Mémo

Représentation machine des données

Résumé Nos ordinateurs actuels utilisent la représentation binaire pour stocker les données. Dans ce mémo, nous allons voir comment les données sont représentées en mémoire.

Table des matières

1. Bases des nombres

1.1. Introduction

Les nombres peuvent être représentés dans différentes bases telles que décimale, binaire, octale et hexadécimale.

1.2. Nombres décimaux

- Base 10
- Chiffres : 0-9
- Exemple : 456, 99988, 67890, 45433

1.3. Nombres binaires

- Base 2
- Chiffres : 0, 1
- Exemple : 1011, 10111101, 101101010, 11111111

1.4. Nombres octaux

- Base 8
- Chiffres : 0-7
- Exemple : 76403, 22334, 54634

1.5. Nombres hexadécimaux

- Base 16
- Chiffres : 0-9, A-F
- Préfixe : 0x ou 0X
- Exemple : AB04, FF0A, 456FEA09

---

2. Conversion entre les bases

2.1. Introduction

La conversion entre différentes bases de nombres est une opération courante en informatique.

2.2. Conversion binaire vers décimal

- Formule : $b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \ldots + b_1 \times 2^1 + b_0 \times 2^0$
- Exemple : Binaire 10110 vers Décimal
  - Calcul : $1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
  - Résultat : 22

2.3. Conversion binaire vers octal

- Regrouper les chiffres binaires en ensembles de trois, en commençant par la droite.
- Convertir chaque groupe en son équivalent octal.
- Exemple : Binaire 1010001101011 vers Octal
  - Regroupement : 001 010 001 101 011
  - Conversion : 1 2 1 5 3
  - Résultat : 12153

2.4. Conversion binaire vers hexadécimal

- Regrouper les chiffres binaires en ensembles de quatre, en commençant par la droite.
- Convertir chaque groupe en son équivalent hexadécimal.
- Exemple : Binaire 11010001101011 vers Hexadécimal
  - Regroupement : 0011 0100 0110 1011
  - Conversion : 3 4 6 B
  - Résultat : 346B

2.5. Conversion décimale vers binaire

- Utiliser une division successive par 2.
- Enregistrer les restes.
- Exemple : Décimal 351 vers Binaire
  - Étapes de division : 351 ÷ 2 = 175 reste 1, 175 ÷ 2 = 87 reste 1, etc.
  - Résultat : 101011111

2.6. Conversion octale vers binaire

- Convertir chaque chiffre octal en une chaîne binaire de 3 bits.
- Concaténer les résultats.
- Exemple : Octal 763015 vers Binaire
  - Conversion : 111 110 011 000 001 101
  - Résultat : 111110011000001101

2.7. Conversion hexadécimale vers binaire

- Convertir chaque chiffre hexadécimal en une chaîne binaire de 4 bits.
- Concaténer les résultats.
- Exemple : Hexadécimal 0x1AF395 vers Binaire
  - Conversion : 0001 1010 1111 0011 1001 0101
  - Résultat : 000110101111001110010101

2.8. Conversion hexadécimale vers décimale

- Formule : $h_{n-1} \times 16^{n-1} + h_{n-2} \times 16^{n-2} + \ldots + h_1 \times 16^1 + h_0 \times 16^0$
- Exemple : Hexadécimal 0xFF vers Décimal
  - Calcul : $15 \times 16^1 + 15 \times 16^0$

- Résultat : 255

---

3. Entiers positifs

3.1. Introduction

Les entiers positifs sont également appelés entiers non signés et sont représentés par leurs équivalents binaires.

3.2. Exemple

- Hexadécimal 0xFF vers Décimal : 255
- Représentation binaire : 11111111

---

4. Forme du complément à deux

4.1. Introduction

Le complément à deux est utilisé pour représenter les entiers négatifs.

4.2. Trouver le complément à deux

- Commencer à partir de la position la plus à droite et procéder vers la gauche jusqu'à ce que le premier 1 soit rencontré.
- Inverser chaque bit à sa forme complémentaire après le premier 1.

4.3. Exemple

- Binaire 10101110001000 vers Complément à deux
  - Étapes : 10101110001000 → 01010001111000
  - Résultat : 01010001111000

---

5. Entiers négatifs

5.1. Introduction

Les entiers négatifs sont représentés sous forme de complément à deux.

5.2. Exemple

- Décimal 17 vers Binaire : 00010001
- Décimal −17 vers Complément à deux : 11101111

---

6. Registres

6.1. Introduction

Les registres sont des blocs de construction de mémoire des unités de mémoire capables de contenir un certain nombre de bits.

6.2. Exemple

- Un registre de 8 bits contient 8 cellules et peut contenir 8 bits d'information.

---

7. Unités de mémoire

7.1. Introduction

Une mémoire est composée d'un certain nombre de registres, chacun avec une adresse.

7.2. Exemple

- Unité de mémoire avec des adresses de registre de 16 bits :
  - Adresse : 0x000000000065FE18
  - Contenu : 0x78

---

8. Comment les entiers sont-ils stockés dans la mémoire de l'ordinateur, Big-Endian et Little-Endian ?

8.1. Introduction

Les nombres sont stockés dans des registres, et il existe deux types de méthodes de stockage de données : little-endian et big-endian.

8.2. Little-Endian et Big-Endian

- Little-Endian : L'octet le moins significatif est écrit en mémoire en premier.
- Big-Endian : L'octet le plus significatif est écrit en mémoire en premier.

8.3. Exemple

- Entier 0x1245A78F stocké en mémoire :
  - Little-Endian : 0x8F, 0xA7, 0x45, 0x12
  - Big-Endian : 0x12, 0x45, 0xA7, 0x8F

9. Exemples

9.1. Conversion binaire vers décimale

```
int binaire = 0b10110;
int decimal = 0;
int base = 1;
while (binaire > 0) {
  int reste = binaire % 10;
  decimal += reste * base;
  binaire /= 10;
  base *= 2;
}
printf("Décimal : %d\n", decimal);
```

```
Décimal : 6
```

9.2. Conversion binaire vers octale

```
int binaire = 0b1010001101011;
int octal = 0;
int base = 1;
while (binaire > 0) {
  int reste = binaire % 1000;
  octal += reste * base;
  binaire /= 1000;
  base *= 10;
}
printf("Octal : %o\n", octal);
```

```
Octal : 425
```

9.3. Conversion binaire vers hexadécimale

```
int binaire = 0b11010001101011;
int hexadecimal = 0;
int base = 1;
while (binaire > 0) {
  int reste = binaire % 10000;
  hexadecimal += reste * base;
  binaire /= 10000;
  base *= 10;
}
printf("Hexadécimal : %x\n", hexadecimal);
```

```
Hexadécimal : d65
```

9.4. Conversion décimale vers binaire

```
int decimal = 351;
int binaire = 0;
int base = 1;
while (decimal > 0) {
  int reste = decimal % 2;
  binaire += reste * base;
  decimal /= 2;
  base *= 10;
}
printf("Binaire : %d\n", binaire);
```

```
Binaire : 101011111
```

9.5. Conversion octale vers binaire

```c
int octal = 0763015;
int binaire = 0;
int base = 1;
while (octal > 0) {
  int reste = octal % 10;
  binaire += reste * base;
  octal /= 10;
  base *= 1000;
}
printf("Binaire : %d\n", binaire);
```

```
Binaire : -1692889791
```

9.6. Conversion hexadécimale vers binaire

```c
int hexadecimal = 0x1AF395;
int binaire = 0;
int base = 1;
while (hexadecimal > 0) {
  int reste = hexadecimal % 16;
  binaire += reste * base;
  hexadecimal /= 16;
  base *= 10000;
}
printf("Binaire : %d\n", binaire);
```

```
Binaire : 1210643093
```

9.7. Conversion hexadécimale vers décimale

```c
int hexadecimal = 0xFF;
int decimal = 0;
int base = 1;
while (hexadecimal > 0) {
  int reste = hexadecimal % 16;
  decimal += reste * base;
  hexadecimal /= 16;
  base *= 16;
}
printf("Décimal : %d\n", decimal);
```

```
Décimal : 255
```

Gazi, Orhan. 2024. *Modern C programming: including standards C99, C11, C17, C23*. Cham: Springer.