12. Číselné soustavy, binární číselná soustava. Kódování informací, binární váhový kód, kódování záporných čísel. Standardní jednoduché datové typy s pevnou a s pohyblivou řádovou tečkou. Základní strukturované datové typy (pole, rekord apod.). Paměť počítače, adresa, uložení základních datových typů v paměti počítače.

Číselné soustavy

Každou hodnotu lze reprezentovat různými způsoby. V historii se používali jako počítadla ruce a proto první číselné soustavy měli základ pět a deset. Dnes se v běžném životě nejvíce používá soustava o základu deset, takzvaná Arabská (vymyšlena ale byla v Indii). Tato soustava používá číslice 0..9, přičemž poloha číslice určuje jeho váhu - tím je možné vyjádřit i čísla větší než deset.

Číslo v desítkové soustavě můžeme obecně rozepsat takto:

$$N_{10}=a_n\times10^n+a_{n-1}\times10^{n-1}+...+a_1\times10^1+a_0\times10^0+a_{-1}\times10^{-1}+...+a_{-m}\times10^{-m}$$

Příklad:
156,2 = 1 x 100 + 5 x 10 + 6 x 1 + 2 x 0,1

Počítače mohou z principu pracovat s libovolnou číselnou soustavou. Protože je však nejjednodušší a nejspolehlivější rozlišovat pouze dva stavy nějaké fyzikální veličiny, používá soustava o základu dva. Tyto

dva stavy se mohou nazývat různě:

0 – 1

lež – pravda

logická 0 – logická 1

V reálném světě se pracuje nejčastěji s napětím. Buňce, která nese takovou informaci, říkáme jeden bit. Protože zápis čísel v soustavě o základu dva je pro člověka nepřehledný, používají se ve výpočetní technice ještě soustavy o základu osm a šestnáct. Mezi těmito soustavami a dvojkovou soustavou lze totiž snadno převádět a pro člověka jsou čísla v těchto soustavách mnohem „představitelnější“.

Pro některé speciální případy lze používat i jiné typy číselných systémů. Např. pomocí číselné soustavy s faktoriály, kde každá pozice má váhu n!, lze velmi přesně pracovat s racionálními čísly.

V určitých případech potřebujeme předem vědět, kolik platných míst (N) budeme potřebovat k vyjádření nějaké hodnoty (X) v různých číselných soustavách. Výpočet provedeme pomocí logaritmu o základu, který odpovídá základu soustavy:
Pro dekadickou soustavu: $N = [\log_{10} X] + 1$
Pro binární soustavu: $N = [\log_{2} X] + 1$
Pro hexadecimální soustavu: $N = [\log_{16} X] + 1$
Výsledek zaokrouhlujeme na celé číslo směrem dolů.

Binární číselná soustava

- používá pouze dva symboly: 0 a 1
- nejjednodušší soustava

Převod z dvojkové soustavy do dekadické

číslo ve dvojkové soustavě:

$$N_2 = a_n \times 2^n + a_{n-1} \times 2^{n-1} + \ldots + a_1 \times 2^1 + a_0 \times 2^0 + a_{-1} \times 2^{-1} + \ldots + a_{-m} \times 2^{-m}$$

př:

$$101101{,}01_2 = 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 + 0 \times 0{,}5 + 1 \times 0{,}25$$

tedy:

$$101101{,}01_2 = 45{,}25$$

Tento postup je velmi rychlý a lze ho provádět pro malý počet míst i přímo z hlavy. Stačí si pamatovat mocniny dvojky (= váhy pozic).

Převod z dekadické soustavy do dvojkové

Matematické odvození postupu:

Napíšeme si rovnici, kde na levé straně je zadaná hodnota v desítkové soustavě a na pravé straně je obecně rozepsané číslo ve dvojkové soustavě.

Př:

$$132 = \ldots + a_4 \times 2^4 + a_3 \times 2^3 + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0$$

Toto je jedna rovnice o mnoha neznámých. Abychom jí mohli vyřešit, musíme jí rozdělit na rovnice pro jednotlivé proměnné. To provedeme tak, že celou rovnici vydělíme číslem 2:

$$\begin{aligned} 132 &= \ldots + a_4 \times 2^4 + a_3 \times 2^3 + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0 \quad \div 2 \\ 66{,}0 &= \ldots + a_4 \times 2^3 + a_3 \times 2^2 + a_2 \times 2^1 + a_1 \times 2^0 + a_0 \times 2^{-1} \end{aligned}$$

Výsledek poté rozdělíme na celou část a část za desetinou čárkou.
Desetinná část:

$$0 = a_0 \times 2^1 = a_0 \times 0{,}5$$

V tomto okamžiku již můžeme přímo napsat, že koeficient $a_0 = 0$.
Celá část:

$$66{,}0 = \ldots + a_4 \times 2^3 + a_3 \times 2^2 + a_2 \times 2^1 + a_1 \times 2^0$$

Se zbytkem rovnice opakujeme tento postup až do doby, kdy na levé straně zůstane 0. V tomto okamžiku máme vypočteny všechny koeficienty.

$$\begin{aligned} 66 &= \ldots + a_4 \times 2^3 + a_3 \times 2^2 + a_2 \times 2^1 + a_1 \times 2^0 \quad \div 2 \\ 33{,}0 &= \ldots + a_4 \times 2^2 + a_3 \times 2^1 + a_2 \times 2^0 + a_1 \times 2^1 \\ 0 &= a_0 \times 2^{-1} \quad \Rightarrow a_1 = 0 \end{aligned}$$

$$\begin{aligned} 33 &= \ldots + a_4 \times 2^2 + a_3 \times 2^1 + a_2 \times 2^0 \quad \div 2 \\ 16{,}5 &= \ldots + a_4 \times 2^1 + a_3 \times 2^0 + a_2 \times 2^{-1} \\ 0{,}5 &= a_2 \times 2^{-1} \quad \Rightarrow a_2 = 1 \end{aligned}$$

$$\begin{aligned} 16 &= \ldots + a_4 \times 2^1 + a_3 \times 2^0 \quad \div 2 \\ 8{,}0 &= \ldots + a_4 \times 2^0 + a_3 \times 2^{-1} \\ 0 &= a_3 \times 2^{-1} \quad \Rightarrow a_3 = 0 \end{aligned}$$

$$\ldots$$

$$132 = 10000100_2$$

Tento postup je poněkud zdlouhavý, proto se používá jeho rychlejší varianta. Dělíme desítkové číslo

celočíselně dvěma a opisujeme zbytky.
Příklad:
132 : 2 = 66 : 2 = 33 : 2 = 16 : 2 = 8 : 2 = 4 : 2 = 2 : 2 = 1 : 2 = 0
zbytek: 0 0 1 0 0 0 0 1
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7
Dělíme do té doby, dokud výsledek není nula a zbytky opisujeme křížem (od a_7 do a_0).
Desetinou část čísla převádíme obdobně. Pouze používáme pro řešení rovnice místo dělení násobení.
Př:

$$0{,}22 = a_{-1}\times 2^{-1} + a_{-2}\times 2^{-2} + a_{-3}\times 2^{-3} + a_{-4}\times 2^{-4} + a_{-5}\times 2^{-5} + \ldots \quad \times 2$$
$$0{,}44 = a_{-1}\times 2^{0} + a_{-2}\times 2^{-1} + a_{-3}\times 2^{-2} + a_{-4}\times 2^{-3} + a_{-5}\times 2^{-4} + \ldots$$
$$0 = a_{-1}\times 2^{0} \Rightarrow a_{-1} = 0$$

$$0{,}44 = a_{-2}\times 2^{-1} + a_{-3}\times 2^{-2} + a_{-4}\times 2^{-3} + a_{-5}\times 2^{-4} + \ldots \quad \times 2$$
$$0{,}88 = a_{-2}\times 2^{0} + a_{-3}\times 2^{-1} + a_{-4}\times 2^{-2} + a_{-5}\times 2^{-3} + \ldots$$
$$0 = a_{2}\times 2^{0} \Rightarrow a_{2} = 0$$

$$0{,}88 = a_{-3}\times 2^{-1} + a_{-4}\times 2^{-2} + a_{-5}\times 2^{-3} + \ldots \quad \times 2$$
$$1{,}76 = a_{-3}\times 2^{0} + a_{-4}\times 2^{-1} + a_{-5}\times 2^{-2} + \ldots$$
$$1 = a_{-3}\times 2^{0} \Rightarrow a_{3} = 1$$

Stejně jako u převodu celé části lze postup zrychlit.
Př:
0,25 x 2 = 0,5 x 2 = 1 (-1) 0 x 2 = 0
0 1
a_{-1} a_{-2}
Jako výsledek opisujeme celé části výsledku po násobení dvěma. V našem případě je tedy výsledek po zaokrouhlení:
$0{,}25 = 0{,}01_2$

Rozvoj necelé části může být i neukončený (periodický nebo neperiodický). V tomto případě musíme rozvoj zkrátit (zaokrouhlit) a dochází tak k zaokrouhlovací chybě.

Př:

0,22 x 2 = 0,44 x 2 = 0,88 x 2 = 1,76 (-1) 0,76 x 2 = 1,52 (-1) 0,52 x 2 = 1,04 (-1) 0,04 x 2 = 0,08 …
0 0 1 1 1 0
$0{,}22 = 0{,}00111000_2$
S čísly ve dvojkové soustavě lze přímo počítat obdobně, jako s čísly v desítkové soustavě.

Binární váhový kód

Binární kód je způsob uložení informace v počítači definovaný jako konečný počet bitů, z nichž každý může nabývat právě jednu ze dvou hodnot (obvykle označených 0 nebo 1). Pro snadnější zápis uložených hodnot (čísel) se dnes převážně používá byte (bajt) s délkou slova osm bitů. Pro výpočet hodnoty binárního zápisu se používá binární soustava.

Kódování záporných čísel

Pro zápis záporných čísel ve dvojkové soustavě můžeme použít znaménko mínus stejně, jako v desítkové soustavě. Pro kódování znaménka v počítačích se proto používají tyto metody:

- Vyhrazení jednoho bitu pro znaménko, další bity zůstávají pro binární váhový kód absolutní hodnoty (např. mantisa u typů s pohyblivou řádovou čárkou)
- Přičtení konstanty (např. exponent u typů s pohyblivou řádovou čárkou)
- Pomocí dvojkového doplňku (např. celočíselné typy ve vyšších programovacích

jazycích)

Dvojkový doplněk

Doplňkem se rozumí rozdíl kapacity soustavy (tj. 2^n, kde n je počet bitů v binárním vyjádření) a absolutní hodnoty čísla. Rozsah čísel, které tímto způsobem lze vyjádřit, je pak (-2^{n-1}) až ($2^{n-1} - 1$). Např. Pomocí osmi bitů tak lze vyjádřit čísla v rozsahu –128 .. +127 .

Interpretace záporných čísel pomocí dvojkového doplňku umožňuje, na rozdíl od ostatních způsobů, přímo provádět sčítání a odečítání. Zároveň lze také přímo zjistit znaménko, protože nejvyšší bit u záporných čísel je vždy jedna.

Postup pro převod absolutní hodnoty na záporné číslo pomocí dvojkového doplňku je následující:

- doplníme dvojkové vyjádření absolutní hodnoty zleva nulami na požadovaný počet bitů
- provedeme negaci všech bitů
- k výsledné hodnotě přičteme binárně hodnotu jedna

Alternativní postup:

- doplníme dvojkové vyjádření absolutní hodnoty zleva nulami na požadovaný počet bitů
- zprava opíšeme všechny nuly až k první jedničce (včetně), další bity negujeme

př:

-46

$46 = 00101110_2$ Nejprve převedeme desítkovou absolutní hodnotu a doplníme nulami zleva na osm bitů

11010001_2 Znegujeme výsledek

11010010_2 Přičteme k výsledku jedničku

Výsledek: $-46 = 11010010_2$

Standardní jednoduché datové typy

- s pevnou řádovou tečkou
  - neznaménkové
    - byte (8b)
    - word (16b)
    - Dword (32b)
    - Char (8b)
  - znaménkové
    - shortInt (8b)
    - integer (16b)
    - longint (32b)
- s plovoucí řádovou tečkou
  - single (32b)
  - double, real (64b)
  - extended (80b)

Datové typy s pevnou řádovou tečkou

Jsou to nejjednodušší celočíselné typy, pomocí kterých lze vyjadřovat celá čísla. Můžeme je dále rozdělit na znaménkové a neznaménkové. Rozsah hodnot, které pomocí nich lze vyjádřit je dán jejich velikostí v pamětí (počtem bitů).

Typ	Rozsah
Byte	0 .. 255
Word	0 .. 65535
Dword	0 .. $2^{32}-1$
ShortInt	-128 .. 127
Integer	-32768 .. 32767
LongInt	-2^{32} .. $2^{32}-1$
Char	0..255 – odkaz do ASCII tabulky

Datové typy s plovoucí řádovou tečkou

Abychom mohli pracovat s reálnými čísly ve dvojkové soustavě, můžeme využít stejný přístup, jako v desítkové. To znamená, že si číslo převedeme do dvojkové soustavy a případně ještě do normalizovaného tvaru.

Př:

Převeďte číslo $2{,}1345 \times 10^{2}$ do dvojkové soustavy v normalizovaném tvaru.

Řešení:

Nejprve si převedeme desítkové číslo z normalizovaného tvaru do tvaru, který je vhodný pro převod do dvojkové soustavy (to je tvar bez exponentu). Provedeme převod celé i desetinné části a nakonec přepíšeme výsledek do normalizovaného tvaru ve dvojkové soustavě (přidáme exponent tak, aby před desetinou čárkou bylo pouze číslo 1).

$2{,}1234 \times 10^{3} = 2123{,}4 = 100001001011{,}011001100_2 = 1{,}00001001011011001100_2 \times 2^{11}$

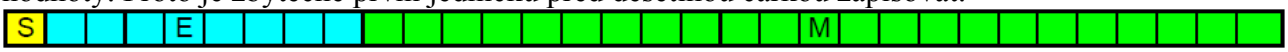
Pokud už máme reálné číslo ve dvojkové soustavě v normalizovaném tvaru, můžeme se zabývat problémem jak ho uložit do paměti počítače. Ve skutečnosti totiž musíme uložit tři různé informace – znaménko, absolutní hodnotu a exponent. Musíme si zvolit způsob kódování znaménka u exponentu, počet bitů u exponentu a počet bitů na vlastní číslo (mantisu). Tím určíme rozsah a přesnost čísel, která můžeme takto kódovat.

Je zřejmé, že možností je velmi mnoho. Ve vyšších programovacích jazycích se používají hlavně tyto tři datové typy: Single, Double (někdy se označuje jako Real), Extended.

Datový typ Single

Pro zápis reálného čísla v tomto datovém typu se využívá celkem 32 bitů. Jeden bit je znaménko, osm bitů je vyhrazeno pro exponent a 23 bitů pro mantisu. Mantisa je absolutní hodnota reálného čísla v normalizovaném tvaru, ale zapisuje se bez první jedničky. V normalizovaném tvaru totiž

binární číslo vždy začíná jedničkou, pak je desetinná čárka a pak následuje zbytek absolutní hodnoty. Proto je zbytečné první jedničku před desetinou čárkou zapisovat.



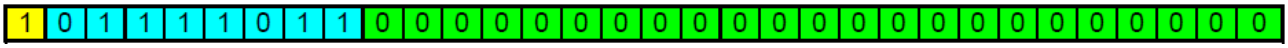
S – znaménko, 1=mínus
E – exponent, kóduje se přičtením konstanty 127, takže rozsah exponentů je –127 až +128
M – mantisa, absolutní hodnota čísla, zapisuje se bez první jedničky

Zpětně pak dostaneme reálné číslo dosazením do vzorce:
$(-1)^{S} \times 2^{E-127} \times (1,M)$

Postup pro vytvoření 32-bitové konstrukce:

1. Vyjádřit absolutní hodnotu daného čísla *X* v binární soustavě (odděleně zjistit binární vyjádření celé a
necelé části, potom obě části oddělit řádovou tečkou)
2. Řádovou tečku posunout za první jedničku zleva. Z počtu pozic, o které se tečka v zápisu posouvá, určit
hodnotu EXPONENTU:
Žádný posun *e* = 0
Posun vpravo *e* <0
Posun vlevo *e* > 0
3. Určit obsah pole S:
X >=0 S=0
X < 0 S=1
4. Určit obsah pole E: Vyjádřit hodnotu *e+127* binárním váhovým kódováním (jako typ Byte)
5. Určit obsah pole M: bity, které zůstaly po posunutí řádové tečky vpravo od ní.

Příklad:
Uložte do paměti číslo –0,0625 ve formátu datového typu Single.
Řešení:
Nejprve převedeme absolutní hodnotu do dvojkové soustavy.
$0,0625 = 0,0001_2$
Poté přepíšeme výsledek do normalizovaného tvaru posunem desetinné čárky.
$0,0001_2 = 1,0_2 \times 2^{-4}$
Obsah pole S je 1, protože číslo je záporné.
Obsah pole E je binárně vyjádřený exponent, ke kterému přičteme konstantu 127.
$E = -4 + 127 = 124 = 01111011_2$
Mantisu získáme opsáním absolutní hodnoty bez první jedničky, takže to jsou v tomto případě samé nuly.



Rekonstrukce číselné hodnoty:
1. 32-bitovou strukturu rozdělit na pole S (1 bit), E (8), M (23)
2. Sestavit zápis . **1.** ...., ve kterém za řádovou tečkou jsou bity z pole M.
3. Číselně interpretovat obsah pole E (binární váhový kód). Zmenšením získané hodnoty o 127 určit hodnotu *e*. V zápisu posunout řádovou tečku o *e* pozic. (Pro *e*>0 se tečka posouvá směrem vlevo).
4. Zápis v binární soustavě převést do dekadické soustavy (binární váhový kód)
5. Doplnit znaménko podle bitu v poli S. (S=1 . číslo je záporné)

př:

S = 0

$E = 10000010_2 = 129$
$M = 1001001_2$
Výsledek je pak:
$2^{130-127} \times 1{,}1001001_2 = 2^3 \times 1{,}1001001_2 = 1100{,}1001_2 = 12{,}5625$

vyjímky:

- Pokud S = 0, E = 0 a M = 0 : hodnota čísla je nula
- Pokud E = -127 : EXP se bere jako –126 a absolutní hodnota se nebere ve tvaru (1,M) ale (0,M)
- Pokud E = 128 : hodnota čísla je nekonečno, M musí být 0. Jiné hodnoty M jsou neplatné.

	Exponent	Mantisa	Velikost	MIN	MAX	Přesnost
Single	8b	23b	32b	1,4E-45	3,4E38	6-7 míst
Double	11b	52b	64b	5,0E-324	17,E308	15-16 míst
Extended	15b	63b	80b	3,4E-4932	1,1E4932	18-19 míst

Strukturované datové typy

Datový typ obsahuje jeden nebo více prvků. Říkáme, že je homogenní, jsou-li prvky stejného typu.
Pole

sdružuje daný počet prvků (čísel, textových řetězců, …) o stejné velikosti. K jednotlivým prvkům pole se přistupuje pomocí jejich indexu (celého čísla, označujícího pořadí prvku). Velikost pole zůstává při běhu programu neměnná (některé programovací jazyky toto omezení nekladou, zvětšení pole je ale časově náročná operace).

Řetězec

Počet znaků řetězce definuje délku textového řetězce. Textový řetězec může být prázdný (obsahuje 0 znaků řetězce).

Typy

- konstantní – neměnný obsah (generovaný při překladu programu)
- staticky alokovaný pamětový prostor pro řetězec – řetězec má omezenou max. Délku
- dynamicky alokovaný pamětový prostor pro řetězec – řetězec má max. délku omezenou jen velikostí volné paměti

Záznam

Tento datový typ se skládá z určitého počtu položek, které mohou být různého typu. každá položka má přiděleno jméno, neboli identifikátor položky, pomocí něhož se provádí výběr položky.

Výčtový typ (enum)

programátorem definovaný typ, např. pro barvu karet.

Paměť

Společná pro data i program (viz. Von Neumann)

- Slouží k uchování dat
- je potřeba
  - prostor pro uložení dat (paměťové buňky)
  - nástroj na přístup (adresa, sběrnice)
- každá buňka má svou adresu

Operační paměť je volatilní (nestálá) vnitřní elektronická paměť číslicového počítače typu RWM-RAM, určená pro dočasné uložení zpracovávaných dat a spouštěného programového

kódu. Tato paměť má obvykle rychlejší přístup než vnější paměť (např. pevný disk). Tuto paměť může procesor adresovat přímo, pomocí podpory ve své instrukční síti. Strojové instrukce jsou adresovány pomocí instrukčního ukazatele a k datům se obvykle přistupuje pomocí adresace prvku paměti hodnotou uloženou v registru procesoru nebo je adresa dat součástí strojové instrukce.

Operační paměť je spojena s procesorem pomocí sběrnice, obvykle se mezi procesor a operační paměť vkládá rychlá vyrovnávací paměť typu cache, neboli paměť, která je přímo přístupná procesoru. Jedná se o nepostradatelný fyzický prostředek, který je spravován jednou z hlavních části operačního systému. Operační paměť je určená pro uchovávání kódu programů respektive procesů spolu s mezivýsledky a výsledky jejich činnosti. Zrovna tak je v operační paměti uchováván stav dalších prostředků a základní datové struktury jádra.

Vezmeme-li v potaz adresování operační paměti, lze zjednodušeně paměť chápat jako souvislý prostor paměťových buněk o nějaké velikosti. Tyto buňky jsou pak lineárně adresovány adresami pevné délky.

Je-li operační paměť reprezentována pamětí s přímým přístupem, označujeme adresový prostor jako fyzický adresový prostor (FAP). Velikost tohoto prostoru je omezena buď fyzickou velikostí paměťových modulů a nebo velikostí adresy tj. adresa o velikosti n bitů umožňuje adresovat 2 na n-tou paměťových míst.

Jednodušší procesory podporují adresovat pouze paměť s přímým přístupem, tedy adresovat pouze fyzický adresový prostor. V dnešní době velká část procesorů umožňuje adresovat i logické adresové prostory. Jedná se o tzv. virtualizaci paměti respektive jde o neomezený počet logických adresových prostorů.

Adresa

- Adresa = číslo, pořadí paměťové buňky
  - Udává se nejčastěji v šestnáctkové soustavě (např. adresa 2F9 je buňka na 761. pozici)
- Adresový prostor = rozsah adres
- Adresový prostor procesoru: rozsah adres, na které dokáže přistupovat procesor
  - Např. 8bitový procesor neumí pracovat s 16bit. Adresou
- Paměti většího rozsahu (disky apod.):
  - Děleny do segmentů; stránek a dalších celků
  - Adresa je kombinací určení segmentu a adresy buňky

Uložení základních datových typů v paměti počítače

Pro zjednodušení budeme o paměti uvažovat jako tabulce, která má na každém řádku osmibitové číslo. Adresa 0 je nahoře a každý další řádek má adresu o 1 vyšší. Překladače vyšších programovacích jazyků ukládají proměnně postupně do paměti v pořadí, jak jsou deklarovány. Proměnné, které mají velikost jeden bajt (Byte, Char) je situace jednoduchá. Více bajtové proměnné se ukládají do paměti postupně od nejméně významného bajtu.

Př typ single: