits

- un quartet est un groupement de 4 bits ;
- un octet (*Byte*) est un groupement de 8 bits ;
- un mot (*Word*) est un groupement d'octets (2 ou plus)
  - unité de base manipulée par un processeur (registres, etc.)

Unités de mesure basées sur l'octet

- Principalement utilisées pour parler du stockage de données
- Deux types d'unités : en puissances de 2 ; en puissances de 10
- Issues de recommandations officielles (IEC 60027-2 - 1998)
- Historiquement, utilisation des dénominations officielles des puissances de 10 pour les puissances de 2

Unités de mesure

Le bit (0 ou 1) est la plus petite unité de mesure

Unités de mesure basées sur l'octet

- Principalement utilisées pour parler du stockage de données

Puissances de 2			
1 Kibioctet	Kio ou KiB	1024 octets	2^{10}
1 Mébioctet	Mio ou MiB	1024 Kio	2^{20}
1 Gibioctet	Gio ou GiB	1024 Mio	2^{30}
1 Tébioctet	Tio ou TiB	1024 Gio	2^{40}

Puissances de 10			
1 Kilooctet	ko ou kB	1000 octets	10^3
1 Mégaoctet	Mo ou MB	1000 ko	10^6
1 Gigaoctet	Go ou GB	1000 Mo	10^9
1 Téraoctet	To ou TB	1000 Go	10^{12}

- 1 Kibioctet = 1 "Kilo binaire octet"

Unités de mesure

Usage encore confus des deux types d'unités

- Au niveau matériel
  - Octets comptés en puissances de 2
  - Souvent en utilisant les dénominations historiques (puissances de 10)
  - Exemple : barrette mémoire de 8 Go à la place de 8 Gio
- Au niveau logiciel
  - Octets pouvant être comptés dans l'une ou l'autre des unités
  - Usage fréquent des dénominations historiques pour les puissances de 2
  - Exemples (quelques commandes à lancer dans un terminal)
    - Puissances de 2 ; bonnes dénominations → `/sbin/ifconfig`
    - Puissances de 2 et 10 ; dénominations "historiques" → `df`
    - Puissances de 2 ; dénominations "historiques" → `cat /proc/meminfo`

Numération en base b

Principes

- Un système de numération écrit les nombres à partir
  - d'une liste finie de symboles appelés chiffres ;
  - de règles définissant comment combiner ces chiffres
- Dans un système de base $b > 1$, un nombre entier naturel N est représenté par la suite de $n + 1$ chiffres

$$c_n c_{n-1} \dots c_1 c_0$$

tels que :

- $c_i \in \{0, 1, \dots, b-2, b-1\}$;
- $c_n \neq 0$

et vérifiant

$$N = c_n \cdot b^n + c_{n-1} \cdot b^{n-1} + \dots + c_1 \cdot b^1 + c_0 \cdot b^0 = \sum_{i=0}^{n} c_i \cdot b^i$$

Numération en base b

Principes (suite)

$$N = c_n \cdot b^n + c_{n-1} \cdot b^{n-1} + \cdots + c_1 \cdot b^1 + c_0 \cdot b^0 = \sum_{i=0}^{n} c_i \cdot b^i$$

- À chaque position est associée un poids : la puissance de b
  - c_n est le chiffre de poids fort ;
  - c_0 est le chiffre de poids faible

Exemple - écriture de 2017 en base 10

- Utilisation de $b = 10$ chiffres $\rightarrow c_i \in \{0, 1, 2, \ldots, 9\}$
- Nombre écrit comme une addition de puissances de $b = 10$

$$\begin{aligned} 2017_{10} &= 2 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 7 \times 10^0 \\ &= 2000 + 0 + 10 + 7 \\ &= 2017 \end{aligned}$$

Numération en base b

Conversion d'une base b quelconque vers la base 10

- Nombre en base 3 $\rightarrow c_i \in ?$

$(201)_3 = 201_3 = ?$

- Nombre en base 11 $\rightarrow c_i \in ?$

$(6A)_{11} = 6A_{11} = ?$

Systèmes de numération usuels en informatique

- Système binaire, $b = 2$ et $c_i \in \{0, 1\}$;
- Système octal, $b = 8$ et $c_i \in \{0, \ldots, 7\}$;
- Système hexadécimal, $b = 16$ et $c_i \in \{0, \ldots, 9, A, B, \ldots, F\}$
- Intérêt des systèmes octal et hexadécimal
  - représentation aisée des nombres ayant beaucoup de bits ;
  - conversion avec le système binaire simple (puissances de 2)

Numération en base b

Conversion d'une base b quelconque vers la base 10

- Nombre en base 3 $\rightarrow c_i \in \{0, 1, 2\}$

$$(201)_3 = 201_3 = (2)_{10} \times 3^2 + (0)_{10} \times 3^1 + (1)_{10} \times 3^0 = 2 \times 9 + 1 \times 1 = 19$$

- Nombre en base 11 $\rightarrow c_i \in \{0, 1, \ldots, 8, 9, A\}$

$$(6A)_{11} = 6A_{11} = (6)_{10} \times 11^1 + (A)_{10} \times 11^0 = 6 \times 11 + 10 \times 1 = 76$$

Systèmes de numération usuels en informatique

- Système binaire, $b = 2$ et $c_i \in \{0, 1\}$;
- Système octal, $b = 8$ et $c_i \in \{0, \ldots, 7\}$;
- Système hexadécimal, $b = 16$ et $c_i \in \{0, \ldots, 9, A, B, \ldots, F\}$
- Intérêt des systèmes octal et hexadécimal
  - représentation aisée des nombres ayant beaucoup de bits ;
  - conversion avec le système binaire simple (puissances de 2)

Numération en base b

Correspondance entre les systèmes de numération usuels

Décimal	Binaire	Octal	Hexadécimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Changement de base

Passage indirect d'un système de numération à un autre

1. Conversion de la base de départ b_d vers la base 10
2. Conversion de la base 10 vers la base d'arrivée b_a

Exemple 1 - Conversion de $6BC_{16}$ en base 12

1. Conversion de la base 16 vers la base 10

$$\begin{aligned} 6BC_{16} &= (6)_{10} \times 16^2 + (B)_{10} \times 16^1 + (C)_{10} \times 16^0 \\ 6BC_{16} &= 6 \times 16^2 + 11 \times 16^1 + 12 \times 16^0 \\ 6BC_{16} &= 6 \times 256 + 11 \times 16 + 12 \times 1 = 1724_{10} \end{aligned}$$

2. Conversion de la base 10 vers la base 12

$$\begin{aligned} 1724/12 = 143 \text{ reste } 8 &\leftrightarrow 1724 = 143 \times 12 + 8 \\ 143/12 = 11 \text{ reste } 11 &\leftrightarrow 143 = 11 \times 12 + 11 \\ 11/12 = 0 \text{ reste } 11 &\leftrightarrow 11 = 0 \times 12 + 11 \end{aligned}$$

On obtient donc $6BC_{16} = 1724_{10} = BB8_{12}$

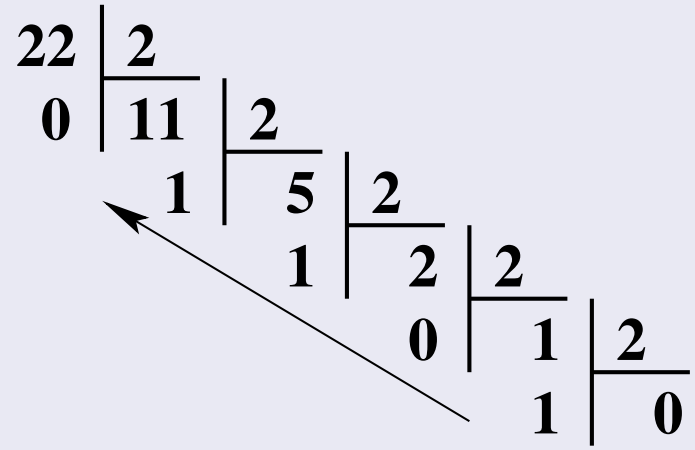
Changement de base

Exemple 2 - Conversion de 22_{10} en base 2

Quotients successifs ↓

22	0
11	1
5	1
2	0
1	1
0	

↑ *Restes successifs*



Conversions usuelles en informatique

- binaire ; octal ; hexadécimal → décimal
  - Additionner, respectivement, des puissances de 2 ; 8 ; 16
- décimal → binaire ; octal ; hexadécimal
  1. Divisions entières successives, respectivement par 2 ; 8 ; 16, tant que le quotient est non nul ;
  2. Lecture des restes du dernier vers le premier

Changement de base

Conversions octal ou hexadécimal $\rightarrow$ binaire

- Chaque chiffre octal ou hexadécimal est éclaté en son équivalent binaire sur respectivement 3 ou 4 bits
- Exemples :
  - $22_8 = 010\ 010_2 = 10010_2$;
  - $8A_{16} = 1000\ 1010_2 = 10001010_2$

Conversion binaire $\rightarrow$ octal ou hexadécimal

- On remplace de droite à gauche chaque groupe de 3 ou 4 bits par son équivalent octal ou hexadécimal
- Si le nombre de bits du nombre binaire n'est pas un multiple de 3 ou 4, on complète à gauche avec des bits à 0
- Exemples :
  - $010101_2 = 010\ 101_2 = 25_8$;
  - $10101_2 = 0001\ 0101_2 = 15_{16}$