Prikaz podataka u računalu

Sadržaj

- Osnovni tipovi podataka
- Cijeli brojevi – prikaz i aritmetika
- Prikaz realnih brojeva – standard „*floating-point*”

Prikaz podataka u računalu

- Nakon odslušanog bit ćete u stanju:
  - imenovati jednostavne tipove podataka
  - objasniti prikaz nenumeričkih tipova podataka
  - objasniti prikaz numeričkih tipova podataka (cijeli brojevi bez predznaka i s predznakom)
  - primijeniti na konkretnim primjerima
  - razumjeti realizaciju osnovnih aritmetičkih operacija
  - predvidjeti moguće probleme.

Osnovni tipovi podataka

Osnovni ili fundamentalni tipovi podataka u računalu su:

- one cjeline ili blokovi bitova s kojima računalo „zna nešto raditi” i to neovisno o njihovom sadržaju.

- To znači da postoje instrukcije koje nešto rade s tim cjelinama kao operandima, bez obzira na eventualni dodatni tip operanda. U ovom kontekstu je

tip = interpretacija sadržaja.

Osnovni tipovi podataka (2)

Ako se sjetimo „pravokutnog” izgleda memorije, onda u te tipove sigurno ulaze

- osnovne cjeline koje možemo adresirati, dakle, ono što smo ranije (u skici memorije) nazivali riječ.

Osim toga, računalo može „znati” raditi i s

- manjim cjelinama – dijelovima osnovne cjeline ako je ona dovoljno velika;
- većim cjelinama – blokovima osnovnih cjelina.

Označavanje bitova i adresa

Obzirom na to da sadržaj nije bitan, zapravo jedino što se može reći o tim cjelinama je

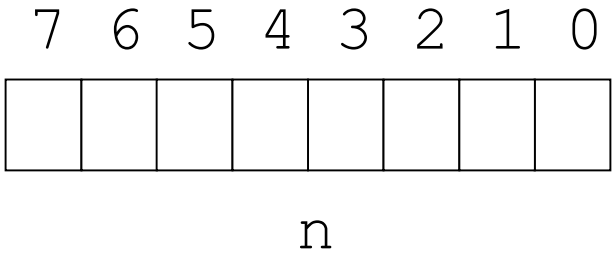
- njihova duljina – u bitovima.

Osnovna cjelina koju možemo adresirati na IA-32 je

1 byte = 8 bitova.

Označavanje bitova i adresa (2)

Jedan bajt, duljine `n=8` bitova, na adresi `N` označavamo na sljedeći način (svaka „kućica” je `1` bit):



Objašnjenje oznaka: Uzmimo da neka cjelina (u ovom slučaju osnovni tip podataka) ima duljinu od `n` bitova.

Označavanje bitova i adresa (3)

- Tradicionalno se pojedini bitovi u cjelini indeksiraju slično kao i riječi u memoriji, dakle od `0` do `n-1`.

Poredak ide tako da

- „najdesniji” bit ima indeks `0` – najniži ili zadnji bit
- „najlijeviji” bit ima indeks `n-1` – najviši ili vodeći bit.

Razlog: Pozicijski prikaz cijelih brojeva u kojem vodeću znamenku pišemo kao prvu (lijevu), a najnižu znamenku kao zadnju (desnu).

Jednostavni tipovi podataka

- Dogovorno, najniža adresa (adresa na kojoj „počinje” cjelina) je ujedno i adresa čitave cjeline.

Ponovimo, osnovni tipovi podataka nisu jako korisni jer s njima ne možemo ništa „pametnije” raditi, osim prijenosa. Za stvarno računanje trebamo

- dodatnu interpretaciju sadržaja cjeline
- operacije s takvom vrstom podataka.

Jednostavni tipovi podataka (2)

Skup podataka i operacije nad njima čine neku algebarsku strukturu koju zajedničkim imenom zovemo tip podataka.

Oni tipovi podataka za koje računalo „zna” ili može

- prikazati pripadni skup podataka
- izvesti pripadne operacije na njima

zovu se **jednostavni** tipovi podataka.

Jednostavni tipovi podataka (3)

Pojam „jednostavni” znači da su operacije na toj vrsti podataka izravno podržane arhitekturom računala, tj.

- postoje instrukcije za njih.

Dakle, te operacije su elementarne operacije (za računalo kao izvršitelja) i u principu su brze.

Standardne jednostavne tipove podataka možemo grubo podijeliti u dvije grupe:

Jednostavni tipovi podataka (4)

- nenumerički tipovi: znakovni tip, logički tip
- numerički tipovi: cjelobrojni tip, realni tip, realni tip u dvostrukoj preciznosti, …

Nenumerički tipovi zapravo se svode na numeričke.

Znakovi:

- prikaz je u nekom kôdu
- sve funkcije na znakovima svode se na elementarne operacije na cijelim brojevima.

Nenumerički tipovi podataka

- Znakovni tip u programskom jeziku C: `char`

Primjer:

```
int main()
{
   char i;
   i = '1';
   printf("%c\n", i);   /* 1 */
   printf("%d\n", i);   /* 49 */
   return 0;
}
```

Nenumerički tipovi podataka (2)

Logička ili Booleova algebra:

- Logičke vrijednosti prikazuju se bitovima

  laž = `0`, istina = `1`,

  koje možemo promatrati i kao cijele brojeve.
- Osnovne operacije **ne**, **i, ili** (engl. not, and, or) svode se na aritmetičke u bazi `2`.
- Logičke operacije mogu se izvesti i **bit-po-bit** na čitavim skupinama bitova u nekoj većoj cjelini.
- Logička algebra služi za formulaciju i kombiniranje uvjeta u uvjetnim naredbama.

Nenumerički tipovi podataka (3)

Primjer:

```
int main( )
{
   int i = 10, j = 20;

   printf("%d\n", i < j);   /* 1 */
   return 0;
}
```

Numerički tipovi podataka

Numerički tipovi podataka moraju realizirati

- četiri osnovne aritmetičke operacije na raznim skupovima brojeva.

Osnovni problem: Standardni skupovi brojeva u matematici

$$\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$$

su beskonačni i ne možemo ih prikazati u računalu.

Umjesto toga, u računalu prikazujemo samo neke konačne podskupove odgovarajućeg matematičkog skupa.

Numerički tipovi podataka (2)

- Konačni podskup je model „beskonačnog” skupa.

Numeričke tipove možemo podijeliti u tri grupe, prema beskonačnom skupu kojeg „modeliramo”:

- „cijeli” brojevi bez predznaka – model za $\mathbb{N} \cup \{0\}$
- „cijeli” brojevi s predznakom – model za $\mathbb{Z}$
- „realni” brojevi – model za $\mathbb{R}$

Navodnici naglašavaju da su pripadni „prikazivi” skupovi brojeva konačni.

Numerički tipovi podataka (3)

- Svaka grupa ima nekoliko podtipova ovisno o veličini pripadnog konačnog skupa prikazivih brojeva.
- Prijelaz na konačne skupove bitno mijenja realizaciju aritmetike na odgovarajućem skupu. Aritmetika se **ne nasljeđuje** projekcijom s originalnog skupa.

Numerički tipovi podataka (4)

Za potpuni opis numeričkih tipova moramo još opisati:

- koji konačni skupovi brojeva modeliraju odgovarajuće matematičke skupove
- kako se točno prikazuju njegovi elementi u računalu
- kako se realizira aritmetika na tim skupovima.

Cijeli brojevi bez predznaka

- Cijeli brojevi bez predznaka modeliraju skup $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.
- U računalu se prikazuje najveći početni komad skupa $\mathbb{N}_0$.

  Ako na raspolaganju imamo n bitova za prikaz, onda je skup prikazivih brojeva jednak

  $$\{0, 1, 2, \ldots, 2^n - 2, 2^n - 1\}.$$

  Veće brojeve ne možemo prikazati pomoću n bitova.

Cijeli brojevi bez predznaka (2)

Primjer:

n	2^n-1
8	255
16	65 535
32	4 294 967 295

Kako stvarno izgleda prikaz tih brojeva u n bitova?

Prikaz pojedinih (prikazivih) brojeva je

- doslovna „kopija” prikaza tih brojeva u pozicijskom zapisu u bazi 2.

Cijeli brojevi bez predznaka (3)

Primjer:

Neka je $n=8$. Pogledajmo zapis broja 123.

$$123<255=2^8-1,$$

pa je 123 prikaziv. Nadalje, njegov binarni prikaz je

$$123=64+32+16+8+2+1$$

$$=0\cdot 2^7+1\cdot 2^6+1\cdot 2^5+1\cdot 2^4+1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+1\cdot 2^0.$$

Dakle, broj 123 kao cijeli broj bez predznaka ima prikaz

$$123 \longleftrightarrow 01111011.$$

- Cjelobrojni tip bez predznaka u programskom jeziku C: `unsigned`

Aritmetika cijelih brojeva bez predznaka

Aritmetika cijelih brojeva bez predznaka s n bitova za prikaz brojeva je tzv. modularna aritmetika, ili, preciznije

- aritmetika ostataka modulo 2^n.

To znači da aritmetičke operacije +, - i * na skupu cijelih brojeva bez predznaka daju rezultat koji je

- jednak ostatku rezultata pripadne cjelobrojne operacije (u skupu $\mathbb{Z}$) pri dijeljenju s 2^n.

Aritmetika cijelih brojeva bez predznaka

Primjer:

```
int main( )
{
   unsigned short i = 65535;

   printf("%d\n", i/10);   /* 6553 */

   i = i + 3;
   printf("%d\n", i);   /* 2 */
   return 0;
}
```

Aritmetika cijelih brojeva bez predznaka

Primjer:

```
int main( )
{
   unsigned short i = 2, j = 4;

   printf("%d\n", i - j);

   i = i - j;
   printf("%d\n", i);

   return 0;
}
```

Cijeli brojevi s predznakom

- Cijeli brojevi s predznakom modeliraju skup cijelih brojeva.
- Ako na raspolaganju imamo n bitova, onda skup prikazivih brojeva ima 2^n elemenata.
- U računalu se prikazuje najveći mogući podskup uzastopnih brojeva iz $\mathbb{Z}$ koji je „gotovo” simetričan oko nule.

Ako na raspolaganju imamo n bitova za prikaz, onda je skup prikazivih brojeva jednak

$$\{-2^{n-1}, -2^{n-1}+1, \ldots, -1, 0, 1, 2, \ldots, 2^{n-1}-2, 2^{n-1}-1\}.$$

Cijeli brojevi s predznakom (2)

- Brojeve izvan tog skupa ne možemo prikazati sa samo n bitova.
- Najmanji i najveći prikazivi cijeli broj s predznakom su, redom -2^{n-1}, $2^{n-1}-1$.
- Tipične vrijednosti za ta dva granična broja su:

n	-2^{n-1}	$2^{n-1}-1$
8	-128	127
16	-32 768	32 767
32	-2 147 483 648	2 147 483 647

INT_MIN= -2 147 483 648, INT_MAX= 2 147 483 647

Cijeli brojevi s predznakom (3)

Kako stvarno izgleda prikaz tih brojeva u `n` bitova?

- Nenegativni brojevi imaju isti prikaz kao i cijeli brojevi bez predznaka.
- Negativni brojevi se prikazuju tehnikom dvojnog komplementa: nule pretvaramo u jedinice i jedinice u nule, a zatim tom komplementu dodajemo `1`.

Cijeli brojevi s predznakom (4)

Primjer: `n = 8.`

45

0	0	1	0	1	1	0	1

1	1	0	1	0	0	1	0

-45

1	1	0	1	0	0	1	1

Primjer:

$-2^{n-1} \longleftrightarrow 1000\ldots0$

$-1 \longleftrightarrow 1111\ldots1$

$2^{n-1}-1 \longleftrightarrow 0111\ldots1$

Cijeli brojevi s predznakom (5)

Primjer: `n = 3`.

0	000
1	001
2	010
3	011
-4	100
-3	101
-2	110
-1	111

Aritmetika cijelih brojeva s predznakom

- Cjelobrojni tip u programskom jeziku C: `int`

Primjer:

```
int main( ){
   short int i;
   i = 32766;
   i += 1;
   printf("%d\n", i);   /* 32767 */
   i += 1;
   printf("%d\n", i);   /* -32768 */
   return 0;
}
```

Aritmetika cijelih brojeva s predzn. (2)

Primjer:

```
50!=30414093201713378043612608166064768844377
641568960512000000000000

#include <stdio.h>
int main( ){
    int i, f50 = 1;
    for(i = 2; i <= 50; i++)
            f50 *= i;
    printf("f50 = %d\n", f50);          /* f50 = 0 */
    return 0;
}
```

Dijeljenje cijelih brojeva s predznakom

- Za nenegativne brojeve s predznakom dobivamo očekivane (i korektne) rezultate pri cjelobrojnom dijeljenju.
- A za negativne operande? Isti program može davati različite rezultate ovisno o računalu i izboru C kompajlera!

Primjer: rezultati za `q = a/b` i `r = a%b` za `a = ±5`, `b = ±3`.

Dijeljenje cijelih brojeva s predznakom

a	b	q	r
5	3	1	2
-5	3	-1	-2
5	-3	-1	2
-5	-3	1	-2

Najčešća realizacija:

- kvocijent se uvijek „zaokružuje” prema nuli
- ostatak ima isti predznak kao `a`.

Dijeljenje cijelih brojeva s predzn. (2)

Primjer: broj znamenki zadanog broja.

```
#include <stdio.h>
int main( ){
    int n = 1073489, broj;
    broj = 0;
    while(n != 0){
        broj += 1;
        n /= 10;}
    printf("broj_znamenki = %d\n", broj);
    return 0;
}
```