

Principy počítačů a operačních systémů

Reprezentace dat

Zimní semestr 2007/2008

Hlavní vlastnosti

- adresovatelné buňky konstatní velikosti
- adresa buňky určena pořadím v paměti
- paměť je jednorozměrná, adresa je skalární

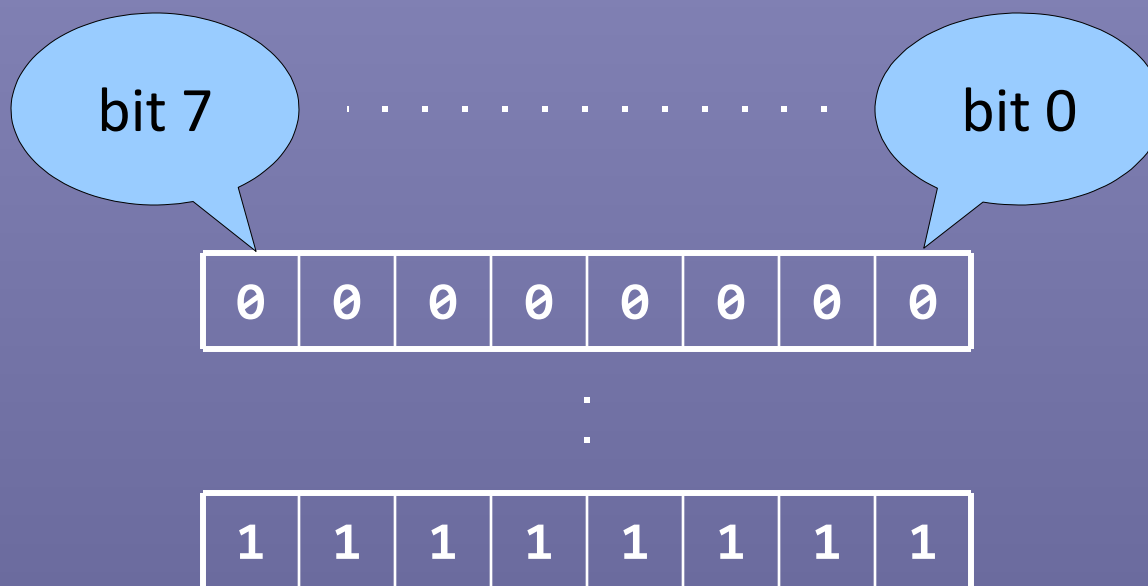
Obsah paměti

- čísla, resp. posloupnosti čísel
- význam určen interpretací
 - ♦ logické hodnoty, znaky, čísla, instrukce
 - ♦ obrázky, zvuky, videa, dokumenty

Paměťová buňka

Hlavní vlastnosti

- typická velikost 8 bitů ~ 1 byte (bajt)
- 1 bit (binary digit) ~ hodnoty 0 a 1
- 8 bitů ~ $2^8 = 256$ hodnot



Logické hodnoty

1 bit informace

- reprezentace jedním bitem

x	x	x	x	0	x	x	x
---	---	---	---	---	---	---	---

x	x	x	x	1	x	x	x
---	---	---	---	---	---	---	---

- ♦ problém s adresací

- reprezentace celou buňkou

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

?	?	?	?	?	?	?	?
---	---	---	---	---	---	---	---

- ♦ problém s velikostí

Znaky

Tisknutelné znaky

- reprezentace textové informace
- velká a malá písmena abecedy, číslice
- diakritika, různé národní abecedy a číslice

Řídící znaky

- ovládání vzdáleného terminálu
 - ♦ přesun na nový řádek
 - ♦ přesun na začátek řádku

Kódování

- zobrazení znaků/číslic na hodnoty v paměti

Příklady kódování

EBCDIC

- 256 závazných znaků
- abeceda není v jednom bloku

ASCII

- původně pro 7 bitů ~ 128 znaků
- rozšíření na 8 bitů, typicky národní znaky

Unicode

- standardizovaná reprezentace (ISO 10646-1, 1993)
- podporuje (téměř) všechny národní abecedy
- 1 znak kódován posloupností více bajtů

Kódování EBCDIC (1)

	0	1	2	3	4	5	6	7
0	NUL	DLE	DS		SP	&	-	
1	SOH	DC1	SOS				\	
2	STX	DC2	FS	SYN				
3	ETX	TM						
4	PF	RES	BYP	PN				
5	HT	NL	LF	RS				
6	LC	BS	ETB	UC				
7	DEL	IL	ESC	EOT				
8		CAN						
9		EM						
A	SMM	CC	SM		¢	!		:
B	VT	CU1	CU2	CU3		\$,	#
C	FF	IFS		DC4	<	*	%	@
D	CR	IGS	ENQ	NAK	()	_	'
E	SO	IRS	ACK		+	;	>	=
F	SI	IUS	BEL	SUB		~	?	"

Kódování EBCDIC (2)

	8	9	A	B	C	D	E	F
0								
1								0
2	a	j			A	J		1
3	b	k	s		B	K	S	2
4	c	l	t		C	L	T	3
5	d	m	u		D	M	U	4
6	e	n	v		E	N	V	5
7	f	o	w		F	O	W	6
8	g	p	x		G	P	X	7
9	h	q	y		H	Q	Y	8
A								
B								
C								
D								
E								
F								

Kódování ASCII (1)

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(8	H	X	h	x
9	HT	EM)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[k	{
C	FF	FS	,	<	L	\	l	
D	CR	GS	-	=	M]	m	}
E	SO	RS	.	>	N	?	n	~
F	SI	US	/	?	O	_	o	DEL

Kódování ASCII (2)

Národní rozšíření

- “horní “ polovina ASCII tabulky
- různá kódování pro různé abecedy
 - ♦ různá kódování pro stejné abecedy

Příklady kódování

- ISO/IEC 8859-1 ... 8859-15
- kódové stránky v MS DOS/Windows
- KOI8-čs, EAST8, ...

Kódování Unicode

Národní abecedy

- Latin, Greek, Cyrillic, ...
- Arabic, Hebrew, ...
- Katakana, Hiragana, ...
- Cherokee, Phoenician, ...
- ... a další

Reprezentace

- UTF-8
 - ♦ proměnný počet bajtů/znak
- UTF-16

	xx0	xx1	xx2	xx3	xx4	xx5	xx6	xx7
0	p	ᄀ	λ	ᄁ	ᄂ	.		
1	p	ᄁ	d	ᄂ	ᄃ	:		
2	q	ᄂ	λ	c	ᄄ	ᄅ	ᄆ	
3	q	ᄃ	o	c	ᄅ	ᄆ	ᄇ	
4	ᄀ	ᄁ	l		ᄂ	ᄃ	ᄄ	
5	ᄁ	ᄂ	i		ᄃ	ᄄ	ᄅ	
6	ᄂ	a	l		ᄄ	ᄅ	ᄆ	
7	ᄃ	a	a		ᄅ	ᄆ	ᄇ	
8	b	ᄄ	ᄅ		ᄆ	ᄇ	ᄈ	
9	b	ᄅ	ᄆ		ᄇ		ᄈ	
A	d	ᄆ	ᄇ		ᄈ	ᄉ	ᄊ	
B	d	ᄇ	ᄈ		ᄉ		ᄊ	
C	ᄀ	ᄁ	ᄂ		ᄃ	ᄄ	ᄅ	
D	ᄁ	ᄂ	ᄃ		ᄄ	ᄅ	ᄆ	
E	ᄂ	ᄃ	ᄄ		ᄅ		ᄆ	
F	ᄃ	ᄄ	ᄅ		ᄆ			

Číselné soustavy (1)

Polyadické číselné soustavy

- poziční systém s jedním nebo více základy (radix number system)
- číslo A reprezentuje uspořádaná $(n+m)$ -tice koeficientů a_i
- pokud má soustava pouze jeden základ z , odpovídají z_i hodnoty $z_i = z^i$
- pokud $m = 0$, potom A je celé číslo
- pokud $m > 0$, potom A má zlomkovou část

$$A = \sum_{i=-m}^{i=n-1} a_i z_i$$

Číselné soustavy (2)

Nepolyadické číselné soustavy

- římské číslice
 - ♦ 1648 ~ MCDXLVIII, 2003 ~ MMIII, 9 ~ IX, VIII
 - ♦ nevhodné pro algebraické operace
- soustava zbytkových tříd
 - ♦ k různých základů – prvočísel
 - ♦ číslo vyjádřeno k -ticí zbytků po dělení základy
 - ♦ jednoznačné pouze pro čísla menší než součin základů
 - soustava o základech 2, 3, 11 ~ 9 zapsáno jako 109
 - ♦ rychlejší aritmetické operace (chybí přenos mezi řády)
 - nejednoznačné dělení, problematické porovnávání, časově náročný převod do/z soustavy

Převod mezi soustavami (1)

Přepis celého čísla

- zápis čísla A v soustavě o základu z

$$\begin{aligned} A &= a_{n-1} z^{n-1} + a_{n-2} z^{n-2} + \dots + a_1 z^1 + a_0 z^0 \\ &= (a_{n-1} z^{n-2} + a_{n-2} z^{n-3} + \dots + a_1) z + a_0 = \dots \\ &= (((a_{n-1} z + a_{n-2}) z + \dots + a_2) z + a_1) z + a_0 \end{aligned}$$

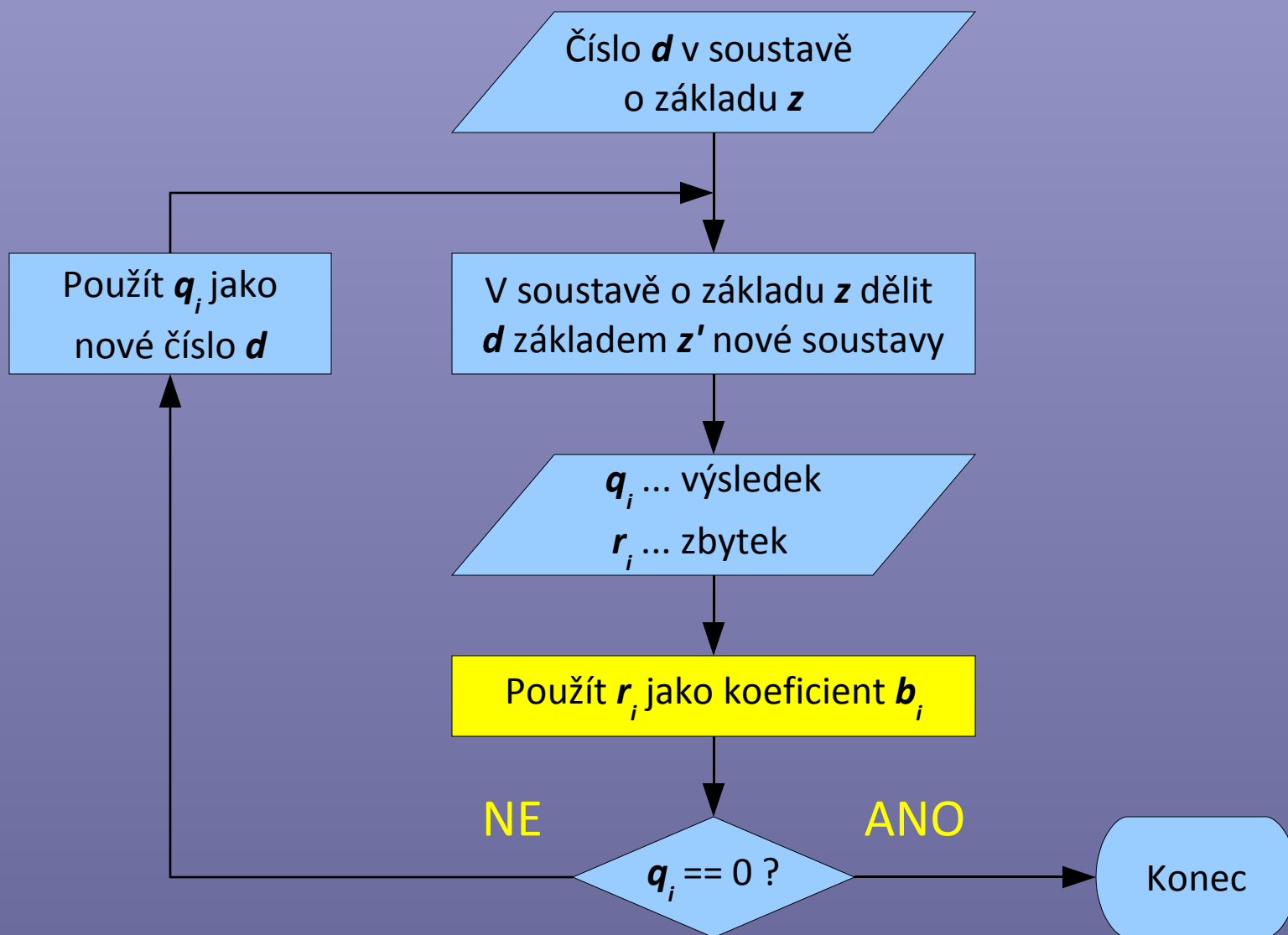
- zápis čísla A v soustavě o základu z'

$$A = (((b_{n-1} z' + b_{n-2}) z' + \dots + b_2) z' + b_1) z' + b_0$$

- rekursivní převod

$$A = \left\lfloor \frac{A}{z'} \right\rfloor z' + A \bmod z'$$

Převodní algoritmus celé části



Příklady převodu celé části

$$151_{10} = ???_2$$

$$151 : 2 = 75 \quad b_0 = 1$$

$$75 : 2 = 37 \quad b_1 = 1$$

$$37 : 2 = 18 \quad b_2 = 1$$

$$18 : 2 = 9 \quad b_3 = 0$$

$$9 : 2 = 4 \quad b_4 = 1$$

$$4 : 2 = 2 \quad b_5 = 0$$

$$2 : 2 = 1 \quad b_6 = 0$$

$$1 : 2 = 0 \quad b_7 = 1$$

$$151_{10} = 10010111_2$$

$$134_9 = ???_7$$

$$134 : 7 = 17 \quad b_0 = 0$$

$$17 : 7 = 2 \quad b_1 = 2$$

$$2 : 7 = 0 \quad b_2 = 2$$

$$134_9 = 220_7$$

Převod do desítkové soustavy

Přepis celého čísla

- koeficienty a_i zápisu čísla o základu z vynásobíme odpovídající (i -tou) mocninou z a sečteme

$$A \approx a_{n-1} a_{n-2} \dots a_1 a_0$$

$$A = a_{n-1} z^{n-1} + a_{n-2} z^{n-2} + \dots + a_1 z^1 + a_0 z^0$$

- Hornerovo schéma

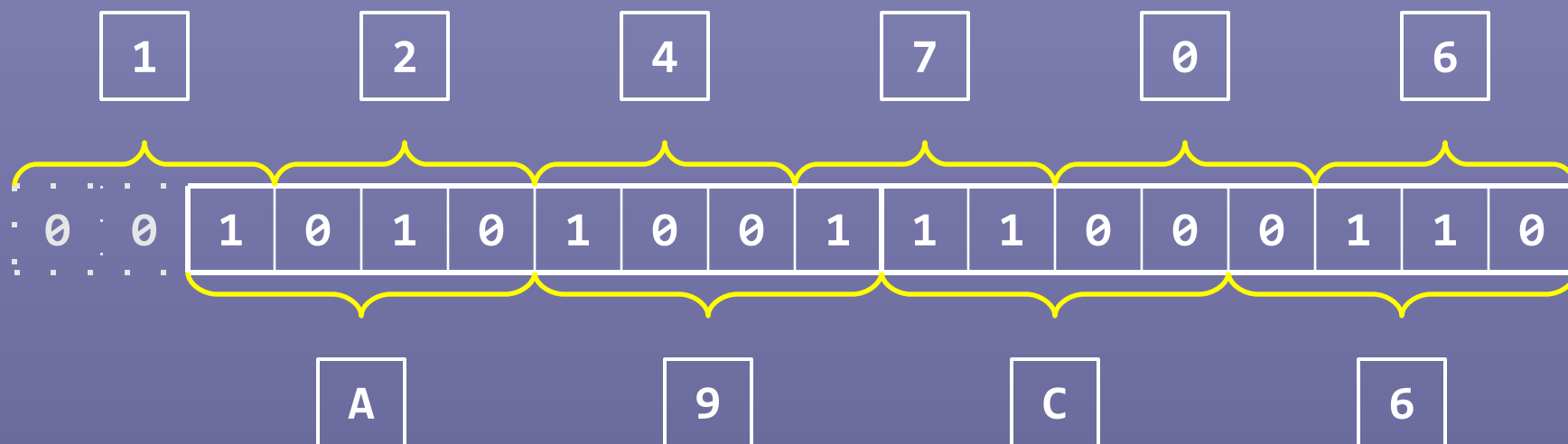
$$A = (((a_{n-1} z + a_{n-2}) z + \dots + a_2) z + a_1) z + a_0$$

Často používané soustavy

Základy odvozené od mocniny 2

- dvojková/binární, $z = 2$
- osmičková/oktalová, $z = 8$
- šestnáctková/hexadecimální, $z = 16$

Příklad zápisu čísla 43463_{10}



Převod mezi soustavami (2)

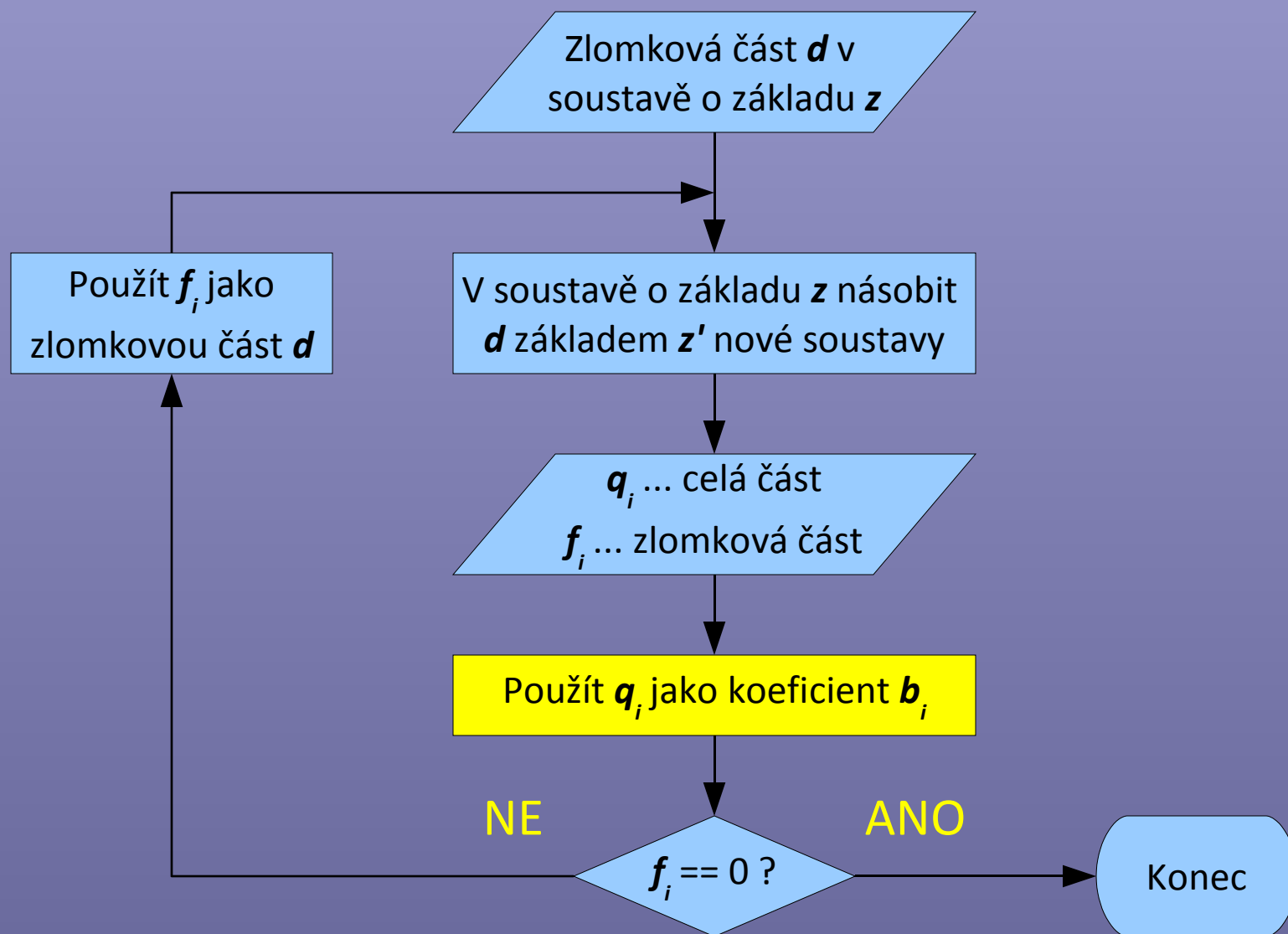
Přepis zlomkové části

- zápis čísla A v soustavě o základu z

$$A = a_{n-1}z^{n-1} + \dots + a_0z^0 + a_{-1}z^{-1} + \dots + a_{-m}z^{-m}$$

- pro zápis v soustavě o základu z' hledáme koeficienty b_i pro $-m \leq i \leq n-1$
- převod celé části pro $0 \leq i \leq n-1$
- převod zlomkové části pro $-m \leq i < 0$
 - ♦ základem nové soustavy se v původní soustavě **násobí**

Převodní algoritmus zlomkové části



Příklady převodu zlomkové části (1)

$$0,1_{10} = ???_2$$

$$0,1 * 2 = 0,2 \quad b_{-1} = 0$$

$$0,2 * 2 = 0,4 \quad b_{-2} = 0$$

$$0,4 * 2 = 0,8 \quad b_{-3} = 0$$

$$0,8 * 2 = 1,6 \quad b_{-4} = 1$$

$$0,6 * 2 = 1,2 \quad b_{-5} = 1$$

$$0,2 * 2 = 0,4 \quad b_{-6} = 0$$

...

Postup nemusí skončit!

$$0,1_{10} \approx 0,000110011_2$$

$$0,678_{10} = ???_2$$

$$0,678 * 2 = 1,356 \quad b_{-1} = 1$$

$$0,356 * 2 = 0,712 \quad b_{-2} = 0$$

$$0,712 * 2 = 1,424 \quad b_{-3} = 1$$

$$0,424 * 2 = 0,848 \quad b_{-4} = 0$$

$$0,848 * 2 = 1,696 \quad b_{-5} = 1$$

$$0,696 * 2 = 1,392 \quad b_{-6} = 1$$

$$0,392 * 2 = 0,784 \quad b_{-7} = 0$$

$$0,784 * 2 = 1,568 \quad b_{-8} = 1$$

...

$$0,678_{10} \approx 0,10101101_2$$

Příklady převodu zlomkové části (2)

$$0,1_3 = ???_{10}$$

$$0,1 * 101 = 10,1 \quad b_{-1} = 10_3 = 3_{10}$$

$$0,1 * 101 = 10,1 \quad b_{-2} = 10_3 = 3_{10}$$

$$0,1 * 101 = 10,1 \quad b_{-3} = 10_3 = 3_{10}$$

...

$$0,1 * 101 = 10,1 \quad b_{-m} = 10_3 = 3_{10}$$

Postup opět nekončí!

$$0,1_3 \approx 0,33\dots33_{10}$$

Vědecká notace

Standardizovaný zápis desítkových čísel

- zápis ve formě $m \times 10^e$
- mantisa m
 - ♦ pouze platné číslice, počet číslic odpovídá přesnosti
 - ♦ normalizovaný zápis $\sim 1 \leq m < 10$
- exponent e
 - ♦ reprezentuje pozici desetinné čárky
- pro $z \neq 10 \sim \textit{floating point}$

Zápis čísla 0,0000000004945

- normalizovaný zápis $\sim 4,945 \cdot 10^{-9}$

Aritmetické operace v jiných soustavách

Sčítání, odčítání, násobení, dělení

- princip stejný jako v desítkové soustavě
- změna základu nemění algebraické vlastnosti
- kladná a záporná čísla rozlišena znaménkem

Obecné koncepty

- přenos a výpůjčka do/z vyššího řádu

Reprezentace čísel

Různé typy čísel

- přirozená, celá
- racionální, reálná

Reprezentace

- řetězce znaků
 - ♦ 1,000,000,000 ~ 10 znaků = 10 bajtů = 80 bitů
- binární reprezentace
 - ♦ 1,000,000,000 ~ 32 bitů = 4 bajty = 4 znaky

Zobrazení přirozených čísel (1)

Binary Coded Decimal

- převod jednotlivých cifer do dvojkové soustavy
 - ♦ cifry 0, 1, ..., 9 zobrazeny jako 0000, 0001, ..., 1001
 - ♦ použití pro vstup/výstup, přesná čísla (bankovníctví)
- problémy
 - ♦ “horní” 4 bity nevyužity ~ packed BCD
 - ♦ složitější obvody pro aritmetické operace
 - 15-20% pro sčítání, složitější algoritmy pro násobení

1930₁₀ » packed BCD

0	0	0	1	1	0	0	1	0	0	1	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Zobrazení přirozených čísel (2)

Přímá reprezentace

- převod do dvojkové soustavy
- uložení do paměti
 - ♦ buňka paměti umožňuje zápis čísel $[0,255]$
 - ♦ pro zápis větších čísel nutno použít více buněk

19_{10} »

0	0	0	1	0	0	1	1
---	---	---	---	---	---	---	---

229_{10} »

1	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---

Zobrazení celých čísel (1)

Nejčastější zobrazení

- se znaménkovým bitem (*sign and magnitude*)
- s posunutím (*biased notation*)
 - ♦ porovnávání čísel s plovoucí řádovou čárkou
- doplňkové (*complementary notation*)
 - ♦ jedničkový doplněk (*ones complement*)
 - ♦ dvojkový doplněk (*two's complement*)
 - ♦ odčítání celých čísel pomocí sčítání

Zobrazení celých čísel (2)

Přímé zobrazení se znaménkovým bitem

- číslo převedeno do dvojkové podoby
- reprezentace doplněna o *znaménkový bit*



Vlastnosti

- dvě reprezentace nuly
- složitější aritmetické operace

Zobrazení celých čísel (3)

Zobrazení s posunutím

- k číslu se přičte *konstanta reprezentující nulu*
- výsledek se převede do binární podoby

-113_{10} s posunutím 127 »

0	0	0	0	1	1	1	0
---	---	---	---	---	---	---	---

při vhodném posunutí
jsou kladná čísla větší

Vlastnosti

- zachovává uspořádání čísel
- složitější aritmetické operace

Zobrazení celých čísel (4)

Zobrazení v jedničkovém doplňku

- přirozená čísla jsou zobrazena přímo
- záporná čísla zobrazena v jedničkovém doplňku
 - ♦ inverze bitů přímého zobrazení

-113_{10} »

1	0	0	0	1	1	1	0
---	---	---	---	---	---	---	---

Vlastnosti

bit 7

bit 0

- dvě reprezentace nuly
- stačí pouze obvod pro sčítání
- nejvyšší bit indikuje znaménko

Zobrazení celých čísel (5)

Zobrazení ve dvojkovém doplňku

- přirozená čísla jsou zobrazena přímo
- záporná čísla zobrazena ve dvojkovém doplňku
 - ♦ inverze bitů přímého zobrazení, přičtení 1

-113_{10} »

1	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---

Vlastnosti

bit 7

bit 0

- pouze jedna reprezentace nuly
- stačí pouze obvod pro sčítání
- nejvyšší bit indikuje znaménko

Příklady zobrazení v dvojkové soustavě

číslo	znaménko	posun 8	inverze	doplňěk
+7	0111	1111	0111	0111
+6	0110	1110	0110	0110
+5	0101	1101	0101	0101
+4	0100	1100	0100	0100
+3	0011	1011	0011	0011
+2	0010	1010	0010	0010
+1	0001	1001	0001	0001
+0	0000	1000	0000	0000
-0	1000	1000	1111	0000
-1	1001	0111	1110	1111
-2	1010	0110	1101	1110
-3	1011	0101	1100	1101
-4	1100	0100	1011	1100
-5	1101	0011	1010	1011
-6	1110	0010	1001	1010
-7	1111	0001	1000	1001
-8	----	0000	----	1000

Znaménkové rozšíření

Rozšíření reprezentace čísla

- nutno zachovat původní číslo
- přirozená čísla doplněna 0 do potřebné šířky

Celá čísla

- zobrazení se znaménkem
 - ♦ znaménko se přesune do širší reprezentace
 - ♦ zbytek včetně znaménkového bitu doplněn 0
- zobrazení v doplňkovém kódu
 - ♦ číslo doplněno hodnotou nejvyššího bitu (reprezentuje znaménko) do potřebné šířky
 - ♦ platí pro jedničkový i dvojkový doplněk

Příklad znaménkového rozšíření

Přímé zobrazení se znaménkem

-18_{10} »

1	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---

1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Zobrazení ve dvojkovém doplňku

$+18_{10}$ »

0	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

-18_{10} »

1	1	1	0	1	1	1	0
---	---	---	---	---	---	---	---

1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Problémy omezené reprezentace

Přetečení

- hodnoty nelze zobrazit v daném počtu bitů
- vzniká při aritmetických operacích

Příklad

- zobrazit výsledek součtu $3 + 6$
 - ♦ přímo s pomocí 3 bitů (vyžaduje 4 bity)
- zobrazit č. 9
 - ♦ se znaménkem s pomocí 4 bitů (vyžaduje 5 bitů)

Vícebajtové posloupnosti

Ukládání čísel většího rozsahu

- 2 bajty ~ 16 bitů $\sim 2^{16} = 65\,536$ hodnot
- 4 bajty ~ 32 bitů $\sim 2^{32} = 4\,294\,967\,296$ hodnot

V jakém pořadí ukládat bajty?

- od nejvyšších “řádů”
- od nejnižších “řádů”
- ... jinak

... po jakých částech?

Ukládání vícebajtových posloupností

Big Endian

- s rostoucí adresou klesá význam bajtů
- IBM 360/370, PDP-10, Motorola 6800, různé RISC

Little Endian

- s rostoucí adresou roste význam bajtů
- VAX, MOS 6502, Intel x86

Middle Endian

- různá uspořádání, raději nemluvit...

Příklad vícebajtové posloupnosti

Posloupnost

- $1234ABCD_{16} \sim 32\text{-bitů} \sim 4 \text{ bajty}$
- uložit do paměti na adresy $N, N+1, N+2, N+3$
- části odpovídají velikosti adresovatelné buňky
 - ♦ jak by to vypadalo v případě 16-bitových slov?

Little Endian

- posloupnost bajtů $CD_{16}, AB_{16}, 34_{16}, 12_{16}$

Big Endian

- posloupnost bajtů $12_{16}, 34_{16}, AB_{16}, CD_{16}$

Problémy s vícebajtovými posloupnostmi

Přenositelnost software a dat

- předpoklad konkrétní architektury v software
- použití dat vytvořených na jiné architektuře
 - ♦ souborové systémy, síťové protokoly
 - ♦ datové struktury obecně

Host vs. Network order

- **network order** ~ big endian
- **host order** ~ odpovídá architektuře
- explicitní konverze při odesílání a příjmu dat

Bi-endian architektury

Podporují Little i Big Endian

- ARM, PowerPC, DEC Alpha, MIPS, PA-RISC, IA-64
- určeno nastavením v řídicím registru nebo zapojením hardware
- výchozí nastavení může záviset na OS
 - ♦ BE: MIPS + IRIX, PA-RISC, většina PowerPC
 - ♦ LE: MIPS + Ultrix, IA-64 + Linux
- někdy je dáno návrhem systémového HW
 - ♦ LE: DEC Alpha

Zobrazení reálných čísel (1)

S pevnou řádovou čárkou

- pozice čárky určuje rozsah a přesnost zobrazení
 - ♦ n cifer celé části, m cifer zlomkové části
 - ♦ interval $0 \leq x \leq 2^n - 2^{-m}$
- celá čísla ~ speciální případ pro $m = 0$



Použití

- úlohy s omezenými nároky na přesnost
- hardware bez podpory *floating point* operací

Zobrazení reálných čísel (2)

S pohyblivou řádovou čárkou

- “vědecká notace” v nedesítkových soustavách
- mantisa m , exponent e , základ θ , přesnost p
 - ♦ základ θ určuje základ mantisy i exponentu
 - ♦ přesnost p určuje počet platných míst mantisy

Příklady

- 0,9 pro $\theta = 10, p = 3$
 - ♦ $9,00 \times 10^{-1}$
- 0,1 pro $\theta = 2, p = 24$ (nelze přesně)
 - ♦ $1,10011001100110011001100110\mathbf{1} \times 2^{-4}$

Zápis čísla s pohyblivou řádovou čárkou

Obecný tvar

$$\pm d_0, d_1 d_2 \dots d_{p-1} \times \beta^e \quad 0 \leq d_i < \beta$$

- reprezentuje číslo

$$\pm (d_0 + d_1 \beta^{-1} + d_2 \beta^{-2} + \dots + d_{p-1} \beta^{-(p-1)}) \beta^e$$

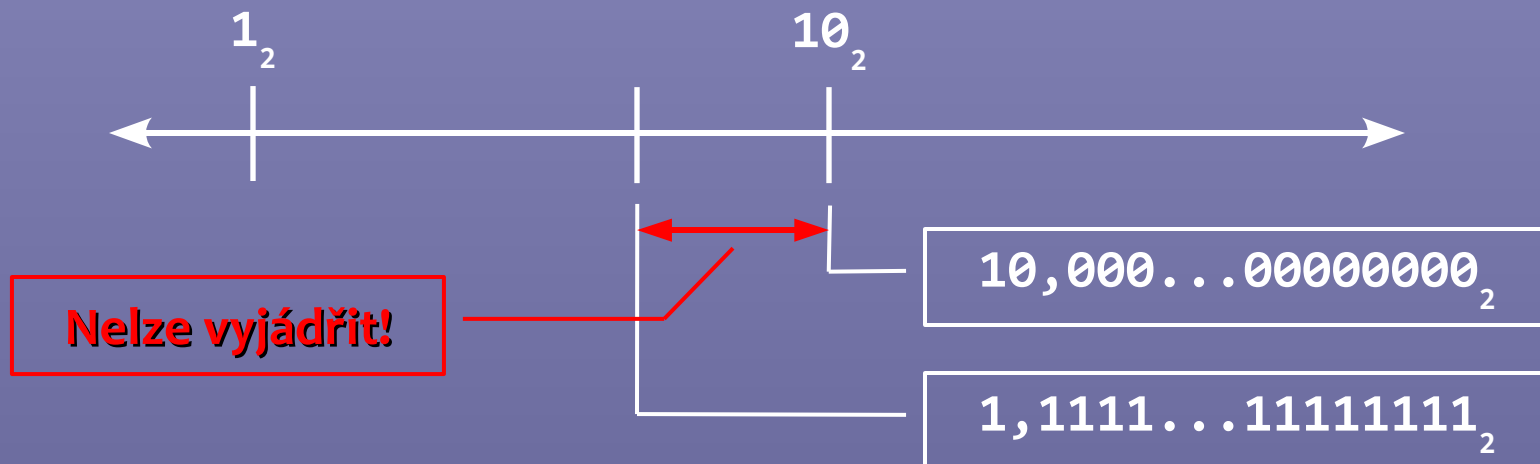
Mantisa $m = d_0, d_1 d_2 \dots d_{p-1}$

- pokud $d_0 = 0$, reprezentace není **nenormalizovaná**
- pokud $d_0 \neq 0$, reprezentace je **normalizovaná**

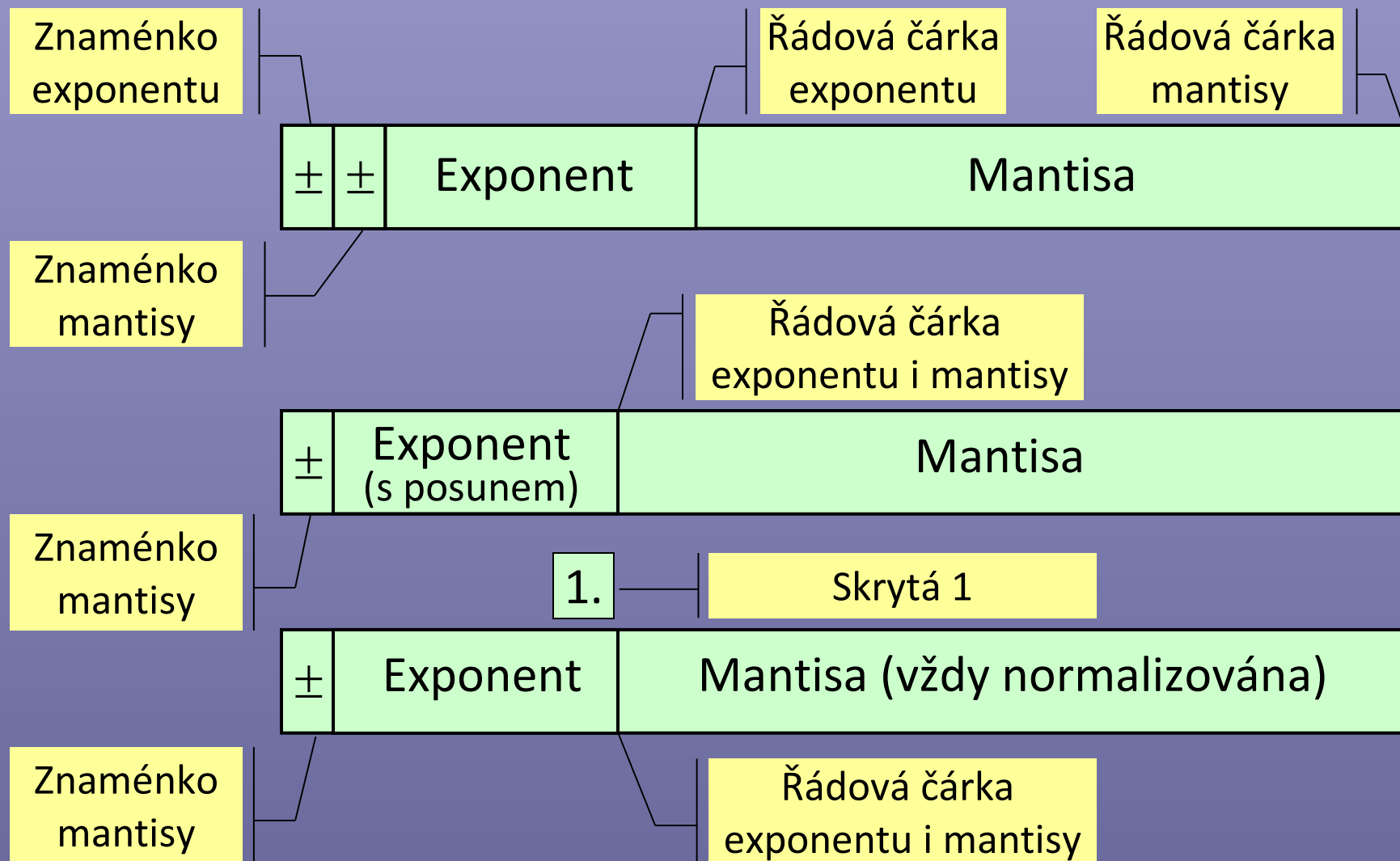
Vlastnosti čísel s pohyblivou řádovou čárkou

Hlavní vlastnosti

- velký rozsah zobrazitelných čísel
 - ♦ stejná přesnost všech čísel
- čísla netvoří kontinuum
 - ♦ pouze přibližné vyjádření některých čísel



Formáty čísel s pohyblivou řádovou čárkou



Převod do reprezentace s p. řádovou čárkou

Obecný postup

- převést absolutní hodnotu čísla do dvojkové reprezentace
 - ♦ celou i zlomkovou část
- normalizovat dvojkovou reprezentaci
 - ♦ výsledkem je mantisa m_2 a exponent e_{10}
- zaokrouhlit mantisu na přesnost p
 - ♦ při použití skryté 1 nejprve “odtrhnout” úvodní 1
- převést exponent do požadovaného zobrazení
 - ♦ posun nebo znaménkové rozšíření
- nastavit znaménkový bit

IEEE Standard 754 (1)

IEEE Std. for Binary Floating-Point Arithmetics

- formát čísel
 - ♦ speciální hodnoty ± 0 , $\pm \infty$, NaN
 - ♦ denormalizovaná čísla (mantisa nezačíná 1)
- operace nad čísly
 - ♦ a se speciálními hodnotami
- zaokrouhlovací režimy
- výjimečné stavy

IEEE Standard 754 (2)

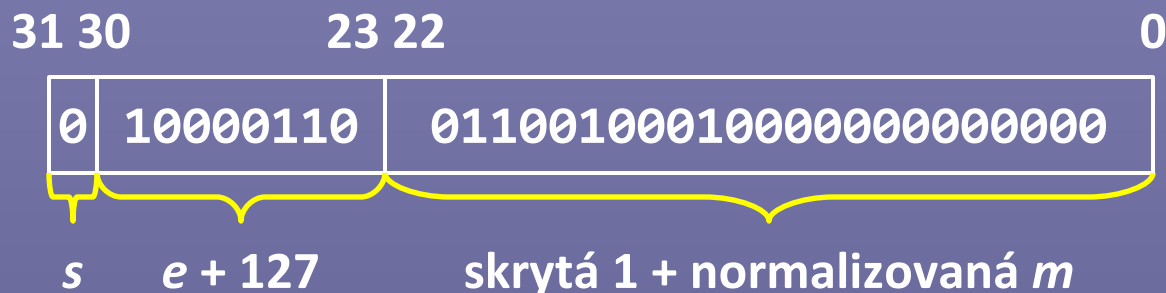
Standardizované formáty

- single precision: $\theta = 2, p = 24, |e| = 8 (+127)$
 - ♦ rozsah $\sim \pm 10^{38}$, přesnost ~ 7 desetinných míst
- double precision: $\theta = 2, p = 53, |e| = 11 (+1023)$
 - ♦ rozsah $\sim \pm 10^{308}$, přesnost ~ 17 desetinných míst
- vnitřní reprezentace (bez skrytého bitu)
 - ♦ single extended: $\theta = 2, p \geq 32, |e| \geq 11$ bitů
 - ♦ double extended: $\theta = 2, p \geq 64, |e| \geq 15$ bitů
- podobné (nestandardní) formáty
 - ♦ quad precision: $\theta = 2, p = 113, |e| = 15$ bitů
 - ♦ half precision: $\theta = 2, p = 11, |e| = 5$ bitů

IEEE Standard 754 (3)

Příklad zobrazení čísla

- desetinný zápis: 178,125
- dvojkový zápis: 10110010,001
- normalizovaný dvojkový zápis: $1,0110010001_2 \times 2^7$
 - ♦ exponent $e = 7_{10} = 111_2$
- zápis ve formátu *single precision*
 - ♦ přesnost 24b, exponent s posunem +127, skrytá 1



IEEE Standard 754 (4)

Definované hodnoty

exponent		mantisa	význam
$e_{min}-1$	000...000	$m = 0$	± 0
$e_{min}-1$		$m \neq 0$	$0, m \times 2^{e_{min}}$
$\langle e_{min}, e_{max} \rangle$		m	$1, m \times 2^e$
$e_{max}+1$	111...111	$m = 0$	$\pm \infty$
$e_{max}+1$		$m \neq 0$ 1xxx...xxx	QNaN
$e_{max}+1$		$m \neq 0$ 0xxx...xxx	SNaN

IEEE Standard 754 (5)

Příčiny vzniku NaN

- NaN nemusí generovat přímo procesor

operace	vznik NaN
+, -	$\infty + (-\infty)$
\times	$0 \times \pm\infty$
/	$0/0, \pm\infty/\pm\infty$
REM	$x \text{ REM } 0, \infty \text{ REM } y$
$\sqrt{}$	\sqrt{x} pro $x < 0$

Propagace NaN v operacích

- zjednodušení řízení výpočtu, pouze QNaN

D. Goldberg

- What Every Computer Scientist Should Know About Floating-Point Arithmetic

Intel Corporation

- IA-32 Intel Architecture Software Developer's Manual (Vol.1 – Basic Architecture)