Reprezentácia údajov

V pamäti počítača sú uložené

- **inštrukcie** (kódy inštrukcií)
- **adresy** (bývajú súčasťou inštrukcie), programátor používa adresy (smerníky na údaje a funkcie (podprogramy))
- **údaje**

Čísla

- počítač všetky druhy informácií uchováva v konečnom počte bajtov
- môže uchovávať len **celé** čísla a **racionálne** čísla **(** nie je možné zobraziť presne iracionálne číslo napr. π, $\sqrt{2}$)
- **obmedzený rozsah** (nemožno zobraziť ľubovolne malé alebo veľké číslo)
- **presnosť zobrazenia** racionálnych čísel je **obmedzená**

Čísla sú kódované v **binárnej sústave** (konkrétna realizácia bude prebraná ďalej)

Celé čísla (násobky bajtov)

	16-bitové C	32-bitové C	Min	Max
1B	char	char	-128	127
	unsigned char	unsigned char	0	255
2B	int (short int)	short int	-32 768	32 767
	unsigned int	unsigned short	0	65 535
4B	long int	int (long int)	-2 147 483 648	2 147 483 647
	unsigned long	unsigned int (long)	0	4 294 967 295
8B	-9 223 372 036 854 775 808		9 223 372 036 854 775 807	

Racionálne čísla

4B	float	$1.18*10^{-38}$	$3.40*10^{38}$
8B	double	$2.23*10^{-308}$	$1.79*10^{308}$
10B	long double	$3.37*10^{-4932}$	$1.18*10^{4932}$

Logické informácie

Najmenší element 1bajt (strojové slovo), agregácia logických premenných do dátových typov, ktoré sú násobkami bajtov

Používajú sa napr. na uchovávanie informácií o stave

- procesora (stavové slovo)
- stavu programu, periférnych zariadení
- stavu procesov alebo zariadení riadených počítačom

Interpretácia významu jednotlivých bitov je daná konvenciou

Príklad : ohrievač vody, pri ktorom sledujeme resp. zapamätávame napr. nasledovné stavy:

- stav vyhrievacieho telesa — 0- zapnuté 1- vypnuté K
- stav ventilu prívodu vody — 0- zatvorený 1- otvorený V
- spínač max. výšky hladiny vody — 0- vypnutý 1- zapnutý H
- spínač min. výšky hladiny vody — 0- vypnutý 1- zapnutý D
- spínač prekročenia max. teploty vody — 0- vypnutý 1- zapnutý M
- spínač prekročenia min. teploty vody — 0- vypnutý 1- zapnutý N

Potom môžeme uložiť informáciu o stave ohrievača do bajtu napr. v štruktúre

XXDH MNVK

Kde symbolom X sú označené nevýznamné bity
Potom dekódovaním bajtu napr.:

XXDH MNVK = 0010 0000

Zistíme informácie:

D = 1	v ohrievači je málo vody
M = 0 & N = 0	teplota vody je v norme
V = 0	prívodný ventil je zatvorený
K = 0	voda sa nezohrieva

Z informácií môžeme rozhodnúť - pretože D=1, treba otvoriť prívodný ventil.

Niektoré kompilátory podporujú štruktúry s bitovými poliami, potom sa dá definovať napr. :

struct vymeník

```
{
 unsigned vyhrievanie  : 1;
 unsigned              : 2;
// neobsadené bity
 unsigned ventil       : 1;
 unsigned hladina_max  : 1;
 unsigned hladina_min  : 1;
 unsigned teplota_max  : 1;
 unsigned teplota_min  : 1;
};
```

Formálne sa s položkami pracuje ako pri štandartných štruktúrach.

Znaky a znakové reťazce (stringy)

- BCD - 5 bit kód
- Fieldata - 6 bit kód
- ASCII - 7 bit kód (celkovo 128 znakov)
  (***A**merican **S**tandard **C**ode **for I**nformation **I**nterchange*)
  kódovanie znakov – základom anglická abeceda
  - prvýkrát zverejnené v roku 1967
  - posledná úprava v roku 1986
  - obsahuje 32 nezobraziteľných znakov (väčšinou už nepoužívaných riadiacich znakov)
  - 96 zobraziteľných znakov
  - spolu 128 znakov - 7 bitový kód (8. bit na paritu)

A = 0x41, a = 0x61

0 = 0x30

0x41 = 0100 0001

NP = 0xC1 = 1100 0001

~~Parita~~

[1] str. 125 Tab. 3.11 Kód ASCII

Riadiace znaky: $<0_{16}$ až 20_{16})

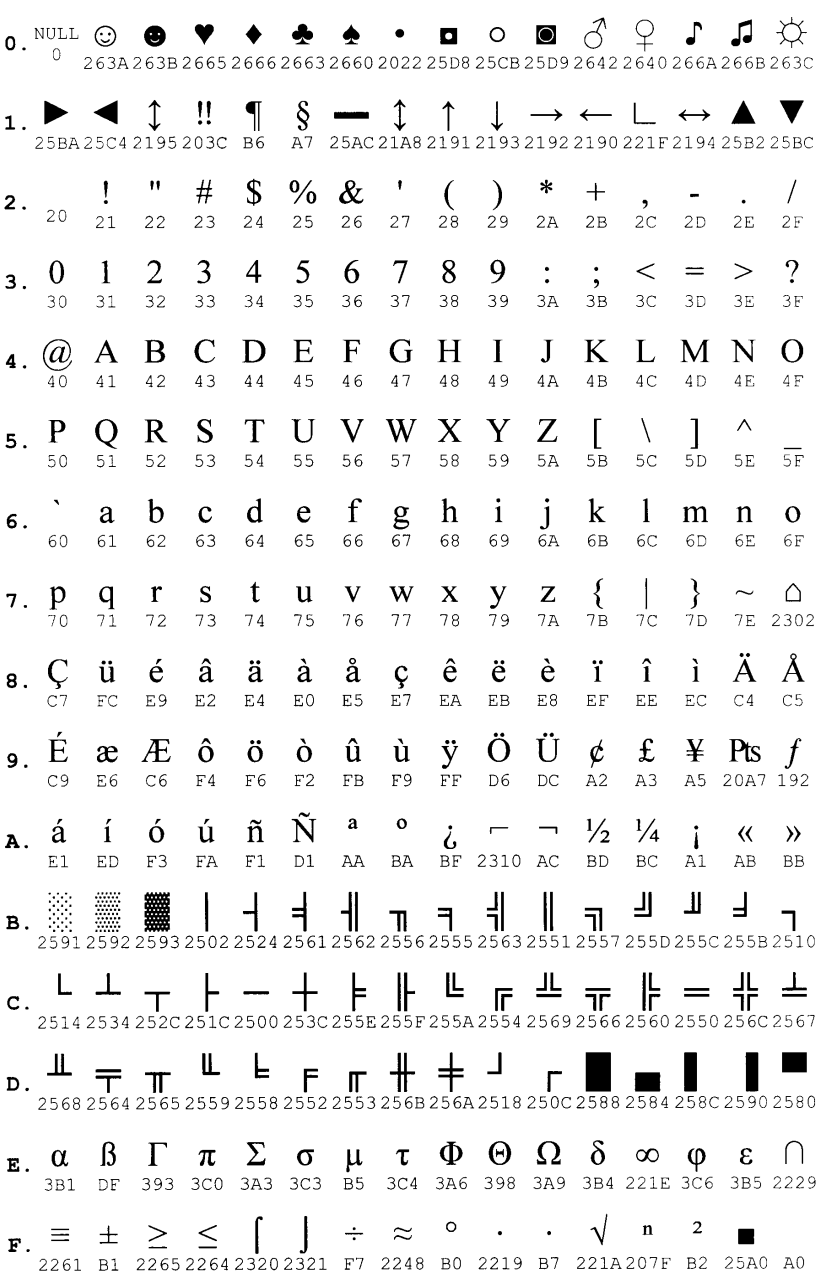
carriage return - CR, line feed - LF

Extended ASCII (high ASCII) – využitie 8. bitu ⇒ ďalších 128 znakov (špeciálne symboly, znaky národných abecied, grafické symboly). Toto označenie sa používa aj pre viacbajtové reprezentácie.

Kódové stránky :
IBM PC resp. MS-DOS code page 437 (**CP437, DOS-US** or **OEM-US)**
Pôvodne znaková sada pre IBM PC (1981) (prevzaté z WANG)

CP 850(**DOS Latin-1**) (West-European) postupne nahradený **ISO-8859-1** (znaková sada pre HTML, XML) a **Windows-1252**
CP 852 Central-European) **Windows-1250**
CP737 (DOS Greek)

IBM PC DOS CP 473



IBM PC DOS CP 852

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
8.	Ç	ü	é	â	ä	ů	ć	ç	ł	ë	Ő	ő	î	Ź	Ä	Ć
	C7	FC	E9	E2	E4	16F	107	E7	142	EB	150	151	EE	179	C4	106
9.	É	Ĺ	Í	ô	ö	Ľ	ľ	Ś	ś	Ö	Ü	Ť	ť	Ł	×	č
	C9	139	13A	F4	F6	13D	13E	15A	15B	D6	DC	164	165	141	D7	10D
A.	á	í	ó	ú	Ą	ą	Ž	ž	Ę	ę	¬	ź	Č	ş	«	»
	E1	ED	F3	FA	104	105	17D	17E	118	119	AC	17A	10C	15F	AB	BB
B.	▒	▒	▓	│	┤	Á	Â	Ě	Ş	╣	║	╗	╝	Ż	ż	┐
	2591	2592	2593	2502	2524	C1	C2	11A	15E	2563	2551	2557	255D	17B	17C	2510
C.	└	┴	┬	├	─	┼	Ă	ă	╚	╔	╩	╦	╠	═	╬	¤
	2514	2534	252C	251C	2500	253C	102	103	255A	2554	2569	2566	2560	2550	256C	A4
D.	đ	Đ	Ď	Ë	ď	Ň	Í	Î	ě	┘	┌	■	▄	Ţ	Ů	▀
	111	110	10E	CB	10F	147	CD	CE	11B	2518	250C	2588	2584	162	16E	2580
E.	Ó	ß	Ô	Ń	ń	ň	Š	š	Ŕ	Ú	ŕ	Ű	ý	Ý	ţ	´
	D3	DF	D4	143	144	148	160	161	154	DA	155	170	FD	DD	163	B4
F.	AD	″	˛	ˇ	˘	§	÷	¸	°	¨	˙	ű	Ř	ř	■	
		2DD	2DB	2C7	2D8	A7	F7	B8	B0	A8	2D9	171	158	159	25A0	A0

Problém diakritiky (CZ/SK)

Kódy :

	Č	č	Š	š	Ž	ž
Kamenických	80	87	98	A8	92	91
Latin II	AC	9F	E6	E7	A6	A7
Windows 1250	C8	E8	8A	9A	8E	9E
Unicode U+	010C	010D	0160	0161	017D	017E

Unicode

Štandard kódovania znakov vyvinutý organizaciou **Unicode Consortium** (1991) (paralelne **ISO 10646** – došlo k zjednoteniu kódovania znakov)

- snaha obsiahnuť takmer všetky znaky používané v jazykoch na svete (japončina, čínština – tisíce znakov)
- prvých 256 znakov je zhodných s rozšíreným ASCII
- pôvodne znaky boli 16-bitové, v súčasnosti 31-bitové

ISO 10646 – definuje **UCS** (Universal Character Set)

Štandard Unicode sa oproti ISO 10646 navyše zaoberá

- algoritmami pre písmo písané zprava doľava (arabština)
- podporou obojstranných textov (ako napr. zmes hebrejštiny a latinky)
- algoritmy pre usporiadavanie a porovnávanie textov

Nevýhodou je dĺžka znakov, prítomnosť kódov v znakoch ako nulový bajt, znak "\", atď.
Vznik systémov kódovania UTF (**Unicode Transformation Format**) ako sú UTF-8, UTF-16 a UTF-32.
Väčšina rozhraní systému Windows používa formu UTF-16.

UTF-8

- kóduje do 1 až 4 bajtov
- do 1 bajtu kóduje prvých 128 znakov zo znakovej sady US-ASCII (U+0000 až U+007F)

Rozsah kodov	Binárna hodnota	UTF-8
000000–00007F	0xxxxxxx	0xxxxxxx
000080–0007FF	00000zzz zxxxxxxx	110zzzzx 10xxxxxx
000800–00FFFF	zzzzzxxx xxxxxxxx	1110zzzz 10zxxxxx 10xxxxxx
010000–10FFFF	000zzzzz xxxxxxxx xxxxxxxx	11110zzz 10zzxxxx 10xxxxxx 10xxxxxx

Reťazce (stringy)

- v jazyku C postupnosť znakov ukončená terminátorom 00H (nulový bajt)
- v Pascale postupnosť, kde prvý bajt obsahuje informáciu o dĺžke textu, po ktorom nasleduje postupnosť kódov max 255 znakov

Obraz

PIXEL – Picture Element

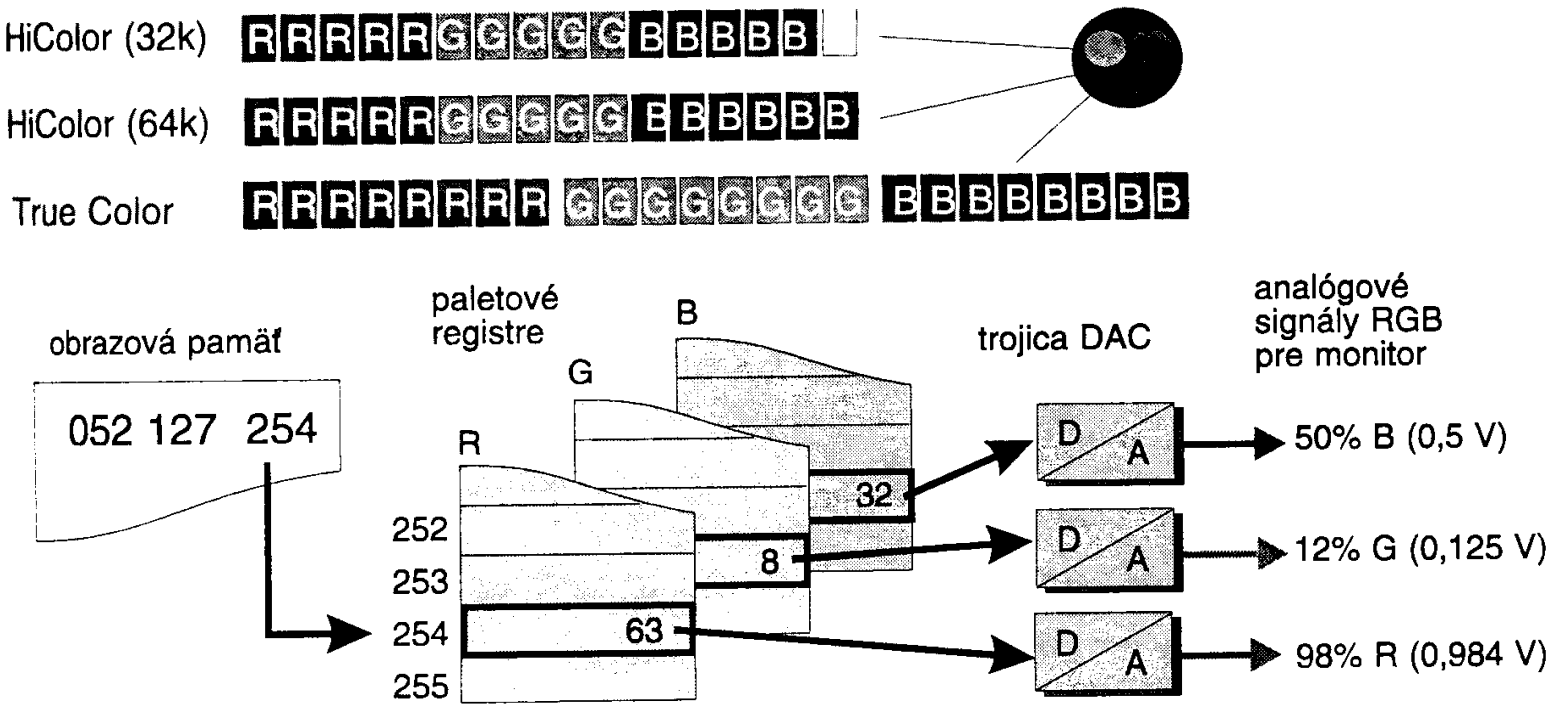
Bodová reprezentácia (rastrová reprezentácia obrazu) – uloženie v matici (bit-map)

- indexov do tabuľky (palety) (monochromatické - 1bit-pixel, farebné napr. 256 farieb – 1bajt/pixel – tabuľka 3*256 bajtov definícií RGB)
- farebných zložiek RGB (Red Green Blue), kódovaných do 2 bajtov

  3*32 úrovní $=2^{15}$ farieb (Hi Color (32k))

  5bR*5bG*6bB úrovní $=2^{16}$ farieb (Hi Color (64k))

  3*256 úrovní = 2^{24} = 16 777 216 farieb (True Color)



Potrebná pamäť:

Monitor: rozlíšenie - 1280*1024 pixelov (True Color)

= 3 932 160 B

Tlačiareň (B/W): papier formátu A4 (210mm x 296 mm) (8.28 inch x 11.65 inch) – pri 600Dpi bude potrebných 8.28*11.65*600*600/8 = 4 340 790 B

Ukladanie údajov do pamäte

Problém ukladania informácií, ak veľkosť údajového typu nie je zhodná s veľkosťou strojového slova.
Ukladanie a čítanie z pamäti v strojových slovách (dané šírkou dátovej zbernice 16b = 2B, 32b = 4B, 64b = 8B)

[1]str.131 Obr.3.5 Ukladanie objektov ....

3 2 1 0

$(1025)_{10} = (00\ 00\ 04\ 01)_H$

ME – 23 01 resp.10 32

	BE	LE
0000	00(MSB)	01
0001	00	04
0002	04	00
0003	01(LSB)	00

Stredný endián – poradie bajtov 2301 (PDP11)

V architektúrach počítačov sa využívajú nasledujúce ukladacie endiány :

- len malý endián (Intel i386)
- len veľký endián (MC 68030)
- nastaviteľný
  - špeciálnym signálom pri Reset (MIPS 2000)
  - nastaviteľný inštrukciou (Intel i860, i486)

Musí byť riešená konverzia rôznych endiánov, ak niektorý podsystém počítača pracuje v inom ukladacom režime.

Dátum:

US : middle - endian

Month, Day, Year (May, 24th,2006 = 5/24/2006)

Europe (~~Hungary~~): little - endian

Day, Month, Year (24th, May, 2006 = 24/ 5/2006)

China, Japan & ISO 8601: big - endian

Year, Month, Day (2006, May, 24th = 2006-05-24)

Zarovnávanie bajtov

Sprístupňovanie (čítanie a zápis objektov) jednoduchých alebo viacbajtových sa môže uskutočňovať v režime:

- **zarovnávaného sprístupňovania** bajtov
  - jednoduchší hardvér, jednoduchšie sprístupňovanie
  - používa sa v architektúrach **RISC**
  - údaje sa ukladajú na adresy, ktoré sú celočíselným násobkom strojového slova (napríklad pre 32-bitové slovo, sa ukladajú na adresy, ktoré sú násobkom čísla 4). V prípade ukladania údajov, ktorých veľkosť nie je celočíselným násobkom strojového slova, nie sú využité niektoré bajty v pamäti.

- **nezarovnávaného sprístupňovania** bajtov
  - zložitejší hardvér (najmä pri stránkovaní pamäte)
  - zložitejšie sprístupňovanie (viacej strojových cyklov pri zápise a čítaní, pomalšie čítanie a zápis do pamäte)
  - používa sa v architektúrach **CISC**
  - pohodlnejšie programovanie
  - údaje sa ukladajú „úsporne“ bezprostredne za sebou (úspora pamäte)

[1]str.132 Obr.3.7 Režimy sprístupňovania bajtov ....

Príklady :

```
char pole[100],c,*cptr;
long a,*lptr;
.
.
.
lptr=(long*)(pole+1);
//alebo lptr=(long*)&pole[1];
a=*lptr;
//-----------------------------------------------
```

v 32-bitovom prostredí kompilátor umiestni začiatok pola **pole** na adresu, ktorá je násobkom 4. Priradením smerníka pole+1, bude určite adresa 4n+1. Vtedy na to, aby priradil do premennnej a hodnotu 4-bajtoveho čísla musí čítanie robiť na dvakrát, t.j. čítanie z adresy **pole** a **pole**+4, a 8-bajtový výraz upraviť na 4-bajtový

Majme

```
struct A{
        char a;
        short b;
};
```

Neplatí obecne, že
sizeof(A) = sizeof(a) +sizeof(b)
teda skôr
sizeof(A)!= sizeof(a) +sizeof(b)

Kompilátory doplnia štruktúru práznym bajtom (bajtami). Dá sa nastaviť typ zarovnávania, napr. pre kompilátory Borland

#pragma option –ax // kde x je počet bajtov zarovnávnia resp.
#pragma option push –ax // nastavenie zarovnania
.
#pragma option pop // vrátenie pôvodneho zarovnania

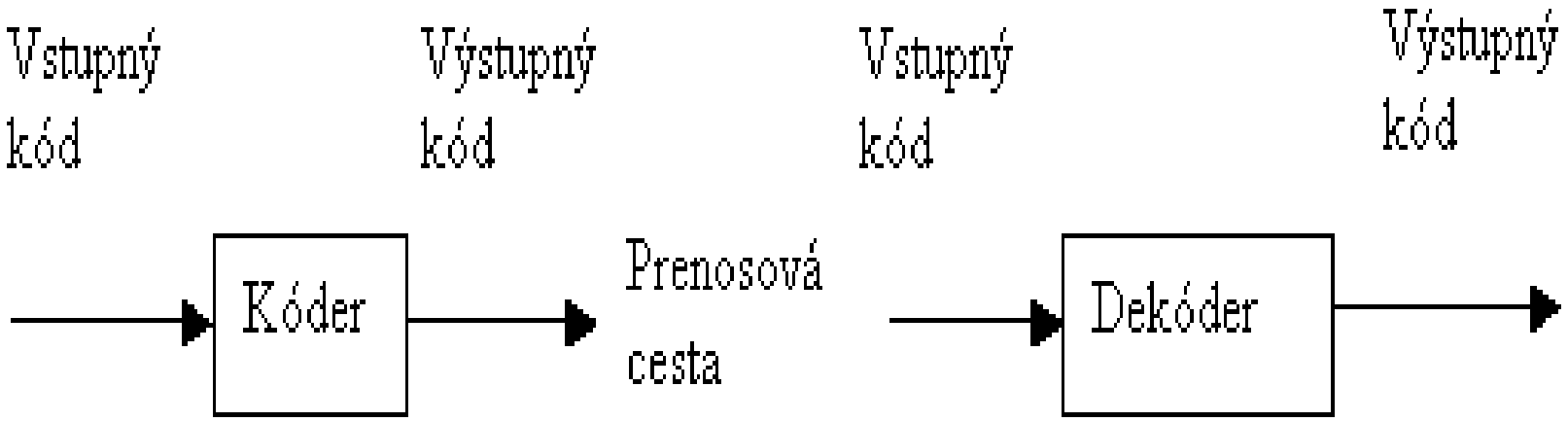
Teda obecne neplatí, že v poli štruktúr bude za premennou b, nasledovať premmená a ďalšej štruktúry, lebo začiatky štruktúr sú od seba sizeof(A).

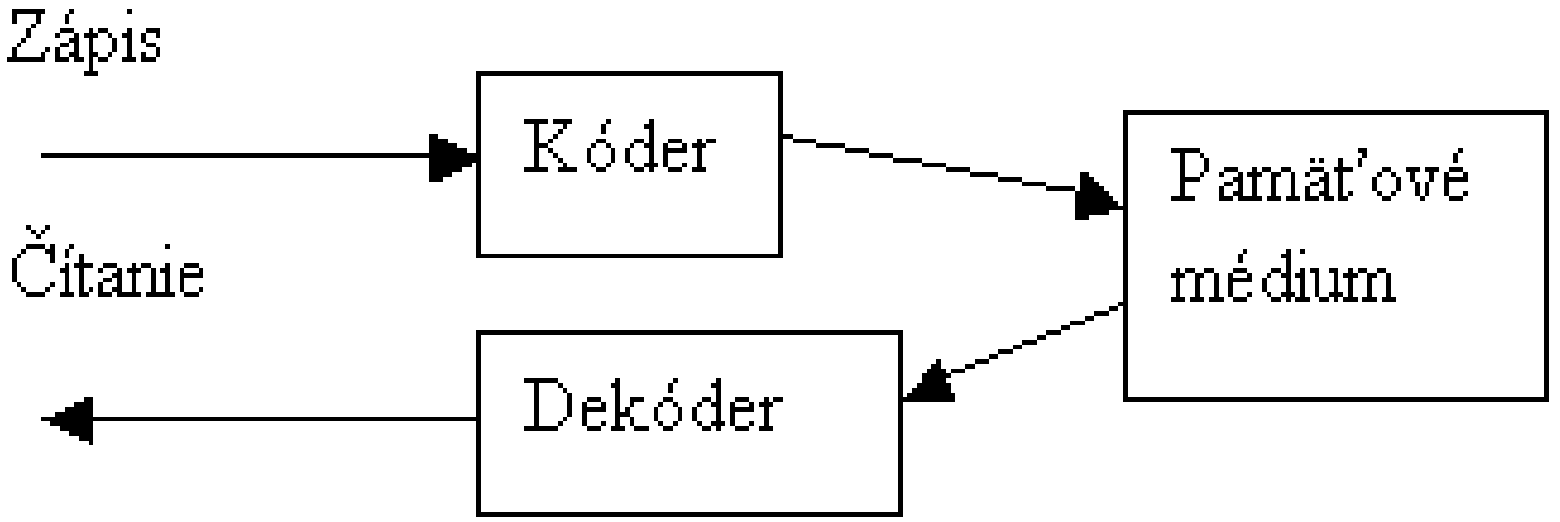
Kódovanie údajov

Kódovanie údajov (binárnych slov) =
mapovanie množiny vstupných slov (kódov) do množiny výstupných slov (kódov)

Cieľ

- zabrániť vzniku chyby pri prenose a uchovávaní informácie
- komprimácia dĺžky informácie (zrýchlenie prenosu informácie, lepšie využitie pamäťových médií)
- prispôsobenie sa konvenciám pri prenose dát (protokol)
- ochrana údajov (šifrovanie),
- počítač – binárne čísla, operátor – dekadické čisla, atď.


Vstupný kód
Výstupný kód
Vstupný kód
Výstupný kód
Kóder
Prenosová cesta
Dekóder


Zápis
Kóder
Pamäťové médium
Čítanie
Dekóder

BCD kod, (kód s váhami 8 4 2 1)

0000 – 0
0001 – 1
0010 – 2
0011 – 3
0100 – 4
0101 – 5
0110 – 6
0111 – 7
1000 – 8
1001 – 9
1010 – x
1011 – x
1100 – x
1101 – x
1111 – x

Prevod 1357 do BCD kodu

1	3	5	7
0001	0011	0101	0111

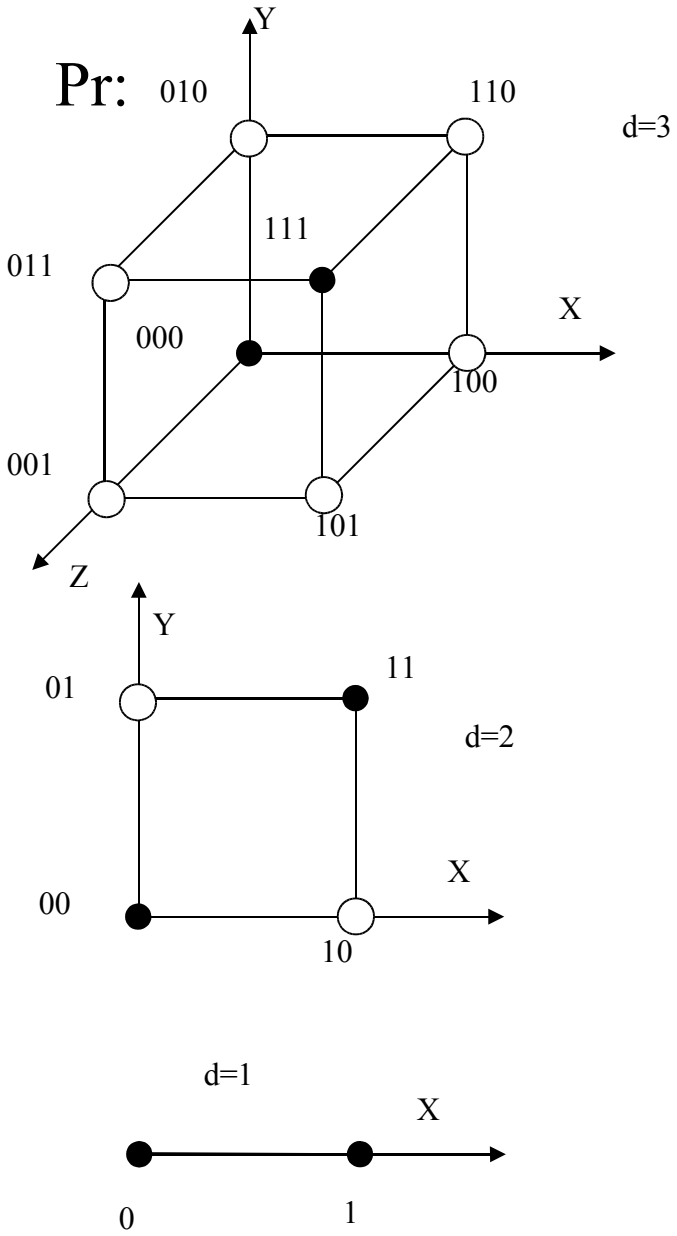
Kódovanie na zabránenie vzniku chyby pri prenose a uchovávaní informácie

Hammingova vzdialenosť medzi dvomi binárnymi slovami (kódmi) sa rovná počtu bitov, v ktorých sa nezhodujú dané slová

Napríklad medzi štvorbitovými slovami

$(0111)_2=(7)_{10}$ $(1000)_2=(8)_{10}$

je Hammingova vzdialenosť 4.



n **informačných bitov**

(kód pôvodnej informácie)

\- k **kontrolných bitov**

(redundantné bity)

d – kodová vzdialenosť

α **- počet detekov. chýb**

β **- počet opraven. chýb**

$$d \geq \alpha + 1$$

$$d \geq 2\beta + 1$$

Grayov kód

Problém správneho dekódovania polohy kódového kotúča

[2]str.161

Problém „presnej“ realizácie prechodov medzi priehľadnými a nepriehľadnými sektormi

Riešením je kódovanie sektorov tak, aby pri zmene sektora sa menil kód len v jednom bite $\Rightarrow$ Hammingova vzdialenosť medzi kódmi susedných sektorov bola 1.

[2]str.162

Dĺžka kódového slova nezmenila

Dekóder:

Kóder:

[2]str.162

Detekčné kódy (EDC)

sú schopné detekovať definovaný počet chýb
Skladajú sa z :

- **n informačných bitov** (kód pôvodnej informácie) a
- **k kontrolných bitov (** nenesú žiadnu novú informáciu, redundantné bity)

Najväčší dôraz na odhalenie jednej chyby (jej pravdepodobnosť je najväčšia z možných chýb).

Podľa spôsobu vytvorenia kódu v štruktúre kódu ich delíme na :

- systematické a
- nesystematické

Pre **nesystematické kódy** nie je definovaná pozícia informačných a zabezpečovacích bitov, počet rádov je konštantný (kódy 2 z 5, 3 z 5, 3 zo 7 ....).
Typické pre prenos informácií sú **systematické kódy** m rádov zahŕňa, je definovaných k informačných a (m-k) zabezpečovacích, označujeme ich K(m,k)

Princíp je založený na tom, že z $\mathbf{2^{n+k}}$ možných prijatých slov je len $\mathbf{2^{n}}$ správnych (neobsahujúcich definovaný počet chýb)

Paritný kód (detekčný)

- používa sa jeden kontrolný bit (paritný bit)
- parita môže byť – párna (even) alebo nepárna (odd)

Stanovenie parity :

$$\sum_{i=0}^{m-1} a_i \,(\mathrm{mod}\, 2) = a_{m-1} \oplus a_{m-2} \oplus \ldots \oplus a_1 \oplus a_0 = C$$

$$kde \quad (a+b)(\mathrm{mod}\, 2) = a \oplus b$$

Kde:

C=1 - kódové zabezpečenie **nepárnou paritou**,

C=0 - kódové zabezpečenie **párnou paritou**.

Príklad : Prenášame informáciu zakódovanú v 7 informačných bitoch

1001110

potom

Pre 11001110 *bude*

$$1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0 \oplus 0 \oplus 0 \oplus 1 = 0 \oplus 1 = 1$$

pre 01001110 *bude*

$$0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1 \oplus 0 \oplus 0 \oplus 1 = 1 \oplus 1 = 0$$

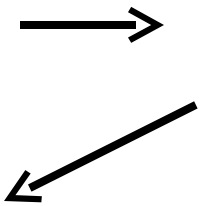
pri párnej parite vysielame **01001110** (pri nepárnej **11001110**).

Ak nastane 1 chyba (nepárny počet chýb) pri prenášaní ľubovolného bitu, paritný bit prijatého slova nebude správny).

Blokový detekčný kód (maticový detekčný kód)

- pri **prenose bloku** viacerých informačných slov,
- použitie **priečnej** a **pozdĺžnej parity**

Bit	W_1	W_2	W_3	W_4	W_5	W_6
D_0	0	1	1	0	1	0
D_1	1	0	0	1	0	1
D_2	1	1	0	1	1	0



Bit	W_1	W_2	W_3	W_4	W_5	W_6	W_7
D_0	0	1	1	0	1	0	1
D_1	1	0	0	1	0	1	1
D_2	1	1	0	1	1	0	0
D_3	0	0	1	0	0	1	0

Bit	W_1	W_2	W_3	W_4	W_5	W_6	W_7	
D_0	0	1	1	0	1	0	1	$\vee$
D_1	1	0	1	1	0	1	1	$\times$
D_2	1	1	0	1	1	0	0	$\vee$
D_3	0	0	1	0	0	1	0	$\vee$
	$\vee$	$\vee$	$\times$	$\vee$	$\vee$	$\vee$	$\vee$	

- detekuje viacero chýb
- opraviť je možné len jednu chybu

Samoopravné kódy (ECC)

Hammingove kódy

Patria medzi najjednoduchšiu skupinu **detekčných** a zároveň **samoopravných kódov**.

Generujú kód zložený z **m informačných bitov** vstupného slova a **k kontrolných bitov**, dĺžka výstupného kódu bude **n = m + k**. Kód sa označuje **H (n,m)**

Pre **H(7,4)**

Poloha bitu	7	6	5	4	3	2	1
Kod bitu	I_4	I_3	I_2	C_3	I_1	C_2	C_1

I_i - hodnota zdrojového bitu

C_i - hodnota kontrolného bitu

Kontrolné bity sa generujú podľa vzťahov

$$C_3 = I_2 \oplus I_3 \oplus I_4$$
$$C_2 = I_1 \oplus I_3 \oplus I_4$$
$$C_1 = I_1 \oplus I_2 \oplus I_4$$

Nech $I_4 I_3 I_2 I_1 = 1101$

potom $C_3 = 0 \oplus 1 \oplus 1 = 0$

$C_2 = 1 \oplus 1 \oplus 1 = 1$

$C_1 = 1 \oplus 0 \oplus 1 = 0$

potom $I_4 I_3 I_2 C_3 I_1 C_2 C_1 = 1100110$

Nech prijatý kód je 1000110 *(1100110)*

potom

$$C_3 = I_2 \oplus I_3 \oplus I_4 = 0 \oplus 0 \oplus 1 = 1$$
$$C_2 = I_1 \oplus I_3 \oplus I_4 = 1 \oplus 0 \oplus 1 = 0$$
$$C_1 = I_1 \oplus I_2 \oplus I_4 = 1 \oplus 0 \oplus 1 = 0$$

Nová trojica kontrolných bitov je **100** namiesto **010**.
Teraz po dvojiciach určíme súčet modulo 2 pre kontrolné bity

010
100 $\quad 0 \oplus 1 \quad 1 \oplus 0 \quad 0 \oplus 0 = 110$

Výsledok operácie nad kontrolnými bitmi dáva binárnu kombináciu, ktorej dekadická hodnota určuje polohu bitu, v ktorom treba opraviť hodnotu bitu. V tomto prípade je to hodnota $\mathbf{110_2 = 6}$, t.j. je to bit ktorý obsahuje informáciu o $\mathbf{I_6}$.

Teraz predpokladajme, že pri prenose bude poškodený kontrolný bit, napr. $\mathbf{C_2}$

1100100

Nová trojica kontrolných bitov **010** namiesto **000**. Určíme polohu poškodeného bitu

$$\begin{array}{llll} 010 & & & \\ 000 & 0 \oplus 0 & 1 \oplus 0 & 0 \oplus 0 = 010 \end{array}$$

V tomto prípade je to hodnota $\mathbf{010_2} = \mathbf{2}$, t.j. je to bit, ktorý obsahuje informáciu o $\mathbf{C_2}$.

Hammingov kód je schopný opraviť chybu v prijatom bajte aj pre hodnoty kontrolných bitov.

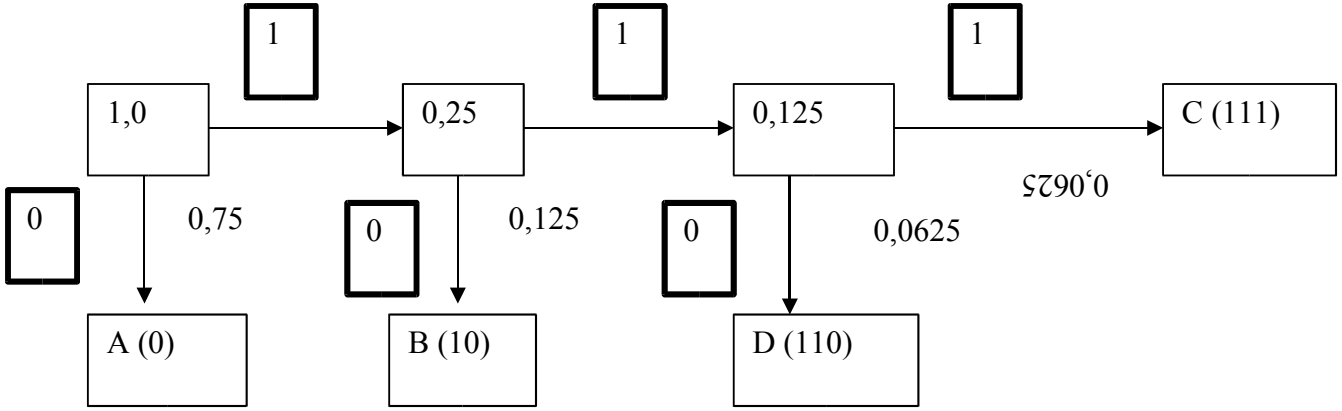
Komprimácia dĺžky informácie
Huffmanov kód

- komprimačný kód
- premenlivá dĺžka výstupného kódu
- vychádza zo štatistiky (z pravdepodobností výskytu kódovaného slova, napr. písmen abecedy v textoch určitého typu), štatistickými metódami sa určuje kódovacia tabuľka (strom)
- tabuľky sú pevné a známe na strane kódera aj dekódera
- nevýhodou je sériové generovanie aj spracovanie kódu po bitoch
- používa sa v kódovaní faxových správ, v kódovaní rastrových obrazov (TIFF)

Príklad : Majme danú množinu 4 symbolov (znakov) s pravdepodobnosťami ich výskytu v prenášaných správach

Znak	Pravdepodobnosť (početnosť) výskytu	Kód
A	0,75	00
B	0,125	01
C	0,0625	10
D	0,0625	11

Vytvoríme kódovací strom, tak aby v koncových uzloch kratších vetiev končili kódy s väčšou pravdepodobnosťou :



Potom kódy jednotlivých znakov budú :

Znak	Pravdepodobnosť (početnosť) výskytu	Kód
A	0,75	0
B	0,125	10
C	0,0625	111
D	0,0625	110

Stredná dĺžka kódu bude :

$$1.\frac{3}{4}+2.\frac{1}{8}+3.\frac{1}{16}+3.\frac{1}{16}=1.375$$

RLE (Run Length Encoding)

Viacero modifikácií princíp kódovania viacero (3-255) rovnakých bajtov do dvojice počet, obsah bajtu (nevýhoda, môže byť komprimovaný záznam dlhší ako pôvodný záznam

Princíp RLE-8

- séria rovnakých bajtov - 1B počet rovnakých bajtov 1B hodnota bajtu
- ak je viacero nerovnakých bajtov, potom generuje 00 (kód úniku) 1B počet bajtov + n-tica nekomprimovaných bajtov

Napr.
Sekvencia

11 11 11 11 11 11 11 22 22 22 22 34 67 88 33 33 33 (17)

07 11 04 22 00 03 34 67 88 03 33 (11)

Kompresia dát

- bezstratové (Huffman, LZW, RLE....)
- stratové (prenos obazu JPEG)

Kódovanie binárnych dát (súborov) do textových

Použitie na prenos dát v e-mailoch (protokol MIME), používa sa kódovanie Base64

Princíp :

- 8-bitové údaje sa prevedú na 6-bitové (3 byty = 4 šestice bitov)
- kóduje sa podľa tabuľky, kde výstupným kódom sú ASCII kódy len písmen anglickej abecedy, číslice a znaky „- _ a =“.

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	- (minus)
12	M	29	d	46	u	63	_ (understrike)
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Input data: 0x14fb9c03d97e
Hex: 1 4 f b 9 c | 0 3 d 9 7 e
8-bit: 00010100 11111011 10011100 | 00000011 11011001
11111110
6-bit: 000101 001111 101110 011100 | 000000 111101 100111
111110
Decimal: 5 15 46 28 0 61 37 62
Output: F P u c A 9 l +

Input data: 0x14fb9c03d9
Hex: 1 4 f b 9 c | 0 3 d 9
8-bit: 00010100 11111011 10011100 | 00000011 11011001
pad with 00
6-bit: 000101 001111 101110 011100 | 000000 111101 100100
Decimal: 5 15 46 28 0 61 36
pad with =
Output: F P u c A 9 k =

Input data: 0x14fb9c03
Hex: 1 4 f b 9 c | 0 3
8-bit: 00010100 11111011 10011100 | 00000011
pad with 0000
6-bit: 000101 001111 101110 011100 | 000000 110000
Decimal: 5 15 46 28 0 48
pad with = =
Output: F P u c A w = =

Literatúra:

- [1] Jelšina, M.: Architektúry počítačových systémov, Princípy, ... ELFA 2002
- [2] Clements,A: The Principles of Computer Hardware, Oxford