Úvod do informačních technologií

Jan Outrata

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

přednášky

Reprezentace dat

Kódování dat

– data v počítači: celá čísla, čísla s řádovou čárkou (necelá), znaky různých abeced (pro písmena, cifry, symboly atd.) – **alfanumerické znaky**, speciální a řídící znaky
– binární reprezentace = **kódování dat do posloupnosti binárních hodnot**

- kód (kódování) = zobrazení čísel a znaků na binární hodnoty, pomocí **kódových schémat a tabulek**
- kód (kódové slovo) = binární hodnota, obecně posloupnost kódových znaků
- dekódování = převod kódového slova na původní číslo nebo znak
- různé kódy pro uložení dat, zpracování dat, zabezpečení (uložení, přenosu) dat proti chybám atd.
- kódující a dekódující log. obvody s pamětí = kodéry, dekodéry

Endianita (Endianness)

= pořadí bytů (byte order, nebo jiných atomických hodnot) v uložení binárních hodnot delších než 1 byte, např. v operační paměti (adresované od nižších adres k vyšším)

Kódování dat

Endianita (Endianness)

- **little-endian** = od nejméně významného bytu (LSB) hodnoty k nejvýznamnějšímu
  - např. pro $(30201)_{16}$ pořadí $01\ 02\ 03\ 00$ při uložení do 4 bytů
  - pro hodnoty nevyužívající všechny byty, do kterých je hodnota uložena, možno ze stejné adresy načíst jen využívané byty
  - platformy např. Intel x86, AMD x86-64, DEC Alpha, ARM (od verze 3 bi-endian = může pracovat buď little nebo big-endian), Ethernet, USB (pořadí bitů)
- **big-endian/network** = od nejvýznamnějšího bytu (MSB) hodnoty k nejméně významému
  - např. pro $(30201)_{16}$ pořadí $00\ 03\ 02\ 01$ při uložení do 4 bytů
  - pro přesnou hodnotu nutno načíst vždy všechny byty, do kterých je hodnota uložena, při čtení ne všech bytů ze stejné adresy ale získána přibližná hodnota (bez nižších řádů)
  - platformy např. Motorola 6800 a 68k, IBM POWER, SPARC (do verze 9, pak bi-endian), internetové protokoly
- **middle/mixed-endian** = kombinace little- a big-endian
  - např. pro $(30201)_{16}$ pořadí $03\ 00\ 01\ 02$ nebo $02\ 01\ 00\ 03$ při uložení do 4 bytů
  - platformy např. ARM (pro čísla s plovoucí řádovou čárkou ve formátu double, viz dále)
- řeší překladač/interpret prog. jazyka pro danou platformu, mezi platformami nutné konverze (např. v síťovém API)

Celá čísla

= **interval** $\langle$**min. nekladné, max. nezáporné**$\rangle$ – hranice závisí na (konečném) počtu n bitů pro reprezentaci a použitém kódu

Nezáporná čísla:

Vážený poziční kód

= zápis čísla ve dvojkové poziční číselné soustavě

- např. $123 = (123)_{10} = [\mathbf{IIII0II}]_2$
- $\langle 0, 2^n - 1 \rangle$

Dvojkově desítkový kód (BCD, Binary Coded Decimal)

= zápis desítkových číslic čísla (zapsaného v desítkové soustavě) ve dvojkové soustavě s pevným počtem 4 dvojkových číslic pro každou desítkovou číslici

- např. $123 = [\mathbf{000I00I000II}]_{BCD}$
- $\langle 0, 10^{n/4} - 1 \rangle$, pro $n = 4^k$
- neefektivní, složitější log. obvody, snadno dekódovatelný pro člověka, použití pro zobrazení čísel

Celá čísla

Nezáporná i záporná čísla:

Přímý kód

= znaménkový bit ($\mathbf{0}$ pro nezáporná, $\mathbf{I}$ pro záporná čísla) + (vážený poziční) kód pro absolutní hodnotu čísla – tzv. sign-magnitude

- např. $-123 = [\mathbf{IIIII0II}]_{S2}$
- $\langle -2^{n-1} - 1, 2^{n-1} - 1 \rangle$
- neefektivní (nevyužitý 1 kód), nevhodný pro aritmetiku (testování znaménka, různé postupy sčítání a odčítání)

Aditivní kód

= vážený poziční kód pro (nezáporné) číslo rovno součtu kódovaného čísla a **zvolené konstanty**

- konstanta obvykle $\mathbf{2^{n-1}}$
- např. $123 = [\mathbf{IIIII0II}]_{A(128)}$, $-123 = [\mathbf{I0I}]_{A(128)}$
- $\langle -2^{n-1}, 2^{n-1} - 1 \rangle$
- jinak reprezentovaná nezáporná čísla, složitější násobení, použití pro reprezentaci exponentu u reprezentace čísel s řádovou čárkou

Celá čísla

Inverzní (jedničkový doplňkový) kód

= pro nezáporná čísla vážený poziční kód, pro záporná log. negace všech bitů váženého pozičního kódu absolutní hodnoty, 1. bit má význam znaménka

- např. $-123 = [\mathbf{I} \ldots \mathbf{0000I00}]_I$
- $\langle -2^{n-1} - 1, 2^{n-1} - 1 \rangle$
- neefektivní (nevyužitý 1 kód), nevhodný pro aritmetiku (různé postupy sčítání a odčítání)

(Dvojkový) doplňkový kód

= pro nezáporná čísla vážený poziční kód, pro záporná log. negace všech bitů váženého pozičního kódu absolutní hodnoty **zmenšené o 1** (ekv. log. negace všech bitů váženého pozičního kódu absolutní hodnoty s **binárním přičtením I**), 1. bit má význam znaménka

- např. $-123 = [\mathbf{I} \ldots \mathbf{0000I0I}]_{2'}$
- $\langle -2^{n-1}, 2^{n-1} - 1 \rangle$
- efektivní, vhodný pro aritmetiku (odčítání pomocí sčítání se záporným číslem)

ÚKOL

Vytvořte binární reprezentace několika celých čísel pomocí aditivního, inverzního (jedničkově doplňkového) a (dvojkově) doplňkového kódu.

Čísla s řádovou čárkou

= **podmnožina racionálních čísel** – přesnost omezena na počet platných číslic, z důvodu konečné bitové reprezentace

Fixní řádová čárka

= pevně zvolený max. počet $\mathbf{n}$ **platných číslic pro necelou část čísla** (část za čárkou)

- číslo x v číselné soustavě o základu B reprezentováno jako zlomek $\frac{x \cdot B^n}{B^n}$
- uložena pouze celočíselná část $\mathbf{x \cdot B^n} \Rightarrow$ přibližná reprezentace
- přesnost (rozlišení čísel) B^{-n}, "přesnost na n platných číslic za čárkou"
- $\Rightarrow$ **celočíselná aritmetika** (se zachováním přesnosti)

Čísla s řádovou čárkou

Fixní řádová čárka

Reprezentace necelé části čísla:

- necelá část F čísla jako součet (případně nekonečné) mocninné řady o základu B:

$$F = a_{-1} \cdot B^{-1} + a_{-2} \cdot B^{-2} + \cdots$$

$$\begin{aligned} (0,625)_{10} &= 6 \cdot 10^{-1} + 2 \cdot 10^{-2} + 5 \cdot 10^{-3} = \\ (0,101)_2 &= 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} \end{aligned}$$

- tatáž necelá část čísla může být v poziční soustavě o jednom základu vyjádřena konečnou řadou, zatímco v soustavě o jiném základu nekonečnou řadou, např.

$$\begin{aligned} (0,4)_{10} &= 4 \cdot 10^{-1} = \\ (0,0110011\ldots)_2 &= 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} + 0 \cdot 2^{-5} + 1 \cdot 2^{-6} + 1 \cdot 2^{-7} + \cdots \end{aligned}$$

- získání zápisu necelé části čísla v dané číselné soustavě a naopak: podobné postupy jako pro celá čísla, jen místo dělení je násobení a naopak

Čísla s řádovou čárkou

Fixní řádová čárka

Získání (případně nekonečného) zápisu $(S_{-1}S_{-2}\ldots)_B$ necelé části F čísla (dané hodnoty) postupným násobením:

$a_{-1} = 0$
$i = -1$
while $F > 0$ **do**
  $F = F * B$
  $a_i = F \bmod B$
  $F = F - a_i$
  $i = i - 1$

Čísla s řádovou čárkou

Fixní řádová čárka

Získání (případně přibližné hodnoty) necelé části F čísla z jejího (konečného) zápisu $(S_{-1}S_{-2}\ldots S_{-n+1}S_{-n})_B$
postupným dělením:

$F = a_{-n}$
for $i = -n+1$ **to** -1 **do**
  $F = F//B + a_i$
$F = F//B$

- $//$ označuje dělení s řádovou čárkou
- převod zápisu necelé části čísla v soustavě o základu B^k na zápis v soustavě o základu B (a naopak) stejný jako u celých čísel

Čísla s řádovou čárkou

Fixní řádová čárka

Binární reprezentace:

= BCD nebo **doplňkový kód celočíselné části čísla vynásobeného $\mathbf{B^n}$** (ekv. doplňkový kód zřetězení vážených pozičních kódů celé a necelé části čísla)

- např. pro doplňkový kód $-5,25 = [\mathbf{I}\ldots\mathbf{0I0II}]_{2'}$ (přesnost na 2 platné číslice za čárkou)
- interval čísel, hranice závisí na počtu $t = m + n$ bitů pro reprezentaci a použitém kódu pro celou a necelou část čísla
- např. pro doplňkový kód: $\langle -2^{m-1}, 2^{m-1} - 2^{-n} \rangle$
- různé formáty binární reprezentace, např. $\mathbf{Q}m.n$ (Texas Instruments), $\mathbf{fx}m.t$
- použití u zařízení bez jednotky pro výpočty s plovoucí řádovou čárkou, při vyžadování konstantní přesnosti nebo kvůli rychlejší celočíselné aritmetice

Čísla s řádovou čárkou

Plovoucí řádová čárka

= **pohyblivá pozice čárky mezi platnými číslicemi celé a necelé části čísla** $\sim$ počítačová realizace vědecké notace čísla

- číslo x reprezentováno v **semilogaritmickém tvaru** o základu b: $\mathbf{x = s \cdot b^e}$
  - (pro $x \neq 0$) $-b < s < 0$ nebo $0 < s < b$, tj. s, e takové, že před čárkou je pouze první nenulová číslice s
  - používaný desítkový ($b = 10$) a dvojkový ($b = 2$) základ
  - např. $123,456 = 1,23456 \cdot 10^2 = 1,929 \cdot 2^6$, $-0,123 = -1,23 \cdot 10^{-1} = -1.968 \cdot 2^{-4}$
- uloženy **znaménko** do 1 bitu, **exponent** e (včetně znaménka) do m bitů a **normovaný tvar** s absolutní hodnoty čísla do n bitů (significand, "mantissa")
  - exponent v aditivním kódu (s konstantou rovnou $2^{m-1} - 1$) – udává rozsah reprezentace, $\langle -b^{b^k}, b^{b^k+1} \rangle$, kde $b^k = 2^{m-1} - 1$
  - normovaný tvar absolutní hodnoty čísla v kódu pro fixní řádovou čárku (u základu 2 se číslice 1 před čárkou neukládá) – udává přesnost reprezentace b^{-n}
- $\Rightarrow$ přibližná reprezentace

Čísla s řádovou čárkou

Plovoucí řádová čárka

Různé formáty s různou přesností (standard **IEEE 754**):

- základ $b = 2$ i $b = 10$, vážený poziční kód pro normovaný tvar
- **single** (**float**, 32 bitů) – 8 bitů pro exponent, 23 bitů pro normovaný tvar, rozsah $\sim \langle -10^{38}, 10^{38} \rangle$, asi 7 platných desítkových číslic

$$\begin{aligned} 123.456 &= [\mathbf{0I0000I0IIII0II0III0I00I0IIII00I}]_2 \\ -0.123 &= [\mathbf{I0IIII0IIIIII0IIIII00III0II0II0I}]_2 \end{aligned}$$

- **double** (64 bitů) – 11 bitů pro exponent, 52 bitů pro normovaný tvar, rozsah $\sim \langle -10^{308}, 10^{308} \rangle$, asi 16 platných desítkových číslic
- další: half (16 bitů, 5 pro exponent), extended (long double, 80 bitů, 15 pro exponent), quad (128 bitů, 15 pro exponent)
- **speciální "čísla"**: $-\infty, +\infty$ (exponent samé $\mathbf{I}$, normovaný tvar nulový), NaN (Not a Number, exponent samé $\mathbf{I}$), $-0 \neq 0$ (exponent i normovaný tvar nulové)

Čísla s řádovou čárkou

Plovoucí řádová čárka

- **aritmetika s plovoucí řádovou čárkou**
  - použité zaokrouhlovací algoritmy a výjimky (pro nedefinované operace)
  - měřítko výkonnosti počítačů (ve vědeckých výpočtech), jednotka **FLOPS** (FLoating point Operations Per Second)
- mnohem širší množina čísel než u fixní řádové čárky na úkor nižší přesnosti

ÚKOL

Vytvořte binární reprezentace několika racionálních čísel s fixní i plovoucí řádovou čárkou.

Text

= posloupnost tisknutelných znaků = znaků různých abeced (pro písmena, cifry, symboly atd.) – **alfanumerické znaky**

\+ speciální a (netisknutelné) řídící znaky – jen některé se zahrnují do **plain text**u

ASCII (American Standard Code for Information Interchange, 1963)

- standarní kódová tabulka pro kódování znaků **anglické abecedy, cifer**, symbolů (matematických aj.), speciálních (mezera, interpunkce, atd.) a **řídících znaků** (původně pro ovládání dálnopisu, odřádkování, návrat vozíku, tabulátory, backspace aj.)
- každý znak kódován původně do **7 bitů** = 128 znaků
- přidán nejvyšší 8. bit, tj. tabulka rozšířena o dalších 128 znaků: některé **znaky národních abeced**, další speciální znaky (**grafické**, jednotky aj.)

Obrázek: ASCII tabulka

Text

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Text

ASCII

- několik rozšíření pro různé národní abecedy – různé kódové tabulky rozšířené ASCII, např. **ISO 8859-1**, CP437 (IBM PC, OS MS DOS)
- pro znaky české abecedy (východoevropské/středoevropské jazyky):
  - **ISO 8859-2 (ISO Latin 2)**: standard ISO, používaný v UNIXových operačních systémech (OS)
  - **Windows 1250 (CP1250)**: kód firmy Microsoft, používaný v OS MS Windows, od ISO 8859-2 se liší např. ve znacích š, ť, ž
  - **Mac CE**: kód firmy Apple, používaný v Apple MAC OS
  - CP852 (PC Latin 2): kód firmy IBM, používaný v OS MS DOS
  - další (česko-slovenské): kód Kamenických (další používané v OS MS DOS), KOI8-ČS (kód v rámci RVHP) a další
- **ASCII art** – výtvarné umění kresby obrázků pomocí znaků ASCII v neproporcionálním fontu, např. emotikony (“smajlíky”), použití u textových negrafických systémů

Obrázek: ASCII art

Text

Text

EBCDIC (1964)

- kódování firmy IBM podle kódu pro děrné štítky
- základní osmibitový, rozšířený 16-bitový, různé pro různé národní abecedy
- **nespojitý pro znaky latinky**, dnes nepoužívaný

Text

Unicode (1987–1991)

- rozšíření ASCII nestačí a jsou ad-hoc (např. problematické pro východoasijské, arabské, hebrejské aj. znaky)

= původně 16-bitová tabulka znaků **UCS-2 (Universal Character Set)**

- později oddělení množiny znaků a kódů pro ně (do tzv. kódových bodů a do binární reprezentace)

= standard **ISO 10646** (definice UCS-4, 31-bitová) + algoritmy pro texty zprava doleva a oboustranné texty

- UCS = **otevřená množina pojmenovaných znaků všech abeced** a kombinovaných znaků (např. diakritických), v současnosti (2012) více než 110 000 znaků (poslední verze 6.2 z roku 2012), znaky jen přidávány, prostor pro více než milion znaků
- znakové sady = kódování podmnožiny znaků do kódových bodů (nezáporných celých čísel, **U+hexčíslo**), např. původní ASCII a její rozšíření, BMP (Basic Multilingual Plane) = prvních 65534 znaků UCS

Unicode

Způsob kódovaní (UTF, UCS Tranformation Format)

= kódování kódových bodů do binární reprezentace

- pro jednoznačné kódování celé tabulky Unicode by bylo potřeba 21 bitů (hodnoty 0_{16} až $10FFF_{16}$)
- **UTF-8**: do posloupnosti 1 až 6 bytů, kompatibilní s ASCII (7bitové, přímo) a ISO 8859-1 (prvních 128 dvoubajtových), nezávislý na "endian"itě systémů, všeobecně používané (zejména v UNIXových OS a na Internetu a WWW), RFC 3629
  - znaky $U + 0$ až $U + 7F$ do 1 bytu 0_{16} až $7F_{16}$ (přímo)
  - další jako posloupnosti bytů, kde každý má nejvyšší bit roven $\mathbf{1}$, 1. byte $C0_{16}$ až FD_{16} určuje, kolik bytů posloupnost má (počtem nejvyšších jedničkových bitů následovaných $\mathbf{0}$), 5 bitů pro kód znaku, další byty 80_{16} až BF_{16}, 6 bitů pro kód znaku, big-endian
  - BMP jen 1 až 3 byty, české 1 nebo 2 byty (diakritické)
  - byty FE_{16}, FF_{16} nepoužity

Unicode

Způsob kódovaní (UTF, UCS Tranformation Format)

Tabulka: Kódování UTF-8

$U+00000000 - U+0000007F$	$0xxxxxxx$
$U+00000080 - U+000007FF$	$110xxxxx\ 10xxxxxx$
$U+00000800 - U+0000FFFF$	$1110xxxx\ 10xxxxxx\ 10xxxxxx$
$U+00010000 - U+001FFFFF$	$11110xxx\ 10xxxxxx\ 10xxxxxx\ 10xxxxxx$
$U+00200000 - U+03FFFFFF$	$111110xx\ 10xxxxxx\ 10xxxxxx\ 10xxxxxx\ 10xxxxxx$
$U+04000000 - U+7FFFFFFF$	$1111110x\ 10xxxxxx\ 10xxxxxx\ 10xxxxxx\ 10xxxxxx\ 10xxxxxx$

- např. "Příliš" $= [50C599C3AD6C69C5A1]_{16}$ ("ř" $= U+159$, "í" $= U+ED$, "š" $= U+161$)

Unicode

Způsob kódovaní (UTF, UCS Tranformation Format)

- **UTF-16**: do posloupnosti 1 až 2 slov (2 byte), používané zejména v OS MS Windows a prog. jazyku Java, dříve UCS-2 (pevně 16 bitů)
  - znaky $U+0$ až $U+FFFF$ do 2 bytů přímo
  - další znaky do 4 bytů, 1. $D8_{16}$ až DB_{16}, 3. DC_{16} až DF_{16}, 2 bity pro kód znaku

Tabulka: Kódování UTF-16

$U+000000$ -- $U+00FFFF$	$xxxxxxxx\ xxxxxxxx$
$U+010000$ -- $U+10FFFF$	$110110xx\ xxxxxxxx\ 110111xx\ xxxxxxxx$

  - např. "Příliš" $= [0050015900ED006C00690161]_{16}$
  - **BOM (Byte-Order Mark, UTF signatura)** $=$ znak $\mathbf{U+FEFF}$ ("nedělitelná mezera nulové šířky") – k rozlišení pořadí ukládání bytů (little/big-endian) v UTF-16 a odlišení UTF-16 od UTF-8, v UTF-16 byty $FE_{16}FF_{16}$ pro big-endian a $FF_{16}FE_{16}$ pro little-endian, v UTF-8 tyto byty neplatné, kód znaku jsou byty $EF_{16}BB_{16}BF_{16}$ (ve standardu explicitně povolené, ale nedoporučované, ale OS MS Windows používají k označení UTF-8)

Unicode

Způsob kódovaní (UTF, UCS Tranformation Format)

- další: UTF-32/UCS-4 (pevně do 4 byte, příliš nepoužívané), **UTF-7** (do posloupnosti 7-bitových ASCII znaků, pro e-mail), aj.

Text

Kód pro nový řádek

- různý v různých operačních systémech
- **LF (Line Feed, odřádkování, $\mathbf{A_{16}}$)**: v UNIXových OS
- **CR (Carriage Return, návrat vozíku, $\mathbf{D_{16}}$) + LF**: v OS MS DOS a Windows
- **CR**: v OS od firmy Apple

Escape sekvence

= posloupnosti znaku **ESC (Escape, $\mathbf{1B_{16}}$)** následovaného jedním nebo více znaky z ASCII

- rozšíření ASCII se speciálním významem sekvencí – pozice kurzoru, barva nebo font textu na obrazovce znakového terminálu, přepnutí módu zařízení aj.

ÚKOL

Vytvořte binární reprezentace několika českých slov s diakritickými znaky pomocí kódování UTF-8 a UTF-16. K dispozici máte Unicode tabulku znaků (UCS).

Detekční a samoopravné kódy

- slouží k **zabezpečení** (binární reprezentace) **dat proti chybám** při jejich přenosu
- chyba = změna bitu
- **detekční kódy**: detekují chyby (změněné bity) v datech, při detekované chybě mohou být data znovu vyžádána (nebo i implicitně pomocí potvrzování správně přijatých dat = **pozitivní potvrzování** a časové prodlevy)
- **samoopravné kódy (error correction code, ECC)**: dále poskytují možnost opravy (jistého množství) chyb a rekonstrukci původních (správných) dat
- kódy bin. reprezentace pro čísla a znaky samy o sobě nejsou zabezpečeny, tzn. změněné (chybné) bity jsou stejně pravděpodobné jako původní (správné)

= (většinou) **redundantní doplnění dat o detekční/samoopravný kód dat**

- příjemce také vypočte kód (i s kódem), pokud je jiný než přijatý (nulový), detekuje/opraví chybu

Detekční a samoopravné kódy

Detekční kódy (error detection codes)

Opakování

- data rozdělena do bloků, bloky opakovány = kód
- příjemce porovná původní (první) a opakované bloky, různé = chyba
- jednoduché, neefektivní, nedetekuje stejné chyby ve všech blocích

Parita

- data rozdělena do bloků, **sudá/lichá** = pro lichý/sudý počet $\mathbf{I}$ v bloku je kód (**paritní bit**) roven $\mathbf{I}$, jinak $\mathbf{0}$
- příjemce provede totéž a porovná paritní bit, různý = chyba
- výpočet paritního bitu pomocí log. operace XOR, příjemce provede XOR i s paritním bitem, nenulový (sudá)/nejedničkový (lichá) = chyba
- např. pro $\mathbf{II0I0}$ je $\mathbf{I}$ (sudá)/$\mathbf{0}$ (lichá)
- detekuje pouze lichý počet chyb
- použití pro detekci chyb při přenosu z/do pamětí

Detekční a samoopravné kódy

Detekční kódy (error detection codes)

Kontrolní součet (checksum)

- sudá parita = log. operace XOR bloků dat
- modulární součet = blok (dvojkového) **doplňkového kódu aritmetického součtu** čísel reprezentovaných bloky dat ve váženém pozičním kódu
- a jiné
- příjemce provede XOR/součet i s kódem, nenulový = chyba
- např. pro **II00 0I0I I0I0** je **00II** (při XOR)/**0I0I** (při aritm. součtu)
- detekuje lichý počet chyb na stejných pozicích v blocích
- nedetekuje změnu pořadí bloků nebo přidání/odebrání nulových bloků

Detekční a samoopravné kódy

Detekční kódy (error detection codes)

Cyklický redundantní součet (Cyclic Redunadacy Check, CRC)

- založen na **cyklických kódech** (vychází z algebraické teorie konečných polí a polynomů nad nimi)
- teoreticky: bity dat reprezentují koeficienty polynomu, který je vydělen tzv. generujícím polynomem řádu n (pro kód řádu n), kód tvoří koeficienty zbytku
- prakticky: za data se přidá blok nul velikosti n (pro kód řádu n), bin. reprezentace generujícího polynomu (divisor) má $n+1$ bitů, od 1. nenulového bitu dat se opakovaně provádí XOR s divisorem dokud nejsou všechny bity dat rovny $\mathbf{0}$, kód $=$ přidaný blok
- příjemce provede totéž s kódem místo bloku nul, nenulový $=$ chyba
- blok např. byte (CRC-8), 2 byte (CRC-16), 4 byte (**CRC-32**) – použití u počítačových sítí a úložných zařízení
- např. pro $\mathbf{II0I00II}$ a divisor $\mathbf{I00II}$ (gen. polynom $x^4 + x + 1$, CRC-4) je $\mathbf{I00I}$
- parita je speciální případ (CRC-1, gen. polynom $x + 1$)

Další: založené na **Hammingově vzdálenosti**, lib. **hashovací funkce** aj.

Detekční a samoopravné kódy

Cyklický redundantní součet (Cyclic Redunadacy Check, CRC)

	data								kód					
$D3_{16}$	**I**	**I**	**0**	**I**	**0**	**0**	**I**	**I**	**0**	**0**	**0**	**0**	0_{16}	
13_{16}	**I**	**0**	**0**	**I**	**I**									gen. polynom $x^4 + x + 1$
$4B_{16}$	**0**	**I**	**0**	**0**	**I**	**0**	**I**	**I**	**0**	**0**	**0**	**0**	0_{16}	XOR
		I	**0**	**0**	**I**	**I**								
7_{16}	**0**	**0**	**0**	**0**	**0**	**I**	**I**	**I**	**0**	**0**	**0**	**0**	0_{16}	XOR
						I	**0**	**0**	**I**	**I**				posun na 1. nenulový bit dat
3_{16}	**0**	**0**	**0**	**0**	**0**	**0**	**I**	**I**	**I**	**I**	**0**	**0**	12_{16}	XOR
							I	**0**	**0**	**I**	**I**			
1_{16}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**I**	**I**	**0**	**I**	**0**	10_{16}	XOR
								I	**0**	**0**	**I**	**I**		
0_{16}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**I**	**0**	**0**	**I**	9_{16}	XOR

Obrázek: CRC-4: postup výpočtu

Detekční a samoopravné kódy

Samoopravné kódy (Error Correction Codes, ECC, Forward Error Correction, FEC)

- použití pro úložná zařízení a u bezdrátové komunikace

Opakování

- většinově se vyskytující blok je správný

Multidimenzionální parita

- data organizována po blocích do mřížky a spočítány parity pro řádky i sloupce
- pro chybný bit jsou chybné řádková i sloupcová parita

0	I	I	I	0
I	I	0	0	I
0	I	I	0	0
0	0	I	I	I
0	I	0	I	I

Obrázek: 2-dimenzionální lichá parita

- n-dimenzionální parita umožňuje opravit $n/2$ chyb

Detekční a samoopravné kódy

Samoopravné kódy (Error Correction Codes, ECC, Forward Error Correction, FEC)

Hammingův kód

- založen na Hammingově vzdálenosti a paritě
- umožňuje detekovat až 2 současné chyby a opravit 1 chybu (Hammingova vzdálenost ≤ 1)
- složitější konstrukce
- použití u operačních pamětí

Další (výkonnější): **Reed-Solomonovy kódy** (CD/DVD, DSL), BCH kódy, konvoluční kódy aj.