

Reprezentace dat

INP 2019
FIT VUT v Brně



- Kód
- Čísla v pevné řádové čárce (FX)
- Kód zbytkových tříd
- Desítková čísla dvojkově kódovaná
- Huffmanův kód
- Čísla v pohyblivé řádové čárce (FP)
- Standard IEEE 754

2

Pojem kód a typy kódů

- **Kód** je vzájemně jednoznačné přiřazení mezi symboly dvou množin.
- **Data** reprezentujeme pomocí kódů, které můžeme mj. zhruba rozdělit do dvou skupin:
 - kódy **pro vnější přenos dat (znaky)** (ASCII, UNICODE atd.)
 - kódy **pro vnitřní reprezentaci dat** (doplňkový kód, BCD atd.),
- přičemž pro **čísla** zadaná ve formátu s **pohyblivou řádovou čárkou** (FP) se používají jiné kódy než pro čísla s **pevnou řádovou čárkou** (FX).

Kódy pro znaky dle Wikipedie

| | |
|-----------------------------|---|
| Early telecommunications | ASCII • ISO/IEC 646 • ISO/IEC 6937 • T.61 • BCD (6-bit) • Baudot code • Morse code • Chinese telegraph code |
| ISO/IEC 8859 | - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14 - 15 - 16 |
| Bibliographic use | ANSEL • ISO 5426 / 5426-2 / 5427 / 5428 / 6438 / 6861 / 6862 / 10585 / 10586 / 10754 / 11822 • MARC-8 |
| National standards | ArmSCII • CNS 11643 • GOST 10859 • GB 2312 • HKSCS • ISCII • JIS X 0201 • JIS X 0208 • JIS X 0212 • JIS X 0213 • KPS 9566 • KS X 1001 • PASCII • TIS-620 • TSCII • VISCI • YUSCII |
| EUC | CN • JP • KR • TW |
| ISO/IEC 2022 | CN • JP • KR • CCCII |
| MacOS codepages ("scripts") | Arabic • CentralEurRoman • ChineseSimp / EUC-CN • ChineseTrad / Big5 • Croatian • Cyrillic • Devanagari • Dingbats • Farsi • Greek • Gujarati • Gurmukhi • Hebrew • Icelandic • Japanese / ShiftJIS • Korean / EUC-KR • Roman • Romanian • Symbol • Thai / TIS-620 • Turkish • Ukrainian |
| DOS codepages | 437 • 667 • 668 • 720 • 737 • 770 • 773 • 775 • 790 • 819 • 850 • 851 • 852 • 853 • 854 • 855 • 857 • 858 • 860 • 861 • 862 • 863 • 864 • 865 • 866 • 867 • 868 • 869 • 872 • 895 • 912 • 915 • 932 • 931 • Kamenický • Mazovia • MK • Iran System |
| Windows codepages | 674 / TIS-620 • 932 / ShiftJIS • 936 / GBK • 949 / EUC-KR • 950 / Big5 • 1250 • 1251 • 1252 • 1253 • 1254 • 1255 • 1256 • 1257 • 1258 • 28604 • 54936 / GB18030 |
| EBCDIC codepages | 37/1140 • 273/1141 • 277/1142 • 278/1143 • 280/1144 • 284/1145 • 285/1146 • 297/1147 • 420/16804 • 424/12712 • 500/1148 • 838/1160 • 871/1149 • 875/9067 • 930/1390 • 933/1364 • 937/1371 • 935/1388 • 939/1399 • 1025/1154 • 1026/1155 • 1047/924 • 1112/1156 • 1122/1157 • 1123/1158 • 1130/1164 • JEF • KEIS |
| Platform specific | ATASCII • CDC display code • DEC-MCS • DEC Radix-50 • ELWRO-Junior • Fieldata • GSM 03.38 • HP roman8 • PETSCII • TI calculator character sets • WISCII • ZX Spectrum character set |
| Unicode ISO/IEC 10646 | UTF-8 • UTF-16/UCS-2 • UTF-32/UCS-4 • UTF-7 • UTF-1 • UTF-EBCDIC • GB 18030 • SCSU • BOCU-1 |
| Miscellaneous codepages | APL • Cork • HZ • IBM code page 1133 • KOI8 • TRON |
| Related topics | control character (C0 C1) • CCSID • Character encodings in HTML • charset detection • Han unification • ISO 6429/IEC 6429/ANSI X3.64 • mojibake |

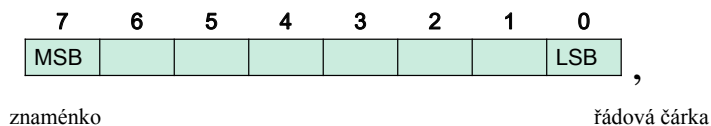
3

4

Opakování FX: Základní kódy

Př. Obrazy +7 a -7 na 8 bitech včetně znaménka

| | +7 | -7 |
|------------------------------|-----------|-----------|
| Přímý kód se znaménkem | 0000 0111 | 1000 0111 |
| Inverzní kód (1- doplněk) | 0000 0111 | 1111 1000 |
| Dvojkový doplňkový kód | 0000 0111 | 1111 1001 |
| Kód se sudým posunutím (128) | 1000 0111 | 0111 1001 |
| Kód s lichým posunutím (127) | 1000 0110 | 0111 1000 |



5

Význam kódových kombinací (8 bitů)

| 76543210 | Význam v kódu | | | | |
|----------|---------------|----------|-----------|---------------------|---------------------|
| | Přímý se zn. | Inverzní | Doplňkový | Se sud. pos. 128 | S lich. pos. 127 |
| 00000000 | 0 | 0 | 0 | -128 | -127 |
| 00000001 | 1 | 1 | 1 | -127 | -126 |
| 00000010 | 2 | 2 | 2 | -126 | -125 |
| ... | ... | ... | ... | ... | ... |
| 01111110 | | | | -2 | -1 |
| 01111111 | 127 | 127 | 127 | -1 | 0 |
| 10000000 | -0 | -127 | -128 | 0 | +1 |
| 10000001 | -1 | -126 | -127 | +1 | +2 |
| ... | ... | ... | ... | ... | ... |
| 11111110 | -126 | -1 | -2 | 126 | 127 |
| 11111111 | -127 | -0 | -1 | 127 | 128 |

6

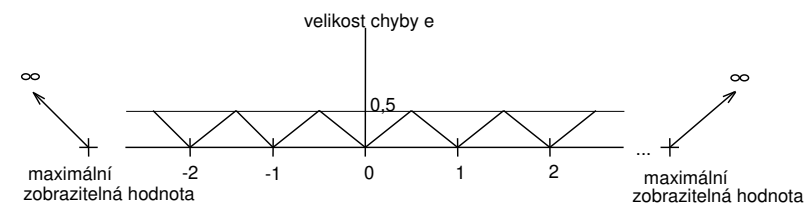
Chyby zobrazení čísla FX

- Zobrazené číslo FX je zatíženo třemi typy chyb:
 - chyba měření:** vzniká při pořizování čísla vlivem chyby metody měření
 - chyba stupnice** (scaling): číselná soustava nemůže na konečném počtu míst vyjádřit přesně všechny hodnoty
 - chyba zanedbáním** (truncation = odseknutí) a **zaokrouhlením** (rounding)

7

Chyba stupnice

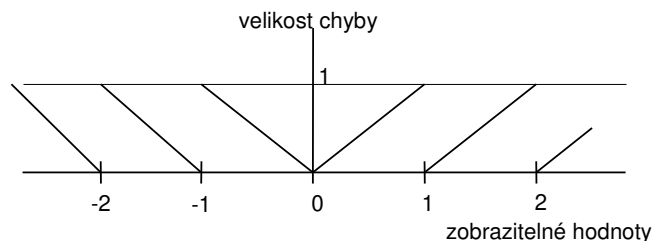
- Na obrázku je průběh funkce **chyby stupnice** pro celá čísla s pevnou řádovou čárkou.
- Vidíme, že jsou přesně vyjádřena pouze celá čísla 0, 1, 2, 3 ..., kdy chyba = 0. Např. obraz čísla 1,5 má maximální velikost chyby, a to 0,5.
- Průběh funkce zobrazující velikost chyby je lineární mezi body celých čísel a čísla 0,5, 1,5, atd. Od obrazu největšího (a nejmenšího) zobrazitelného čísla začíná chyba lineárně růst nad všechny meze.



8

Chyba zanedbáním

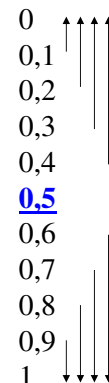
- Vidíme, že chyba roste v intervalu $<0,1)$ od nuly do 1, a obdobně v dalším intervalu $<1,2)$, atd.



Pozn: Způsob zaokrouhlování je věcí konvence

Statistické zaokrouhlení

Čísla "přesně uprostřed" zaokrouhluje jednou nahoru a jednou dolů. To se obvykle v praxi dělá zaokrouhlením na sudé (např. v normě IEEE), nebo na liché číslo.



Př. Zaokrouhlení k **sudému** číslu:

1,35 → 1,4

1,45 → 1,4

9

10

Nepolyadické soustavy

Příkladem nepolyadické soustavy je soustava římských číslic

I V X L C D M
1 5 10 50 100 500 1000

Tato soustava je pro počítání nevhodná.

Použitelná nepolyadická soustava je **soustava zbytkových tříd** (RNS - Residue Number System), označovaná jako **kód zbytkových tříd** KZT.

Soustava je definovaná pomocí uspořádané k -tice vzájemně různých **prvočíselných** základů $z_0 \dots z_{k-1}$. Obrazem čísla A je uspořádaná k -tice celých čísel $a_0 a_1 a_2 \dots a_{k-1}$, pro která platí $a_i = A \bmod z_i$

| | z_0 | z_1 | z_2 |
|-----|-------|-------|-------|
| | 2 | 3 | 5 |
| A | a_0 | a_1 | a_2 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 0 | 2 | 2 |
| 3 | 1 | 0 | 3 |
| 4 | 0 | 1 | 4 |
| 5 | 1 | 2 | 0 |
| 6 | 0 | 0 | 1 |
| 7 | 1 | 1 | 2 |
| 8 | 0 | 2 | 3 |
| 9 | 1 | 0 | 4 |
| 10 | 0 | 1 | 0 |
| 11 | 1 | 2 | 1 |
| 12 | 0 | 0 | 2 |
| 13 | 1 | 1 | 3 |
| ... | | | |

11

Soustava zbytkových tříd

Př. Máme zadány základy 2, 3, 5. Zbytkové třídy pak jsou:

$\{0, 1\}$ pro 2

$\{0, 1, 2\}$ pro 3

$\{0, 1, 2, 3, 4\}$ pro 5

Např. číslo 5 pak vyjádříme trojicí zbytků po dělení zadanými základy, tedy $(1 \ 2 \ 0)$. Jednoznačně lze vyjádřit pouze číslo A , pro které platí

$$A < \prod_i z_i$$

pro všechna i , tedy pouze číslo, které je menší než tzv. perioda, v našem případě je to $2 \cdot 3 \cdot 5 = 30$.

Soustava zbytkových tříd umožňuje rychlé operace sčítání, odčítání a násobení, **protože se neuplatňují přenosy mezi jednotlivými stupni**.

Dělení není jednoznačně definovaná operace, rovněž porovnávání velikosti čísel je prakticky obtížné.

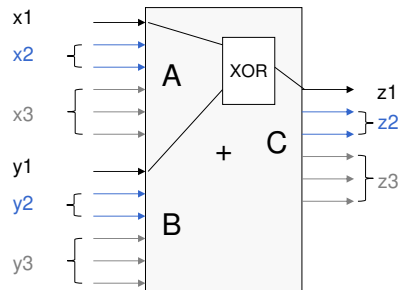
Doba převodu do a zpět ze soustavy KZT může často pohltit časovou úsporu získanou na rychlých aritmetických operacích.

Rychlých operací v KZT se používá ve speciálních případech, např. v kryptografii (RSA provádí operace na 2048 bitech).

12

Realizace sčítačky v KZT(2|3|5)

$$(z1|z2|z3)_{KZT(2|3|5)} = (x1|x2|x3)_{KZT(2|3|5)} + (y1|y2|y3)_{KZT(2|3|5)}$$



Výpočet z2 (operandy i výsledek na 2b)

| x2 | y2 | z2 |
|----|----|----|
| ab | cd | uv |
| 00 | 00 | 00 |
| 00 | 01 | 01 |
| 00 | 10 | 10 |
| 01 | 00 | 01 |
| 01 | 01 | 10 |
| 01 | 10 | 00 |
| 10 | 00 | 10 |
| 10 | 01 | 00 |
| 10 | 10 | 01 |

$$u = f1(a,b,c,d)$$

$$v = f2(a,b,c,d)$$

Výpočet z1 (operandy i výsledek 1b)

| x1 | y1 | z1 |
|----|----|----|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$$z1 = x1 \text{ XOR } y1$$

Výpočty z1, z2 a z3 jsou
vzájemně nezávislé =>
rychlé!

Výpočet z3 (operandy i výsledek na 3b)
obdobně jako z2

$$r = f1(e,f,g,h,i,j)$$

$$s = f2(e,f,g,h,i,j)$$

$$t = f3(e,f,g,h,i,j)$$

13

Desítková čísla dvojkově kódovaná

- Člověk pracuje s desítkovými čísly, kdežto nejpřirozenější vnitřní reprezentace v počítači je dvojková. Z toho vyplývá nutnost převodu čísel v obou směrech. Doba převodu však není zanedbatelná a proto se v počítačích často používá rovněž **aritmetika desítková**, která pracuje s desítkovými číslicemi kódovanými binárně.
- Označení **BCD** je vyhrazené pro jediný kód, přestože toto označení je obecně použitelné pro celou skupinu desítkových dvojkově vyjádřených kódů:

| Číslice | BCD | ASCII | n + 3 | 2 z 5 |
|---------|------|-----------|-------|-------|
| 0 | 0000 | 0011 0000 | 0011 | 11000 |
| 1 | 0001 | 0011 0001 | 0100 | 00011 |
| 2 | 0010 | 0011 0010 | 0101 | 00101 |
| 3 | 0011 | 0011 0011 | 0110 | 00110 |
| 4 | 0100 | 0011 0100 | 0111 | 01001 |
| 5 | 0101 | 0011 0101 | 1000 | 01010 |
| 6 | 0110 | 0011 0110 | 1001 | 01100 |
| 7 | 0111 | 0011 0111 | 1010 | 10001 |
| 8 | 1000 | 0011 1000 | 1011 | 10010 |
| 9 | 1001 | 0011 1001 | 1100 | 10100 |

14

Sčítání v kódu BCD

- Pro návrh desítkové aritmetiky (příslušných obvodů) je třeba zjistit aritmetické vlastnosti uvedených kódů.
- Analýzou sčítání dvou číslic v BCD zjistíme, že je-li binární součet **větší než 9**, je pro návrat do kódu BCD nutná **korekce**, a to přičtení konstanty 6 (binárně 0110).
- Nevýhoda BCD: Neúspornost, složitější HW

$$\begin{array}{r} 2 \quad 0010 \\ +3 \quad 0011 \\ \hline 5 \quad 0101 \end{array}$$

$$\begin{array}{r} 5 \quad 0101 \\ +6 \quad 0110 \\ \hline ? \quad 1011 \end{array}$$

$$\begin{array}{r} +K_1 \quad 0110 = 6 \\ \hline 1 \quad 0001 = 11_{\text{dec}} \end{array}$$

$$\begin{array}{r} 9 \quad 1001 \\ +9 \quad 1001 \\ \hline 1 \quad 0010 = 12 \end{array}$$

$$\begin{array}{r} +K_1 \quad 0110 \\ \hline 1 \quad 1000 = 18_{\text{dec}} \end{array}$$

15

Počítání v kódu 2 z 5

- Kód 2 z 5 je neváhový kód, který kóduje informaci nadbytečným množstvím bitů, je tedy redundantní.
- Redundance se projevuje příznivě schopností kódu detekovat jednobitové chyby (viz dále).
- Jeho aritmetické vlastnosti jsou však natolik nepříznivé, že použití binární sčítačky je prakticky nemožné. Je proto třeba navrhnout speciální sčítačku, pracující v tomto kódu.
- Často se realizuje **tabulkou v paměti**.
- Adresu tvoří všechny kombinace hodnot vstupních operandů a obsah je hodnota výsledku včetně případného přenosu.
- Jaké je využití paměťové kapacity: adresových bitů je 5 + 5 = 10, paměťových míst je tedy $2^{10} = 1024$. Využitých paměťových míst je $10 \times 10 = 100$. Využití paměti je $100:1024 = 9,76\%$.

| Adresa ROM | Data ROM |
|-------------|----------|
| ... | ... |
| 00101 00110 | 01010 |
| A=2 B=3 | C=5 |
| ... | ... |

16

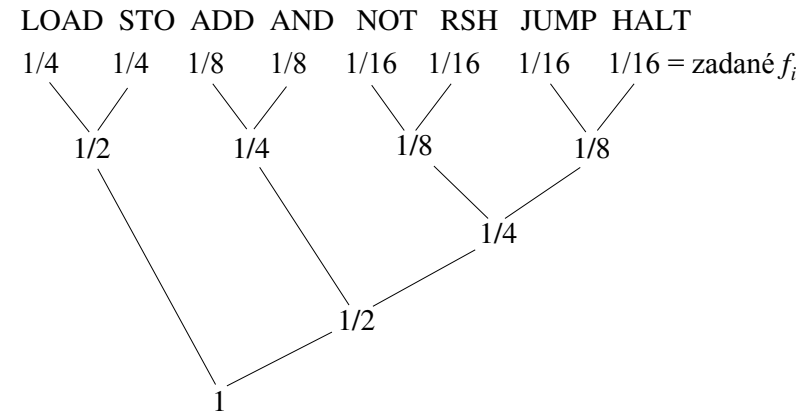
Huffmanův kód

- Jakým způsobem zakódovat znaky abecedy tak, aby **častěji** se vyskytující znaky byly zakódovány pomocí kratší **binární** sekvence?
 - Př. Morseovka, JPEG, ZIP, ...
- Huffmanovo kódování** umožňuje **optimálně** vyřešit tento problém. Vychází ze známých frekvencí jednotlivých kódových značek. Pokud je četnost výskytu značek neznámá, musí se odhadnout.
- Huffmanovo kódování patří mezi **kódy s proměnnou délkou** (VLC – Variable Length Coding)
- Příklad:** Máme zadán hypotetický instrukční soubor a frekvence výskytu jednotlivých instrukcí. Jak zakódovat častěji se vyskytující instrukce kratším kódem a zřídka se vyskytující instrukce delším kódem, abychom ušetřili místo v paměti (na disku) při ukládání programů?

| | frekvence výskytu f_i |
|----------|-------------------------|
| – LOAD | 1/4 |
| – STORE | 1/4 |
| – ADD | 1/8 |
| – AND | 1/8 |
| – NOT | 1/16 |
| – SHIFTR | 1/16 |
| – JUMP | 1/16 |
| – HALT | 1/16 |

17

Př. Huffmanův kód



18

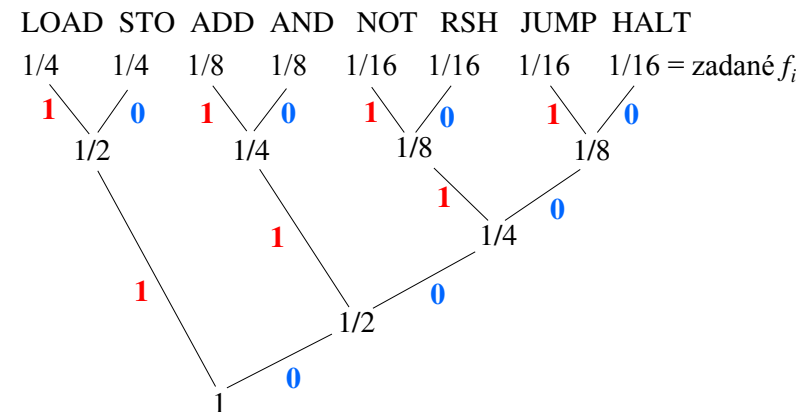
Huffmanův kód – postup

Konstrukce stromu: Najdeme dvojici operačních znaků, jejichž součet pravděpodobností je nejmenší. Tyto dva znaky se nahradí společným uzlem v grafu s pravděpodobností výskytu danou součtem pravděpodobností. Dostali jsme tak skupinu znaků s počtem znaků o jedničku menším než dřív. Na této nové skupině opět hledáme dvojici s nejmenším součtem pravděpodobností. Naznačený postup opakujeme tak dlouho, až spojíme poslední dvojici do jednoho kořenového uzlu s pravděpodobností výskytu rovnou jedné.

Kódování: Vycházíme z kořenového uzlu. Systematicky ohodnotíme hrany stromu (např. hrany vedoucí do uzlu s menším ohodnocením budou 0, jinak 1). Postup opakujeme tak dlouho, až označíme všechny hrany. Kód jednotlivých znaků zjistíme tak, že procházíme pro každý znak celou cestu od kořenového uzlu do příslušného listového uzlu a zaznamenáváme si popis hran, kterými procházíme.

19

Př. Huffmanův kód



20

Př. Huffmanův kód – zakódování

| | kód | délka l_i | |
|----------|------|-------------|--------------------------------|
| • LOAD | 11 | 2 | Huffmanův kód je prefixový. |
| • STORE | 10 | 2 | |
| • ADD | 011 | 3 | |
| • AND | 010 | 3 | |
| • NOT | 0011 | 4 | |
| • SHIFTR | 0010 | 4 | |
| • JUMP | 0001 | 4 | |
| • HALT | 0000 | 4 | |

Příklad 1: Dekódujte posloupnost 00110010111110011111010.

Příklad 2: Kolik bitů by bylo potřeba pro zakódování posloupnosti instrukcí z příkladu 1 pomocí standardního binárního kódování?

21

Parametry kódů s proměnnou délkou

- Střední aritmetická délka (celkem je N značek) [bit]:
$$l_{ar} = \frac{1}{N} \sum_{i=1}^N l_i$$
- Střední dynamická délka [bit]:
$$l_{dyn} = \sum_{i=1}^N l_i f_i$$
- Teoreticky optimální délka [bit]:
$$l_{opt} = - \sum_{i=1}^N f_i \log_2 f_i$$
- Redundance kódu:
$$R = \frac{l_{dyn} - l_{opt}}{l_{dyn}}$$
- Pro náš příklad: $l_{ar} = 3,25; l_{dyn} = l_{opt} = 2,75; R = 0$
V tomto případě je sestrojený kód optimální!

22

FX vs. FP v ASIC

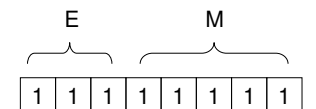
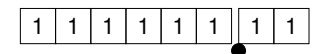
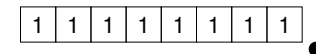
| Multiplier | Operand width | Output width | Area μm^2 | Power mW | Delay ns | PDP 10^{-12} Ws |
|-----------------|---------------|--------------|----------------|----------|----------|-------------------|
| 8 bit unsigned | 8 | 16 | 688 | 0.41 | 1.35 | 0.554 |
| 16 bit unsigned | 16 | 32 | 3020 | 2.68 | 3.14 | 8.415 |
| 32 bit unsigned | 32 | 64 | 10503 | 10.07 | 5.84 | 58.809 |
| 8 bit signed | 8 | 16 | 714 | 0.48 | 1.48 | 0.710 |
| 16 bit signed | 16 | 32 | 2614 | 2.39 | 3.02 | 7.218 |
| 32 bit signed | 32 | 64 | 10357 | 10.05 | 5.55 | 55.778 |
| 32 bit IEEE FP | 32 | 32 | 6787 | 5.96 | 6.12 | 36.475 |

45 nm FreePDK technology; synthesized with Synopsys Design Compiler
PDP is a Power Delay Product.

23

FX vs. FP: Hlavní rozdíly

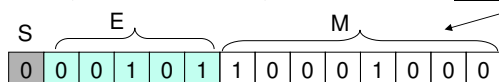
- Pevná řádová čárka – FX
 - bez „řádové čárky“ (8 bitů)
 - Přímý kód: 0 až 255
 - Doplňkový kód: -128 až 127
 - aj.
 - s „řádovou čárkou“ (8 bitů)
 - Př. Přímý kód: 0 až 63,75
 - aj.
 - Čísla jsou na číselné ose rozložena rovnoměrně.
- Pohyblivá řádová čárka – FP
 - $X = (-1)^S M \cdot B^E$
 - M je mantisa,
 - S je znaménko,
 - B je základ,
 - E je exponent
 - Čísla nejsou na číselné ose rozložena rovnoměrně, což umožňuje zvýšit přesnost (více bitů M) nebo rozsah (více bitů E) oproti FX.



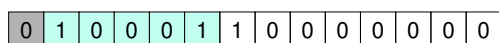
24

Příklad FP na 14 bitech

- E: 5 bitů, M: 8 bitů, S: 1 bit, B = 2
- Interpretace: $y = (-1)^S 0,M \times 2^E$
- Př. $17_{10} = 10001 \times 2^0 = 1000,1 \times 2^1 = 100,01 \times 2^2 = 10,001 \times 2^3 = 1,0001 \times 2^4 = \underline{0,10001 \times 2^5}$



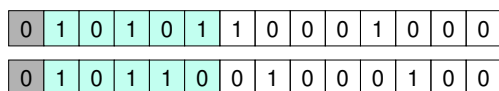
- Př. $65536 = 2^{16} = 0,1 \times 2^{17}$ – se na 14 bitů FX v přímém kódu nevejde, ale v FP to lze



25

Příklad FP na 14 bitech

Problém 2: zobrazení čísel není unikátní



- Př. $17_{10} = 0,10001 \times 2^5 = 0,010001 \times 2^6$
- Unikátnost podpoříme zavedením **normalizované mantisy**
 - nejlevější bit mantisy musí být 1
- Explicitní jednička** – v nejlevějším bitu vždy musí být 1

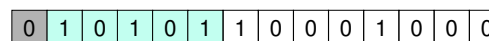
$= 0,1 \times 2^{-4} = 0,03125$
- Implicitní jednička**
 - protože víme, že v nejlevějším bitu mantisy musí být vždy jednička, není nutné ji v mantise reprezentovat, ale stále ji uvažujeme
 - výhoda: získáme jeden bit rozlišení navíc
- Problém: Pokud se zavede normalizace, musí být nula ošetřena zvláštním způsobem

27

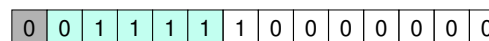
Příklad FP na 14 bitech

Problém 1: malá čísla nelze přesně zobrazit

- Potřebujeme záporný exponent
 - Řešení 1: Přidat znaménkový bit k exponentu – nepoužívá se
 - Řešení 2: Posunout exponent – používá se, je potom jednodušší obvodová realizace porovnání čísel v FP
- Skutečný_exponent = hodnota_pole_exponentu – **BIAS**
 - tj. exponent uložen v kódu s lichým nebo sudým posunutím
- Použijeme **BIAS** = 16 (polovina 2^5)
- Př. $17_{10} = 0,10001 \times 2^5$ (protože $16 + 5 = 21$)



- Př. $0,25_{10} = 0,1 \times 2^{-1}$ ($15 - 16 = -1$)



26

Příklad FP na 14 bitech

Problém 3: chyby zobrazení

- Rozsah zobrazení je $-0,11111111 \times 2^{15}$ až $+0,11111111 \times 2^{15}$.
- Nejmenší kladné číslo (pokud neuvažujeme normalizaci): $0,00000001 \times 2^{-16}$
- tj. například 2^{-39} nebo 2^{128} nelze zobrazit
- Není ale možné ani dostatečně přesně zobrazit např. 128,5.
- $128,5_{10} = 10000000,1$ je na 9 bitů, nejnižší bit se musí zanedbat nebo zaokrouhlit, vzniká chyba: $(128,5 - 128)/128,5 \sim 0,39\%$.
- Chyba se při použití výsledku v dalších operacích zvyšuje a zvyšuje.

28

Standard pro FP: IEEE 754

- Standard **IEEE 754** z roku 1985, poprvé implementován v koprocesorech I 8087.
- IEEE 754-2008 rozšiřuje IEEE 754-1985; převzaly ho také ISO/IEC/IEEE 60559:2011
- Kromě definice B, M, E definuje standard další výjimečné situace.
 - Nečíselný výsledek**, označený zkratkou **NaN** - Not a Number. Tento výsledek se ohlásí např. při výpočtu odmocniny z -1.
 - Definice nekonečna**, které vznikne podílem $1/0$. S tím souvisí definice aritmetiky na nekonečných hodnotách $+\infty$, $1/\infty$, $\arctan(\infty) = \pi/2$, $\arccos(-1) = \pi$.
- decimal – přesně emuluje desítkové zaokrouhlování (účetnictví...)

| IEEE 754-2008 | IEEE 754-1985 | bitů | základ | znaménko | exponent | mantisa | pozn. |
|---------------------------|---------------|------|--------|----------|----------------|----------------------|----------------------------|
| binary16 | - | 16b | 2 | 1b | 5b | 10+1b ^(*) | poloviční přesnost, "Half" |
| binary32 | single | 32b | 2 | 1b | 8b | 23+1b | základní přesnost |
| binary64 | double | 64b | 2 | 1b | 11b | 52+1b | dvojitá přesnost |
| - | extended | 80b | 2 | 1b | ? | ? | dvojitá rozšířená přesnost |
| binary128 | - | 128b | 2 | 1b | 15b | 112+1b | čtyřnásobná přesnost |
| decimal32 ^(x) | - | 32b | 10 | 1b | -95 až +96 | 7 číslic | základní přesnost |
| decimal64 ^(x) | - | 64b | 10 | 1b | -383 až +384 | 16 číslic | dvojitá přesnost |
| decimal128 ^(x) | - | 128b | 10 | 1b | -6143 až +6144 | 34 číslic | čtyřnásobná přesnost |

29

IEEE 754: Vybrané formáty čísel

| | rozsah | přesnost mantisy | zlomková část f | |
|---------------|----------------|------------------|---------------------|------------------|
| krátké reálné | $10^{\pm 38}$ | 24 bitů | S E7...E0 F1...F23 | F0 je implicitní |
| dlouhé reálné | $10^{\pm 308}$ | 53 bitů | S E10...E0 F1...F52 | F0 je implicitní |

Číslo N se získá z hodnoty E uvedené v poli exponentu a z hodnoty z pole mantisy (zlomková část) podle vzorce

$$N = (-1)^S (2^{E-BIAS}) (F0.F1 \dots F23, \text{ nebo } F52), \text{ kde}$$

$$BIAS = 127 \text{ nebo } 1023.$$

Mantisa je vyjádřena přímým kódem se znaménkem, exponent kódem s lichým posunutím. Pozor, v poli exponentu je uvedeno číslo zvětšené o hodnotu $BIAS$. Rozlišujeme tedy pojmy **pole exponentu**, což je posunutý exponent, a **exponent**, resp. neposunutý exponent. Zlomková část f udává číslo menší než 1. Mantisu však získáme součtem $1 + f$, což můžeme zapsat $1.f$.

30

IEEE 754: Příklad

Příklad.

Jaké číslo je zaznamenáno na 32 bitech v jednoduché přesnosti?

1 1000 0001 0100 0000 0000 0000 0000

v poli exponentu je číslo 129

exponent je tedy $129 - 127 = 2$

zlomková část $f = ,01_2 = ,25$

mantisa je tedy 1,25

Jde tedy o číslo $-1,25 \cdot 2^2 = -5$

IEEE 754: Výjimečné hodnoty v „single precision“

Povolené hodnoty exponentu čísel leží v intervalu $\langle -126, +127 \rangle$, po posunu $+127$ získáme povolené hodnoty v poli exponentu $\langle 1, 254 \rangle$. Je-li tedy v poli exponentu 0, nebo 255, jde o hodnoty vyhrazené pro speciální účely:

| pole exponentu | Zlomková část | Význam |
|----------------|---------------|-------------------------------------|
| 255 | 0 | $\pm \infty$ |
| 255 | $\neq 0$ | NaN – je jich mnoho |
| 0 | 0 | 0 |
| 0 | $\neq 0$ | subnormalizované číslo - nenaplnění |

Subnormalizované (denormalizované) číslo: nepočítá se se skrytou 1 a exponent je chápán jako -126.

31

32

IEEE 754: Příklady

| X | Reprezentace X v IEEE 754 – single precision |
|-----------------------|--|
| 1,0 | 0 01111111 000000000000000000000000 |
| 2,0 | 0 10000000 000000000000000000000000 |
| 19,5 | 0 10000011 001110000000000000000000 |
| -3,75 | 1 10000000 111000000000000000000000 |
| 0 (spec.) | 0 00000000 000000000000000000000000 |
| +/- nekonečno | 0/1 11111111 000000000000000000000000 |
| NaN | 0/1 11111111 cokoliv nenulového |
| Denormalizované číslo | 0/1 00000000 cokoliv nenulového |

33

IEEE 754: Rozsah „single precision“

- $MAX = 2^{127} \times 1,111111111111111111111111$
– je to normalizované číslo $\sim 3,4 \times 10^{38}$
- $MIN = 2^{-126} \times 0,000000000000000000000001$
– je to denormalizované číslo $\sim 1,4 \times 10^{-45}$
- Čtyři intervaly nelze reprezentovat
 - Záporná čísla menší než $-MAX$ (negative overflow)
 - Záporná čísla větší než $-MIN$ (negative underflow)
 - Kladná čísla menší $+MIN$ (positive underflow)
 - Kladná čísla větší než $+MAX$ (positive overflow)

34

IEEE 754: Zaokrouhlování

- K zaokrouhlování dochází v případě, že dané číslo nelze přesně vyjádřit.
 - Např. při násobení v desítkové soustavě máme výsledek operace $2,1 \times 0,5 = 1,05$ zaokrouhlit na 1 desetinné místo. Je věcí konvence, zda za výsledek prohlásíme 1,1, nebo 1,0. Oba výsledky jsou zatíženy stejně velkou chybou.
- Norma IEEE zaokrouhluje na číslo, jehož nejnižší číslice je **sudá** (ve dvojkové soustavě). Zaokrouhlovací procedura je definovaná pro 4 případy:
 - Zaokrouhlení k nejbližšímu číslu
 - Zaokrouhlení k nule
 - Zaokrouhlení k $+\infty$
 - Zaokrouhlení k $-\infty$
- Implementace: mimo náplň INP

35

Absolutní chyba zobrazení

- Maximální (absolutní) chyba zobrazení **Err** čísel FP (resp. vzdálenost zobrazitelných bodů) závisí na počtu číslic v mantise a na intervalu definovaném v exponentu:

$$Err = \frac{\text{délka intervalu stupnice pro jistou hodnotu exponentu}}{\text{počet možných číselných kombinací v mantise}} = \frac{\text{základ}^{\text{exponent}} - \text{základ}^{\text{exponent} - 1}}{\text{základ}^{\text{počet číslic v mantise}}}$$

- Příklad: E: 2 bity (v přímém kódu), M: 3 bity (0,M - nenorm.)
- Pro $E = 01$ a $M = 010$ platí $0,010 \times 2^1 = 0,5$
 - $Err = (2^1 - 2^0) / 2^3 = (2 - 1) / 8 = 0,125$
- Pro $E = 10$ a $M = 001$ platí $0,001 \times 2^2 = 0,5$.
 - $Err = (2^2 - 2^1) / 2^3 = (4 - 2) / 8 = 0,25$
- **Zvýšení exponentu o 1 vede na dvojnásobnou chybu Err.**
- **Machine epsilon** je 2^{-b} , kde b je počet bitů mantisy (vč. implicitní 1). Pro $b = 3$ je to $2^{-3} = 0,125$

| | |
|-------------|--------------------|
| $2^1 \cdot$ | .000 = 0,00 |
| | .001 = 0,25 |
| | .010 = 0,50 |
| | .011 = 0,75 |
| | .100 = 1,00 |
| | .101 = 1,25 |
| | .110 = 1,50 |
| | .111 = 1,75 |
| | 1.00 = 2,00 |
| $2^2 \cdot$ | .000 = 0,00 |
| | .001 = 0,50 |
| | .010 = 1,00 |
| | .011 = 1,50 |
| | .100 = 2,00 |
| | .101 = 2,50 |
| | .110 = 3,00 |
| | .111 = 3,50 |
| | 1.00 = 4,00 |

36

Problémy s přesností čísel v FP – př. 1

Dekadické číslo 0,1 není možné přesně reprezentovat ve FP s konečným počtem bitů. Na nekonečném počtu bitů se opakuje 1100:

$E = -4$; $M = 1,110011001100110011001100110011...$,

Po zaokrouhlení na 24 bitů dostáváme:

$E = -4$; $M = 1,110011001100110011001101$

Což je desítkově 0,100000001490116119384765625.

37

Problémy s přesností čísel v FP – př. 2

Uvažme sčítání s 5ti významovými číslicemi v HW (pro jednoduchost v desítkové soustavě)

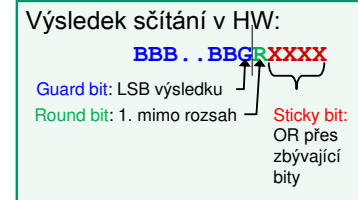
$$\begin{array}{r} 4,5674 \quad .10^0 \\ + 2,5001 \quad .10^{-4} \end{array}$$

po převodu na stejný exponent

$$\begin{array}{r} 4,5674 \quad .10^0 \\ + 0,00025001 \quad .10^0 \\ \hline 4,56765001 \end{array}$$

přesný výsledek

zaokrouhlíme na 4,5677 – to bude výstup ALU



FP ALU používá pro zaokrouhlování kromě LSB výsledku (tzv. Guard bit) dva další bity: Round bit a Sticky bit. Pokud existují nenulové bity za bitem R, podle kterého se zaokrouhluje, nastaví se příznak Sticky bit = 1. Zaokrouhluje se podle trojice GRS. Tento postup vede na snížení počtu bitů sčítačky mantis.

38

Problémy s přesností čísel v FP – př. 3

FP aritmetika nepracuje, jak jsme zvyklí, např. sčítání není asociativní!

Asociativita: $a + (b + c) = (a + b) + c$

$$a = 0.123\,41 \times 10^5 \quad b = -0.123\,40 \times 10^5 \quad c = 0.143\,21 \times 10^1$$

$$\begin{aligned} a +_{fp} (b +_{fp} c) &= 0.123\,41 \times 10^5 +_{fp} (-0.123\,40 \times 10^5 +_{fp} 0.143\,21 \times 10^1) \\ &= 0.123\,41 \times 10^5 -_{fp} 0.123\,39 \times 10^5 \\ &= 0.200\,00 \times 10^1 \end{aligned}$$

$$\begin{aligned} (a +_{fp} b) +_{fp} c &= (0.123\,41 \times 10^5 -_{fp} 0.123\,40 \times 10^5) +_{fp} 0.143\,21 \times 10^1 \\ &= 0.100\,00 \times 10^1 +_{fp} 0.143\,21 \times 10^1 \\ &= 0.243\,21 \times 10^1 \end{aligned}$$

Porovnávání FP čísel: if (x == y) ...

Např. výsledek testu $0.6/0.2-3 == 0$ bude na řadě počítačů FALSE.

V IEEE 754 (double precision) je 0.6/0.2-3 přibližně

-4.44089209850063e-16.

39

Literatura

Drábek, V.: Výstavba počítačů, skripta VUT v Brně, PC-DIR, Brno, 1995

Parhami, B.: Computer Arithmetic: Algorithms and Hardware Designs, Oxford U. Press, 2nd ed., 2010

Null, L., Lobur, J.: The Essentials of Computer Organization and Architecture, 2nd. Ed., Jones and Bartlett Publ., 2006

https://en.wikipedia.org/wiki/Floating-point_arithmetic

HW realizace základních operací v FP bude nastíněna později.

40