Bitovi

Jelena Hadzi-Puric

January 2021

1 Uvod

Svi podaci u računarstvu su interno registrovani kao bitovi, tj. kao cifre 0 i 1. U algoritmici, operacije nad bitovima i binarnim predstavama celih brojeva zauzimaju znacajno mesto.

2 Primene

U programiranju, n bitni ceo broj se interno u memoriji cuva kao binarni broj koji se sastoji od n bitova. Na primer, tip `int` u C++ je 32-bitni tip, tj. svaka vrednost tipa `int` je predstavljena zapisom duzine 32 bita.

Pogledajte predstavljanje `int` broja 43:

$$00000000000000000000000000101011$$

Bitovi u reprezentaciji su indeksirani sdesna nalevo. Da konvertujemo bitsku predstavu $b_k \cdots b_2 b_1 b_0$ u dekadni broj, mozemo koristiti formulu

$$b_k 2^k + \ldots + b_2 2^2 + b_1 2^1 + b_0 2^0.$$

Na primer,

$$1 \cdot 2^5 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^0 = 43.$$

Bitska reprezentacija broja moze podrazumevati signed ili unsigned brojeve.

Oznaceni ceo broj zapisan sa n bitova moze sadrzati bilo koji ceo broj izmedju -2^{n-1} i $2^{n-1} - 1$. Na primer, tip `int` u C++ je oznacen, tako da `int` promenljiva moze da sadrzi bilo koji ceo broj izmedju -2^{31} i $2^{31} - 1$.

Na primer, bitska reprezentacija `int` broja -43 je

$$11111111111111111111111111010101.$$

Neoznaceni ceo broj zapisan sa n bitova moze sadrzati bilo koji ceo broj izmedju 0 and $2^n - 1$. Na primer, u C++, `unsigned int` promenljiva moze da sadrzi bilo koji ceo broj izmedju 0 i $2^{32} - 1$.

Postoji relacija izmedju ovih reprezentacija: oznaceni zapis celog broja $-x$ je jednak $2^n - x$ u zapisu neoznacenog celog broja. Na primer, naredno kod pokazuje da oznaceni zapis broja $x = -43$ je jednak zapisu neoznacene vrednosti $y = 2^{32} - 43$:

```
int x = -43;
unsigned int y = x;
cout << x << "\n"; // -43
cout << y << "\n"; // 4294967253
```

Ukoliko neki broj svojom vrednoscu prevzilazi gornju granicu tipa, tada dolazi do prekoracenja, eng. overflow.

Na primer, razmotrimo kod:

```
int x = 2147483647
cout << x << "\n"; // 2147483647
x++;
cout << x << "\n"; // -2147483648
```

Inicijalno, vrednost za x je $2^{31} - 1$. Ovo je najveca vrednost koja se moze smestiti u `int` promenljivoj, tako da naredni broj nakon $2^{31} - 1$ je -2^{31}.

3 Bitovske operacije

3.0.1 And opercija

Na primer, 22 & 26 = 18, jer

$$\begin{array}{cll} & 10110 & (22) \\ \& & 11010 & (26) \\ \hline = & 10010 & (18) \end{array}$$

Provera da li je broj x paran: $x \,\&\, 1 = 0$ ako x je paran, i $x \,\&\, 1 = 1$ ako x je neparan. Zapravo, x je deljiv sa 2^k samo ako je $x \,\&\, (2^k - 1) = 0$.

3.0.2 Or operacija

$$\begin{array}{cll} & 10110 & (22) \\ | & 11010 & (26) \\ \hline = & 11110 & (30) \end{array}$$

3.0.3 Xor operacija

$$\begin{array}{cll} & 10110 & (22) \\ \hat{} & 11010 & (26) \\ \hline = & 01100 & (12) \end{array}$$

3.0.4 Operacija komplement

Operacije kompleneta broja $\tilde{}x$ kao rezulta daje sve bitove u x invertovane. Vazi formula $\tilde{}x = -x - 1$, i na primer, $\tilde{}29 = -30$.

Rezultat ove operacije na nivou bitova zavisi od duzine bitske reprezentacije.

Na primer, za 32-bitni `int` broj:

$$\begin{array}{rcrl} x & = & 29 & 00000000000000000000000000011101 \\ \tilde{}x & = & -30 & 11111111111111111111111111100010 \end{array}$$

Dodatne funkcije

Ukoliko koristite verziju C++ koju podrzava kompajler g++, bice Vam od pomoci funkcije za prebrojavanje bitova:

- `__builtin_clz`(x): broj vodecih 0 na pocetku binarnog zapisa broja x
- `__builtin_ctz`(x): broj 0 na kraju binarnog zapisa broja x
- `__builtin_popcount`(x): broj 1ca u binarnom zapisu broja x
- `__builtin_parity`(x): parnost (paran ili neparan) broja 1ca u binarnom zapisu broja x

Primer upotrebe:

```
int x = 5328; // 00000000000000000001010011010000
cout << __builtin_clz(x) << "\n"; // 19
cout << __builtin_ctz(x) << "\n"; // 4
cout << __builtin_popcount(x) << "\n"; // 5
cout << __builtin_parity(x) << "\n"; // 1
```

Navedene funkcije podrzavaju samo `int` brojeve, ali postoje i `long long` verzije funkcija dostupne uz sufix `ll`.

4 Predstavljanje skupova

Svaki podskup skupa $\{0, 1, 2, \ldots, n-1\}$ se moze predstaviti kao n bit-ni ceo broj koji u binarnom zapisu ima jedinice na pozicijama koje odgovaraju pozicijama elemenata u polaznom skupu, a koji pripadaju skupu. Ovo je efikasan nacin za predstavljanje skupova, jer za svaki element je potreban samo jedan bit memorije, dok se operacije nad skupovima mogu implementirati kso bitske operacije.

Na primer, kako je `int` 32-bitni tip u C++, `int` brojem se moze reprezentovati bilo koji podskup skupa $\{0, 1, 2, \ldots, 31\}$. Bitovna reprezentacija skupa $\{1, 3, 4, 8\}$ je

$$00000000000000000000000100011010,$$

koja odgovara broju $2^8 + 2^4 + 2^3 + 2^1 = 282$.

4.0.1 Implementacija

U primeru koji sledi, deklarisna je `int` varijabla x koja sadrzi podskup skupa $\{0, 1, 2, \ldots, 31\}$. Nakon toga se podskupu dodaju elementi 1, 3, 4 i 8 i ispsiuje njegova velicina.

```
int x = 0;
x |= (1<<1);
x |= (1<<3);
```

```
x |= (1<<4);
x |= (1<<8);
cout << __builtin_popcount(x) << "\n"; // 4
```

Naredni primer ispisuje sve elemente koji pripadaju skupu:

```
for (int i = 0; i < 32; i++) {
    if (x&(1<<i)) cout << i << " ";
}
// output: 1 3 4 8
```

4.0.2 Skupovne operacije

Skupovne operacije se mogu implementirati kao bitske operacije:

	set syntax	bit syntax
intersection	$a \cap b$	a & b
union	$a \cup b$	a — b
complement	$\bar{a}$	˜a
difference	$a \setminus b$	a & (˜b)

Na primer, the following code first constructs the sets $x = \{1, 3, 4, 8\}$ and $y = \{3, 6, 8, 9\}$, and then constructs the set $z = x \cup y = \{1, 3, 4, 6, 8, 9\}$:

```
int x = (1<<1)|(1<<3)|(1<<4)|(1<<8);
int y = (1<<3)|(1<<6)|(1<<8)|(1<<9);
int z = x|y;
cout << __builtin_popcount(z) << "\n"; // 6
```

4.0.3 Struktura bitset u C++