Fakulta elektrotechnická
Katedra aplikované elektroniky a telekomunikací

Bakalářská práce

Zaokrouhlovací chyby v číslicových systémech

Autor práce: Tomáš Sak
Vedoucí práce: Ing. Jiří Lahoda, Ph.D.

Plzeň 2012

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

V Plzni dne 10. června 2012

Tomáš Sak

.....

Podpis

Obsah

Seznam obrázků

Seznam tabulek

1

Seznam symbolů a zkratek

BO	Borrow Out. Záporný přenos.
CPU	Central Processing Unit. Procesor/Mikroprocesor.
CY	Carry Out. Kladný přenos.
FP	Floating Point. Pohyblivá řádová čárka.
FPGA	Field Programmable Gate Array. Programovatelná hradlová pole.
FPU	Floating-Point Unit. Matematický koprocesor.
FX/FXP	Fixed Point. Pevná řádová čárka.
GPU	Graphic Processing Unit. Grafický procesor.
LSB	Least Significant Bit. Nejméně významný bit.
MSB	Most Significant Bit. Nejvýznamnější bit.

2

Reprezentace numerických hodnot

V této kapitole se pokusím popsat některé ze způsobů reprezentace numerických hodnot v operační paměti počítače, registrech mikroprocesoru **(CPU)** či v matematickém koprocesoru **(FPU)**. Tyto hodnoty se pro výše uvedené aplikace ukládají v binárním (dvojkovém) formátu. Což znamená, že pro vyjádření takovéto hodnoty využijeme N bitů. Pomocí této N-tice bitů je možné vyjádřit 2^N navzájem odlišných stavů. Jeden bit je základní a zároveň nejmenší jednotkou informace, která se používá především v číslicové technice. Jeden bit reprezentuje informaci, získanou odpovědí na jednu otázku typu ano/ne, u které je předem daná pravděpodobnost obou odpovědí stejná (jinými slovy, u které nemáme žádnou předchozí informaci, která by jednu z odpovědí upřednostňovala). Tyto odpovědi můžeme označit binárními číslicemi 0 a 1. Jako alternativa k značení binárních číslic 0 a 1 jsou například symboly ano/ne *(yes/no)* nebo pravda/lež *(true/false)*.

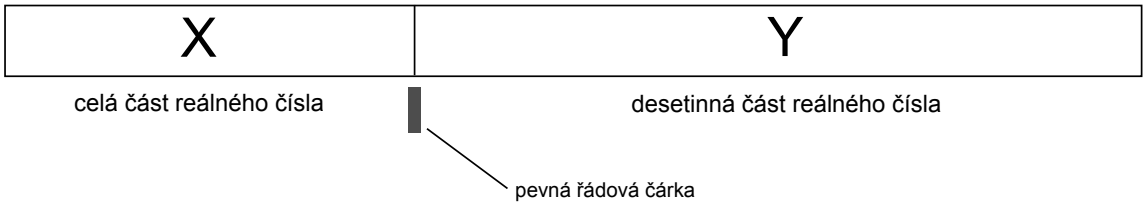
Nejpoužívanější zobrazení binárních stavů je zobrazení na interval celých kladných čísel *(Unsigned Integers)*, popřípadě na interval celých čísel *(Signed Integers)*. Nejčastějšími formami reprezentace číselných hodnot jsou pak formát *(pevné řádové čárky)* a formát *(plovoucí řádové čárky)*. Z anglické literatury jsou zavedeny zkratky **FX** nebo **FXP** *(Fixed Point)* pro formát pevné řadové čárky a **FP** *(Floating Point)* pro formát plovoucí řadové čárky. Princip obou formátů pro ukládání podmnožiny racionálních čísel je odlišný, ať už pro způsob reprezentace binárního čísla, tak pro aritmetické operace prováděné *CPU* nebo *FPU*.

Každá operační paměť počítače má konečný počet bitů a je jen na nás jak je použijeme. To se zdá býti jedním s největších problémů dnešních programátorů, protože nedokáží efektivně využít všechny dostupné bity a některé promrhají ukládáním nepodstatných informací. Většina mikroprocesorů je však navržena na zpracovávání konstantního počtu bitů. Jako příklad může posloužit dnes hojně využívaná řada procesorů *x86*, která je nejlepší pro práci s 32 bity. Avšak při výpočtu s odlišným počtem bitů její efektivita klesá. Řešením tohoto problému je využití programovatelných polích **FPGA**, na kterých je možno vytvořit obvody pracující s libovolným počtem bitů.

2.1 Formát s pevnou řádovou čárkou *(Fixed point)*

Tato podkapitola se bude věnovat popisu méně používaného formátu a to formátu s pevnou řádovou čárkou. Pokusím se objasnit principy uchování dat v tomto formátu, aritmetické operace s těmito čísly a nakonec se pokusím porovnat výhody a nevýhody tohoto formátu oproti formátu s plovoucí řádovou čárkou.

Jak již napovídá jméno formátu, řádová čárka má přesně definovanou, pro všechna čísla neměnnou polohu. Ta poté rozděluje binární slovo na dvě různé části a to na část X, která obsahuje celou část reálného čísla a na část Y, která vyjadřuje desetinnou část reálného čísla (Obr. 3.1).



Obr. 2.1: Rozdělení na desetinnou a celou část binárního slova ve formátu pevné řádové čárky

Pomocí celé části X lze měnit rozsah čísel, protože každý bit této části reprezentuje kladnou mocninu dvojkového základu, naopak pomocí desetinné části Y lze měnit přesnost, protože každý bit této části reprezentuje zápornou část dvojkového základu. Nejdůležitější při zvolení tohoto formátu je předešlé důkladné rozmyšlení potřebné přesnosti a rozsahu. Pokud chce programátor plně využít dostupnou délku slova ve formátu pevné řádové čárky, tak musí udělat jedno z těchto rozhodnutí. Je-li binární slovo moc velké, může posunout všechny bity směrem doprava, kde dojde ke ztrátě nižších bitů. Při posunu o jeden bit doprava ztrácíme nejméně významný bit z angličtiny známy jako **LSB** *(Least Significant Bit)*. Pro opačný případ lze posunout všechny bity směrem doleva, aby nejvyšší bit byl na pozici **MSB** *(Most Significant Bit)*. Pro obě tyto operace je důležité zapamatovat si o kolik bitů došlo k posunu, aby bylo možné obnovit všechna čísla ve stejném rozsahu v pozdějším stádiu.

Přesnost a rozsah hodnot ve formátu pevné řádové čárky, lze reprezentovat například na bitovém slově o délce 8 bitů, kde celá část X zabíra 5 bitů a desetinná část Y pak zbylé 3 bity. Poloha řádové čárky v tomto uvedeném příkladu je tedy mezi 3. a 4. bitem slova (Tab. 3.1).

Číslo bitu	8	7	6	5	4	3	2	1
Váha bitu	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
Dekadická váha bitu	16	8	4	2	1	$0,5$	$0,25$	$0,125$

Tab. 2.1: Příklad rozsahu hodnot ve formátu pevné řádové čárky

2.1.1 Základní matematické operace

Zde bych se chtěl pokusit popsat základní matematické operace používané ve formátu s pevnou řádovou čárkou a to zejména ty nejpoužívanější jako sčítání, odčítání, násobení a dělení.

- Součet a rozdíl dvou hodnot A a B

  Jako první jsem zvolil popis dvou jednodušších operací prováděných ve formátu pevné řádové čárky a to sčítání a odčítání. Záměrně jsem tyto dvě operace spojil do společného textu, neboť pro ně platí stejná pravidla. Nejprve je nutné zjistit, zda se operace neprovádí se speciálními typy čísel jako jsou nekonečna nebo jiné výrazy, které vzniknou například po dělění nulou. Poté je nutné zajistit, aby obě čísla byly ve stejném FX formátu, tedy aby u nich byla stejná velikost jak celé části X, tak i desetinné části Y.

  Po ošetření výše uvedených výjimek můžeme použít následující rovnice. A to pro sčítání (rov. 3.1) a analogicky k tomu pro odčítání (rov. 3.2).

$$A \times 2^Y + B \times 2^Y = (A + B) \times 2^Y \qquad [-] \tag{2.1}$$

$$A \times 2^Y - B \times 2^Y = (A - B) \times 2^Y \qquad [-] \tag{2.2}$$

- Součin dvou hodnot A a B

  Od jednodušších operací přecházíme ke složitějším, konkrétně součinu dvou hodnot. Zde není nutná podmínka rovnosti celé části X a desetinné části Y u obou čísel jako tomu bylo u předešlých operací. Je zde ale jiný problém, pakliže násobíme dvě čísla o N bitech, výsledné číslo pak má velikost 2N bitů (pokud násobíme dvě osmibitová čísla, výsledek je šestnáctibitový). Toto řešení je však značně nevýhodné, neboť budeme potřebovat dvakrát více bitů.

  Řešením je zmenšení desetinné oblasti Y u obou čísel o polovinu a posun o tuto zmenšenou oblast směrem doprava. Přebývající bity v celé části X se pak ořízno na požadovanou velikost formátu. To má však logicky za následek zmenšení přesnosti jak obou počátečních hodnot A a B, tak i výsledku vzniklého násobením takto upravených čísel. Dalším problémem se kterým se budeme při této operaci často potýkat a na který si musíme dát obzvlášť velký pozor je **přetečení** a **podtečení**.

  Při součinu pak rozlišujeme jestli se jedná o formát se znaménkem *(signed)*(rov. 3.3) nebo bez *(unsigned)*(rov. 3.4). Vidíme, že rovnice se liší pouze v přičtení jednotky k celé části X, právě ta nám určuje znaménko.

$$A(X_1, Y_1) \times B(X_2, Y_2) = C(X_1 + X_2 + 1, Y_1 + Y_2) \qquad [-] \tag{2.3}$$

$$A(X_1, Y_1) \times B(X_2, Y_2) = C(X_1 + X_2, Y_1 + Y_2) \qquad [-] \qquad (2.4)$$

- Podíl dvou hodnot A a B

  Stejně jako součet a rozdíl mají podobné vlastnosti i součin s podílem. Není tedy vyžadována stejná velikost celé X a desetinné Y části. Obdobný je i problém s velikostí výsledku, kde se dostaneme k číslu s větším počtem bitů, než měli hodnoty počáteční (A a B). Řešením je opět zmenšení desetinné části Y a posun směrem doprava. Logicky stejně dochází i ke zmenšení přesnosti výsledku. Také u této operace rozlišujeme zdali se jedná o znaménkový formát *(signed)*(rov. 3.5) či formát bez znaménka *(unsigned)*(rov. 3.6).

$$\frac{A(X_1, Y_1)}{B(X_2, Y_2)} = C(X_1 + Y_1, |\log_2(2^{X_2+Y_1} - 2^{Y_1-Y_2})|) \qquad [-] \qquad (2.5)$$

$$\frac{A(X_1, Y_1)}{B(X_2, Y_2)} = C(X_1 + Y_2 + 1, X_2 + Y_1) \qquad [-] \qquad (2.6)$$

2.2 Formát s plovoucí řádovou čárkou *(Floating point)*

Jako druhý se pokusím představit nejpoužívanější formát v dnešní době a to formát s plovoucí řádovou čárkou (FP). Jak již název napovídá největší odlišností od formátu s pevnou řádovou čárkou (FX), je to, že řádová čárka není pevně určená a může se měnit její poloha. Z tohoto vyplývá, že si každá hodnota nese polohu řádové čárky v sobě.

Formát se skládá z dvou částí a to **mantisy** a **exponentu**, obě tyto části mohou být jak kladné tak záporné. Mantisa má v sobě obsaženou informaci o významných číslicích numerické metody, naopak exponent udává mocninu o určitém základu (nejčastěji 2,8,10 a 16), kterou jsou významné číslice v mantise děleny či násobeny. Původní hodnotu reálného čísla dopočítáme podle (rov. 3.7).

$$R = b^e \times m \qquad [-] \tag{2.7}$$

kde:

R - původní hodnota reálného čísla
b - báze, neboli radix
e - exponent
m - mantisa

Různé formáty hodnot, které reprezentují čísla ve formátu plovoucí řádové čárky *FP* se od sebe liší volbou báze **(radixu)**, ale i počtem bitů vyhrazených pro uložení mantisy či exponentu. Přehled těchto formátu specifikuje mezinárodní norma IEEE 754. Tato norma je v platnosti od roku 1985 a kromě specifikací uložení numerických hodnot obsahuje také způsoby provádění základních matematických operací, pravidla pro konverzi mezi jednotlivými formáty a pravidla pro práci se specifickými hodnotami jako je nekonečno.

2.2.1 Norma IEEE 754

V následujícím textu se pokusím objasnit nejčastěji používané formáty pro uložení numerických hodnot ve formátu s plovoucí řádovou čárkou *FP*. Základní popis podmnožiny racionálních čísel může, vyjadřovat vztah (rov. 3.8).

$$R = (-1)^s \times 2^{e-bias} \times m \qquad [-] \tag{2.8}$$

kde:

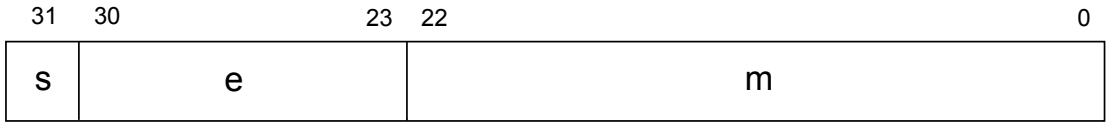
R - numerická hodnota racionálního čísla, umožňuje rozlišení speciálních typu čísel, jako nuly a nekonečna na kladné a záporné.
2 - báze, neboli radix, u normy IEEE 754 je báze vždy dvojka, pro zjednodušení výpočtu u číslicových obvodů
e - exponent, kladný posunutý o hodnotu **bias**
bias - taková hodnota, aby byla zaručena kladnost exponentu
m - mantisa, vždy kladná
s - bit určující znaménko, pokud se rovná nule, je výsledná hodnota kladná a naopak

Podle těchto parametrů rozlišuje norma IEEE 754 na dva základní formáty a to na formát jednoduché přesnosti *(single)* a na formát s dvojitou přesností *(double)*.

- Formát jednoduché přesnosti *(single)*

  Tento formát je možná známější pod názvem **float**, což zapříčiňuje hojné rozšíření programovacích jazyků Java a C++. Charakteristickým znakem tohoto formátu je uložení hodnoty na 32 bitech. Ty jsou dále rozloženy na 3 části. Znaménko je uloženo

na nejvyšším bitu *(MSB)*, dalších osm bitů napravo od *MSB* je vyhrazeno pro uložení posunutého exponentu a zbylých 23 bitů je využito pro uložení mantisy (Obr. 3.2). Velikost 32 bitů je zvolena pro svůj dobrý poměr mezi přesností a rozsahem hodnot, nehledě na to, že má optimální nároky na úložný prostor a většina v dnešní době používaných architektur má právě 32-bitovou sběrnici. Proto se s tímto formátem můžeme setkat u většiny **FPU** a **GPU**. Hodnota *bias* je v tomto formátu rovna 127 (rov. 3.9).



Obr. 2.2: Struktura formátu s plovoucí řádovou čárkou, jednoduchá přesnost

$$\begin{aligned} bias &= 2^{eb-1} - 1 \\ &= 2^{8-1} - 1 \\ &= 128 - 1 \\ &= 127 \qquad [-] \end{aligned} \tag{2.9}$$

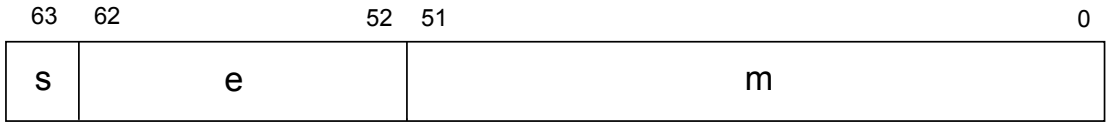
kde: *eb* - počet bitů určených pro exponent

Vztah vyjadřující popis podmnožiny racionálních čísel ve formátu jednoduché přesnosti pak můžeme popsat jako (rov. 3.10).

$$R = (-1)^s \times 2^{e-127} \times m \qquad [-] \tag{2.10}$$

- Formát dvojité přesnosti *(double)*

  Druhým ze základních formátů, které specifikuje norma IEEE 754 je formát s dvojitou přesností **(double)**. Jak již název napovídá, počet bitů do kterých je možné hodnotu uložit se zdvojnásobil. Takže tento formát operuje s 64 bity, narozdíl od 32 bitů formátu **single**. Bity jsou pak obdobně rozloženy na tři části. *MSB* je opět vyhrazen pro znaménko, dalších 11 bitů pro exponent a zbylých 52 pro uložení mantisy (Obr. 3.3). Hodnota *bias*, která určuje o kolik je exponent posunutý je v tomto případě 1023 (rov. 3.11).



Obr. 2.3: Struktura formátu s plovoucí řádovou čárkou, dvojitá přesnost

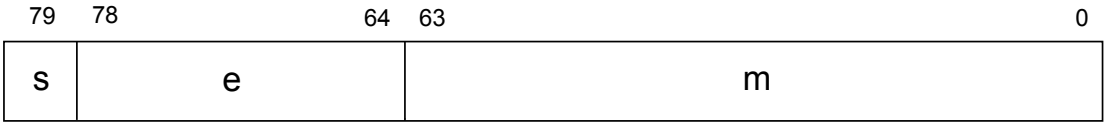
$$\begin{aligned} bias &= 2^{eb-1} - 1 \\ &= 2^{11-1} - 1 \\ &= 1024 - 1 \\ &= 1023 \qquad [-] \end{aligned} \tag{2.11}$$

Poté vztah, popisující podmnožinu racionálních čísel u formátu dvojité přesnosti je následující (rov. 3.12).

$$R = (-1)^s \times 2^{e-1023} \times m \qquad [-] \tag{2.12}$$

- Rozšířený formát *(extended)*

  Pro doplnění základních formátu *(single a double)* z normy IEEE 754, představím ještě rozšířeny formát *(extended)*, který norma také definuje. Ze všech tří formátu má tento největší kapacitu pro práci s hodnotami a to celých 80 bitů. Bity jsou opět analogicky rozloženy na tři části. *MSB* pro znaménko, 15 bitů pro exponent a 64 bitů pro mantisu (Obr. 3.4). Tento formát je spolehlivější a přesnější, pro svoji minimalizaci *přetečení* a zaokrouhlovacích chyb. Jeho hlavní určení je pro výpočty na *FPU*. Hodnotu *bias* spočteme podle již známého vzorečku (rov. 3.13) a vyjde nám 16383.



Obr. 2.4: Struktura formátu s plovoucí řádovou čárkou, rozšířená přesnost

$$\begin{aligned} bias &= 2^{eb-1} - 1 \\ &= 2^{15-1} - 1 \\ &= 16384 - 1 \\ &= 16383 \qquad [-] \end{aligned} \tag{2.13}$$

Obdobně jako v předešlých případech je vztah pro popis podmnožiny racionálních čísel vyjádřen (rov. 3.14).

$$R = (-1)^s \times 2^{e-16383} \times m \qquad [-] \tag{2.14}$$

Na konec této kapitoly bych se ještě rád zmínil o dalších formátech popsaných v normě IEEE 754. Těmito jsou například formát **ZX Spectrum**, jež byl používám na stejnojmenném osmibitovém počítači. Tento velmi populární počítač vyrobený v roce 1982, jsme v České republice znali spíše pod názvem jednoho z jeho klonů - Didaktik.

Dalším formátem stvořeným firmou Borland pro programovací jazyk Turbo Pascal byl šestnáctibitový formát **real**. Jiným šestnáctibitovým formátem je **Minifloat**, používaný například na grafických procesorech firmy nVidia. Posledním a nejmenším formátem pro praktické využití je formát **Microfloat**, u kterého jsou hodnoty uloženy na pouhých osmi bitech.

2.2.2 Základní matematické operace

Stejně jako u formátu s pevnou řádovou čárkou, i zde se pokusím vypracovat přehled základních aritmetických operací a jejich realizaci u formátu s plovoucí řádovou čárkou.

- Součet a rozdíl dvou hodnot A a B

  Obdobně jako u formátu s pevnou řádovou čárkou začnu s popisem dvou jednodušších operací a to sčítáním a odčítáním. Záměrně jsem tyto dvě operace spojil do společného textu, neboť pro ně platí stejná pravidla. Opět je nejprve nutné zjistit, zda se operace neprovádí se speciálními typy čísel jako jsou nekonečna nebo jiné výrazy, které vzniknou například po dělění nulou. Poté je potřeba zjistit, které ze sčítaných hodnot A a B je větší. To se provede porovnáním hodnot exponentů u obou čísel. Pokud se exponenty nerovnají, je nutné vypočítat číselný rozdíl exponentů (rov. 3.15). Mantisu menšího z obou porovnávaných čísel pak přesuneme doprava právě o tolik bitů, kolik jsme vypočítali rozdílem obou exponentů. Poté je nutné zjistit hodnoty bitů určujících znaménka obou čísel a podle nich je rozhodnuto zda půjde o operaci sčítání či odčítání. Výsledek této operace má pak znaménkový bit rovný znaménkovému bitu většího z čísel (tzn. pokud bude větší číslo kladné, výsledek bude kladný a naopak).

  Rizikem těchto operaci je *přetečení* či *podtečení* výsledku. K tomuto dochází při dosažení maximální respektive minimální hodnoty exponentem. Při přetečení dochází u sčítačky ke generování impulsu **CY** *(Carry Out)* a výsledek je pak vrácen jako nekonečno, o znaménku pak rozhoduje samostatný výpočet. Naopak při podtečení dochází k vygenerování impulsu **BO** *(Borrow Out)* a výsledkem je pak kladná nebo záporná nula.

$$difference = e_1 - e_2 \qquad [-] \tag{2.15}$$

  Pro názorné předvedení operace součtu u formátu s plovoucí řádovou čárkou, poslouží pár příkladů. Jako první příklad jsem zvolil součet 2 hodnot na kterém vidíme, že poslední tři hodnoty čísla B jsou nenávratně ztraceny, díky závěrečnému zaokrouhlení. (rov. 3.16).

Př.1 A = 538623,6 ; B = 105,4239

$$\begin{aligned} 538623,6 + 105,4239 &= (5,386236 \times 10^5) + (1,054239 \times 10^2) \\ &= (5,386236 \times 10^5) + (0,001054239 \times 10^5) \\ &= (5,386236 + 0,001054239) \times 10^5 \\ &= 5,387290239 \times 10^5 \\ &= 5,387290 \times 10^5 \qquad [-] \end{aligned} \tag{2.16}$$

Extrémním případem však může být, že menší z čísel nemá kvůli konečnému zaokrouhlení na součet vůbec žádný vliv. Což demonstruje druhý příklad (rov. 3.17).

Př.2 A = 538623,6 ; B = 0,0001054239

$$\begin{aligned} 538623,6 + 0,0001054239 &= (5,386236 \times 10^5) + (0,00000001054239 \times 10^5) \\ &= (5,386236 \times 10^5 + 0,00000001054239) \times 10^5 \\ &= 5,38623601054239 \times 10^5 \\ &= 5,386236 \times 10^5 \qquad [-] \end{aligned} \tag{2.17}$$

- Součin dvou hodnot A a B

  Další a značně složitější aritmetickou operací je součin. Je komplikovanější z toho důvodu, že jsou hodnoty rozděleny na mantisu a exponent. NA začátku této operace budeme postupovat stejně jako sčítání či odčítání. Musíme zkontrolovat zda nebudeme operovat se speciálními hodnotami jako jsou nekonečna a tzv. **NaN** (Not a Number), které vznikají například po dělení nuly nulou. Výsledný exponent poté získáme tak, že sečteme exponenty obou hodnot A i B (rov. 3.18). Takto vzniklý výsledek pak ještě musíme ponížit o hodnotu **bias**, z důvodu zabránění posunu výsledku směrem doleva, který by mohl vzniknout kvůli umístění binární čárky v mantise za prvním bitem. Pokud dojde při součtu exponentů k přetečení (překročí-li se maximální hodnota) je výsledkem nekonečno. O znaménku pak rozhoduje znaménkový bit. Naopak jestliže dojde při součtu exponentů k podtečení (překročí-li se minimální hodnota) výsledkem je nula a o znaménku opět rozhoduje znaménkový bit. Výslednou mantisu pak spočítáme jako součin mantis hodnot A a B (rov. 3.19). Pokud dojde o přetečení výsledné mantisy o určitou hodnotu, je právě o tuto hodnotu exponent navýšen. Pokud se *MSB* mantisy nerovná jedné, posouvají se bity směrem doleva a exponent naopak klesá. Nakonec spočítáme výsledný znaménkový bit pomocí bitové operace nonekvivalence **XOR** obou znaménkových bitů hodnot A a B (rov. 3.20).

$$e = e_1 + e_2 \qquad [-] \tag{2.18}$$

$$m = m_1 \times m_2 \qquad [-] \tag{2.19}$$

$$s = s_1 \oplus s_2 \qquad [-] \tag{2.20}$$

Podle výše uvedených skutečností je tedy součin definován pro hodnotu A jako (rov. 3.21) a pro hodnotu B (rov. 3.22).

$$A = (-1)^{s1} \times 2^{e_1} \times m_1 \qquad [-] \tag{2.21}$$

$$B = (-1)^{s2} \times 2^{e_2} \times m_2 \qquad [-] \tag{2.22}$$

Výsledný součin těchto hodnot pak můžeme vyjádřit jako (rov. 3.23).

$$A \times B = (-1)^{s1} \times (-1)^{s2} \times 2^{e_1+e_2} \times m_1 \times m_2 \qquad [-] \tag{2.23}$$

I u této operace musíme brát v potaz rizika s ní spjatá. A to hlavně riziko přetečení, podtečení a problémy spjaté se zaokrouhlováním hodnot.

- Podíl dvou hodnot A a B

  Poslední ze základních aritmetických operací je dělení. U této operace si musíme dávat obzvlášť velký pozor na situace, kdy budeme dělit nulou. Při tomto úkonu musí být zachováno znaménko, aby bylo možné detekovat, zda-li je výsledek kladné nebo záporné nekonečno. Nepovolené je samozřejmě dělení nuly nulou, kdy se rovná návratová hodnota *NaN*. Samotná operace dělění lze provést třemi různými způsoby. Za prvé můžeme využít hardwarové děličky, kterou hojně využívají v dnešní době *FPU*. Druhým způsobem může být postupné odečítání dělence od dělitele, postup s kterým jsme se seznámili na základní škole, při jednoduchých výpočtech. A do třetice postup, který využívá k výpočtu převrácené hodnoty dělitele, s kterou je následně vynásoben dělenec. Samozřejmě i zde platí všechna předešlá pravidla, pro počítání se speciálními čísly a rizika s nimi spjatá. Podíl dvou hodnot A a B u formátu s plovoucí řádovou čárkou je tedy definován jako (rov. 3.24). Vidíme zřetelnou analogii s předešlým vztahem pro součin.

$$\frac{A}{B} = (-1)^{s1} \times (-1)^{s2} \times 2^{e_1-e_2} \times \frac{m_1}{m_2} \qquad [-] \tag{2.24}$$

2.3 Porovnání FX a FP

Jak již vyplývá z předchozího textu, používanějším z obou formátu je Floating point. Jednou z hlavních předností tohoto formátu je podpora FP operací na hardwarových jednotkách FPU. Až už ve formě samostatného matematického koprocesoru nebo jako přímá součást modernějších mikroprocesorů. Další z nezanedbatelných výhod je určitě samotná existence normy IEEE 754, jež jasně definuje pravidla pro používání formátu s plovoucí řádovou čárkou. Tyto skutečnosti pak logicky vedly k tomu, že tento formát je primárně implementován jako datový typ v drtivé většině dnes používaných programovacích jazyků. Toto je obrovský rozdíl oproti formátu s pevnou řádovou čárkou, který je podporován pouze malým množstvím programovacích jazyků.

Kde však FX formát před svým používanějším protějškem vyhrává na celé čáře, jsou situace, kdy dopředu známe rozsah zpracovávaných hodnot a také požadovanou přesnost. Další nespornou výhodou je předem známá pozice řádové čárky, není totiž nutné společně s číselnou hodnotou uchovávat i pozici řádové čárky, což vede ke značné úspoře použitých bitů. Matematické operace prováděné v FX formátu, jsou mnohdy rychlejší a jednodušší, oproti počítání jednoduchých aritmetických operací ve formátu FP bez použití matematického koprocesoru. Asi největší výhodou formátu s pevnou řádovou čárkou je dodržení požadované přesnosti u všech prováděných operací. Jak jsem naznačil dříve, Součet nebo rozdíl dvou od sebe se lišících hodnot v mnoha řádech, může skončit naprostou ignorací menšího z čísel, nebo ještě hůře zacyklením výpočtů v nekonečných smyčkách.

Nevýhodou FX formátu jsou logicky případy, kde předem nevíme s jakými hodnotami budeme pracovat. Pokud bychom zpracovávali hodnoty s velkým poměrem mezi nejvyšší a nejnižší absolutní hodnotou, musíme počítat s alokováním velkého počtu bitů.

Nevýhody FP formátu můžeme hledat paradoxně v komplexnosti tohoto formátu. Neboť rozložení na exponent a mantisu nepřináší jenom výhody. Kvůli tomuto rozdělení je ztrátový i jednoduchý převod mezi formátem **int** a **single**, protože dochází k nenávratnému ztracení hodnoty na osmi nejnižších bitech, které slouží pro uložení exponentu. Proto jsou například signálové procesory vyráběny pouze s podporou FX formátu, protože u nich víme, že na vstupu i výstupu jsou prakticky vždy celá čísla. Další nevýhodou, jež vyplývá z komplexnosti formátu s plovoucí řádovou čárkou, je prakticky nemožné využití v tzv. **embedded** zařízeních, jež jsou v dnešní době využívány vice než klasické osobní počítače, neboť jak jsem psal již dříve matematické koprocesory jsou velmi komplikované.