۳.۱ استاندارد IEEE برای حساب ممیز شناور

اولین کامپیوتر بنام Z3 در سال ۱۹۴۱ توسط کنراد تسوزه [۱۷] در برلین ساخته شد. Z3 یک کامپیوتر الکترومکانیکی بود که دستگاه اعدادش دودویی بود، در سال ۱۹۴۳ در جریان بمباران‌های جنگ جهانی دوم نابود شد. گفته می‌شود اولین کامپیوتر کاملا الکترونیکی اِنیاک [۱۸] است که در سال ۱۹۴۵ در دانشگاه پنسیلوانیا ساخته شد و دارای دستگاه اعداد ده‌دهی بود.

اولین استفاده‌هایی که از کامپیوتر در دهه‌ی ۱۹۵۰ می‌شد، محاسبات عددی در کاربردهای علمی بود اما در دهه‌ی ۱۹۶۰ استفاده‌ی اصلی آن‌ها در تجارت بود و نه برای محاسبات عددی. امروزه بیشتر کاربران از کامپیوترها برای پردازش اطلاعاتی چون متن، تصویر و فیلم، فایل‌های صوتی و سایر انواع اطلاعات استفاده می‌کنند بدون آن‌که اطلاع داشته باشند که پردازش چنین اطلاعاتی نیاز به حجم بزرگی از محاسبات عددی دارد.

درابتدای عصر محاسبات، حساب ممیز شناور در هر کامپیوتر بصورت خاصِ همان کامپیوتر پیاده‌سازی شده بود . در نتیجه حاصل هر محاسبه بستگی به نوع خاص کامپیوتر در حال استفاده داشت و همچنین امکان انتقال برنامه‌های نوشته‌شده در یک کامپیوتر به کامپیوترهای دیگر بسیار محدود بود یعنی برنامه ای که بسیار خوب در یک کامپیوتر اجرا می‌شد ، می‌توانست در کامپیوتر دیگری غیر قابل‌اجرا باشد. درآن دوره هم پیاده‌سازی حساب ممیزشناور در کامپیوترها متفاوت بود یعنی مثلا تعداد بیت‌های اختصاص داده‌شده به مانتیس و نما در پیاده‌سازی‌های مختلف حساب ممیز شناور تفاوت داشت، علاوه بر اینکه کامپیوترهایی بودند که مبنای ۲ ،۱۰ و یا حتی ۱۶ را استفاده می‌کردند.) جدول زیر را ببینید.

کامپیوتر	β	p	$U=-L$
IBM 7090	2	27	2^7
$Borroughs$ 5000 $Series$	8	13	2^6
IBM 360/370	16	6	2^6
DEC 11/780 VAX	2	24	2^7
$Hewlett\ Packard$ 67	10	10	99

درسال ۱۹۸۵ در نتیجه‌ی همکاری دانشمندان علوم کامپیوتر از دانشگاه‌ها و متخصصان سخت‌افزار

[۱۷] Konrad Zuse

[۱۸] ENIAC: Electronic Numerical Integrator and Computer

از صنعت، یک استاندارد برای نمایش اعداد ممیزشناور دودویی و حساب ممیزشناور آن‌ها بوجود آمد. این استاندارد که IEEE p754 نام دارد با حمایت انجمن مهندسین برق و الکترونیک که ((آی تریپل ای)) خوانده می‌شود، ارائه شد. سرپرستی دانشمندان دانشگاهی علوم کامپیوتر به عهده‌ی ویلیام کاهان [۱۹] استاد دانشگاه کالیفرنیا در برکلی بود و کارخانجات کامپیوتری همچون Apple ، Intel ، Hp ، Motorola نیز درگیر ایجاد این استاندارد بودند. این استاندارد تاثیر شگرفی بر صنعت کامپیوتر داشته و ویلیام کاهان در سال ۱۹۸۹ بخاطر تلاشهایش در سرپرستی استاندارد جایزه‌ی معتبر تورینگ را که از انجمن ماشین‌آلات محاسباتی [۲۰] دریافت کرد. در سال ۱۹۸۵، استاندارد دیگری بنام IEEE p854 برای هر دو مبنای ۲ و ۱۰ ابداع شد.

استاندارد IEEE دو نوع پایه‌ای از اعداد ممیزشناور را معین می‌کند: یگانه [۲۱] و دوگانه [۲۲]. (نوع دیگری بنام چهارگانه [۲۳] نیز در استاندارد معرفی شده که در مورد آنها بحث نخواهیم کرد). در استاندارد ، تعدادی نماد خاص معرفی شده‌اند. برخی از آن‌ها عبارتنداز `NaN`[۲۴]، `-Inf` و `+Inf`. دو نماد `-Inf` و `+Inf` مقادیر مشابه با مفاهیم ریاضی $\pm\infty$ بوده و نماد `NaN` برای مواقعی همچون `Inf - Inf` که حاصل یک عمل ممیزشناور تعریف نشده باشد، پیش‌بینی شده است. کمیت دیگری که بصورت خاص در استانداردمشخص می شود، عدد صفر است. استاندارد `IEEE` چگونگی انجام اعمال حساب ممیزشناور در سبک‌های مختلف گردکردن را نیز مشخص می‌کند.

ساختار کلی هر دو نوع پایه‌ای یگانه و دوگانه در مبنای دو یکی بوده و تنها در تعداد بیت‌های اختصاص‌یافته به مانتیس و همچنین تعداد بیت‌های اختصاص‌یافته به توان متفاوتند. در قالب یگانه کلا ۳۲ بیت داریم که اولی به علامت عدد [۲۵]، هشت بیت بعدی به توان و نهایتا ۲۳ بیت پایانی به مانتیس اختصاص یافته‌اند. پس به کمک ایده بیت پنهان در این قالب دقت برابر است با $p = 23+1$ و در نتیجه اپسیلون ماشین برابر است با

$$\varepsilon_M = 2^{-(p-1)} = 2^{-23} \approx 1.2 \times 10^{-7}.$$

استاندارد مملو از جزییات دیگری است که به آنها نپرداخته‌ایم از جمله اینکه به منظور ذخیره‌ی

[۱۹] William Kahan
[۲۰] Association of Computing Machinery (ACM)
[۲۱] single
[۲۲] double
[۲۳] quad
[۲۴] Not a Number
[۲۵] sign bit

توان اعداد نرمال در هر دو قالب یگانه و دوگانه از ایده‌ای به نام توان اریب استفاده می‌شود تا جلوی اختصاص یکی از بیت‌های توان به علامت توان گرفته شود. محدوده توان دودویی اعداد در قالب یگانه بین -126 و $+127$ است. کوچک‌ترین عدد نرمال مثبت در این قالب عبارت است از

$$N_{\min} \approx 1.2 \times 10^{-38}$$

و بزرگترین عدد نرمال مثبت در قالب یگانه برابر است با:

$$N_{\max} \approx 1.7 \times 10^{+38}.$$

به قالب دوگانه (که پیش‌فرض در بسیاری از نرم افزارهای محاسبات علمی همچون متلب است) دو برابر کل قالب یگانه بیت اختصاص داده شده یعنی ۶۴ بیت که اولی برای ذخیره‌سازی علامت عدد، ۱۱ بیت بعدی برای توان و نهایتا ۵۲ بیت پایانی ویژه‌ی مانتیس هستند. با توجه به ایده‌ی بیت پنهان داریم $p = 52 + 1$ و در نتیجه

$$\varepsilon_M = 2^{-(p-1)} = 2^{-52} \approx 2.2 \times 10^{-16}.$$

کوچک‌ترین و بزرگ‌ترین اعداد نرمال مثبت عبارتند از:

$$N_{\min} \approx 2.2 \times 10^{-308}$$

و بزرگترین عدد نرمال مثبت در قالب یگانه برابر است با:

$$N_{\max} \approx 1.8 \times 10^{+308}.$$

سه عدد آخر در نرم افزار متلب با دستورهای `eps`، `realmin` و `realmax` مشخص شده‌اند.

همانگونه که دیدیم هر چه از صفر دورتر می‌شویم فاصله‌ی بین اعداد ممیزشناور نیز بیشتر می‌شود. در مورد استاندارد IEEE می‌توان این موضوع را در متلب به کمک دستور `eps(x)` که همان `ulp(x)` است و فاصله‌ی بین عدد x و نزدیک‌ترین عدد مجاور x در دستگاه اعداد ممیزشناور را مشخص می‌کند مشاهده کرد:

```
>> eps

ans =

   2.2205e-16

>> eps(1)

ans =

   2.2205e-16

>> eps(10)

ans =

   1.7764e-15

>> eps(100000)

ans =

   1.4552e-11

>> eps(realmin)

ans =

  4.9407e-324

>> eps(realmax)

ans =

  1.9959e+292
```

این تفاوت‌ها بسیار چشم‌گیر است . با این حال فاصله‌ی نسبی اعداد ممیزشناور در سرتاسر خط اعداد ماشین تقریبا یکنواخت است:

```
>> eps(realmin)/realmin

ans =

   2.2205e-16
```

```
>> x = 10000; eps(x)/x
ans =
   1.8190e-16
>> eps(realmax)/realmax
ans =
   1.1103e-16
```

می‌توان دید که فاصله‌ی نسبی بین اعداد ماشین تقریبا برابراست با اپسیلون ماشین.

۱.۳.۱ حساب ممیز شناور

مجموعه‌ی اعداد حقیقی تحت هرچهار عمل اصلی بسته است اما یکی از مهم‌ترین مشکلات حساب ممیز شناور بسته نبودن مجموعه‌ی اعداد ماشین تحت عملیات حسابی است. یعنی اگر $x, y \in F_{\beta,p}^{L,U}$ و $* \in \{+, -, \times, /\}$ آنگاه ممکن است $x * y \notin F_{\beta,p}^{L,U}$. بعنوان مثال تقسیم یک بر سه را در نظر بگیرید که حاصل آن، در مبنای دو یا ده، دارای بینهایت رقم بوده و در نتیجه قابل نمایش به صورت دقیق در دستگاهی همچون $F_{2,p}^{L,U}$ نمی‌باشد.

پس راهی که برای انجام محاسبات روی مجموعه‌ی اعداد ماشین به ذهن می‌رسد، این است که جوابِ عمل ممیزشناور را که به طور کلی عددی در $\mathbb{R}$ است، گردکرده و با عددی متعلق به $F_{\beta,p}^{L,U}$ تقریب بزنیم. فرض کنید $* \in \{+, -, \times, /\}$ یکی از چهار عمل اصلی ریاضی بوده و $\circledast \in \{\oplus, \ominus, \otimes, \oslash\}$ عمل متناظر با $*$ در $F_{\beta,p}^{L,U}$ باشد. خوشبختانه استاندارد آی-تریپل-ای برای حساب ممیزشناور تضمین زیر را که به **خاصیت بیشترین کیفیت** معروف است برای چهار عمل اصلی ارائه می‌کند:

$$x, y \in F_{\beta,p}^{L,U}, \quad * \in \{+, -, \times, /\} \implies x \circledast y = fl(x * y).$$

به بیان دیگر حاصل عمل ممیز شناور با دقت متناهی $x \circledast y$ با سناریویی که ابتدا $x * y$ به صورت ریاضی با دقت بینهایت محاسبه شده و پاسخِ کاملا درستِ ریاضی آن که عددی در $\mathbb{R}$ است تنها یک‌بار به عدد ممیزشناورِ همسایه گرد شود مطابقت خواهد داشت.

پس تنها خطایی که در هر تک مرحله از اجرای یکی از چهارعمل اصلی در حساب ممیز شناور با دقت متناهی وجود خواهد داشت همان خطای پایانی گردکردن در $F_{\beta,p}^{L,U}$ می‌باشد. نتیجه‌ی مهم برای ما این است که لزومی ندارد جزییات الگوریتم‌های پیاده‌سازی عملیات ممیزشناور پایه‌ای در مبنای دو را (که بیشتر مبحثی علوم کامپیوتری است تا ریاضی) بدانیم تا بتوانیم در مورد میزان درستیِ جوابی که از آنها می‌گیریم اظهارنظر کنیم! مثلا لازم نیست نگران باشیم که اگر در یکی از مراحل میانی یک محاسبه‌ی عددی زیرنرمال ظهور کرد چه برخوردی باید با آن کنیم و یا اینکه لازم نیست نگران باشیم که از کاربرد بیت نگهبان یا بیت گردکردن یا بیت چسبناک (که در پیاده‌سازی چهار عمل اصلی استفاده می‌شوند) مطلع نیستیم.

قضیه‌ی زیر نتیجه‌ی مستقیم قضیه‌ی ۱.۳.۱ و خاصیت بیشترین کیفیت حساب ممیز شناور است.

قضیه ۱.۳.۱. *فرض کنید x و y دو عدد ممیزشناور نرمال باشند. در این‌صورت میزان خطای نسبی چهار عمل اصلی ممیزشناور به یکی از دو صورت زیر کران‌دار می‌شود:*

- *اگر از یکی از دو سبک گردکردن به پایین یا بالا استفاده شود آنگاه*

$$\frac{|x * y - x \circledast y|}{|x * y|} < \varepsilon_M.$$

- *اگر از سبک گردکردن به نزدیک‌ترین استفاده شود آنگاه*

$$\frac{|x * y - x \circledast y|}{|x * y|} \leq \frac{\varepsilon_M}{2}.$$

۲.۳.۱ برخی از خواص نامتعارف حساب ممیز شناور

۱. جمع ممیزشناور، لزوما شرکت‌پذیر نیست یعنی

$$\exists a, b, c \in F_{\beta,p}^{L,U} \ \ s.t. \quad (a \oplus b) \oplus c \neq a \oplus (b \oplus c).$$

بعنوان مثال دستگاه بازیچه‌ی $F_{2,3}^{-1,2}$ را با سبک گردکردن به نزدیک‌ترین (زوج) در نظر بگیرید. داریم:

$$0.5 \oplus (2.5 \oplus 0.75) = 0.5 \oplus fl(3.25) = 0.5 \oplus 3 = fl(3.5) = 3.5,$$
$$(0.5 \oplus 2.5) \oplus 0.75 = fl(3) \oplus 0.75 = 3 \oplus 0.75 = fl(3.75) = 4,$$

در حالی که هیچ یک از این دو نیز جواب درست ریاضی نیستند!

۲. ضرب ممیزشناور، لزوما شرکت‌پذیر نیست یعنی

$$\exists a, b, c \in F_{\beta,p}^{L,U} \ \ s.t. \quad (a \otimes b) \otimes c \neq a \otimes (b \otimes c).$$

۳. ضرب ممیزشناور، لزوما بر جمع ممیزشناور پخش‌پذیر نیست یعنی

$$\exists a, b, c \in F_{\beta,p}^{L,U} \ \ s.t. \quad a \otimes (b \oplus c) \neq (a \otimes b) \oplus (a \otimes c).$$

۴. ترتیب انجام عملیات ممیزشناور، گاهی اوقات بر درستی نتیجه تاثیرگذار است.

۵. خاصیت حذفی عمل جمع (و همینطور عمل ضرب) لزوما برقرار نیست یعنی

$$\exists a, b, c \in F_{\beta,p}^{L,U} \ \ s.t. \quad a \oplus b = a \oplus c \ \ \& \ \ b \neq c.$$

$$\exists a, b, c \in F_{\beta,p}^{L,U} \ \ s.t. \quad a \otimes b = a \otimes c \ \ \& \ \ b \neq c.$$

۶. حاصل‌ضرب یک عدد ممیزشناور در معکوسش لزوما مساوی یک نیست.

۱.۳.۳ برخی از فجایعی که در اثر استفاده‌ی نامناسب از محاسبات ممیزشناور رخ داده‌اند

با اینکه خطاهای گردکردن معمولا کوچک هستند ، وقتی در الگوریتم‌های طولانی و پیچیده چنین خطاهایی تکرار و انباشته می‌گردند، می‌توانند آثار فاچعه‌باری داشته باشند. در اینجا برخی از اتفاقاتی که در جهان واقعی بخاطر خطاهای محاسبات کامپیوتری رخ داده‌اند را مرور می‌کنیم:

۱. انفجار موشک آریان۵ در ۴ ژوئن ۱۹۹۶ در گینه‌ی فرانسه. این انفجار در اثر خطای سرریز در کامپیوتر تنظیم‌کننده‌ی مسیر حرکت موشک رخ داد. این موشک که توسط آژانس فضایی اروپا و با هزینه ۷ میلیون دلار ساخته شده بود، ۴۰ ثانیه پس از پرتاب در ارتفاع ۳۷۰۰ متری منفجر شد. دو هفته بعد از این رخداد گروهی از بازرسان گزارش خود را از دلایل انفجار موشک ارائه کردند. در گزارش بیان شده که یک عدد ممیزشناور ۶۴بیتی مربوط به شتاب افقی موشک نسبت به سکوی پرتاب باید پس از یک تبدیل در محل یک عدد صحیح ۱۶ بیتی ذخیره می‌شد. این عدد، بزرگ‌تر از ۳۲۷۶۷ که بزرگ‌ترین عدد قابل ذخیره در۱۶بیت است، بوده و در نتیجه با خطای سرریز رخ داده است و انحراف موشک از مسیر و انفجار آن در نتیجه‌ی این ایراد نرم افزاری بوده است.

۲. شکست ماموریت موشک آمریکایی در جریان جنگ خلیج فارس در ۲۵ فوریه ۱۹۹۱. موشک آمریکایی پاتریوت که در واقع یک موشک ضدموشک است، قرار بود پس از پرتاب از ظهران عربستان، موشک اسکادی را که توسط ارتش عراق پرتاب شده بود ردگیری کند اما بواسطه‌ی اشتباه در محاسبه‌ی زمان نتوانست موشک اسکاد را مورد اصابت قرار دهد و در نتیجه ۲۸ سرباز آمریکایی کشته شدند. در واقع زمان محاسبه‌شده توسط ساعت داخلی سیستم در واحد ۱۰برابر یک ثانیه اندازه‌گیری شده و نهایتا در عدد $\frac{1}{10}$ ضرب می شده تا زمان بر حسب ثانیه بدست آید. هرچند بسط دهدهی عدد $\frac{1}{10}$ فقط یک رقم بامعنا دارد، بسط دودویی آن نامتناهی است:

$$(0.1)_{10} = (1.10011001100\cdots)_2 \times 2^{-4} = (1.10011001\overline{100})_2 \times 2^{-4}.$$

در سیستم موشک پاتریوت ، عدد $\frac{1}{10}$ بعد از قطع‌شدن در یک ثبات ۲۴ بیتی ذخیره می‌شد. این خطای گردکردن البته کوچک بوده است. اما باتری موشک پاتریوت به مدت ۱۰۰ساعت در وضعیت آماده‌باش بوده است. زمان پرتاب موشک پاتریوت باید در ۱۰برابر تعداد ثانیه‌های موجود در۱۰۰ساعت (یعنی $10 \times 60 \times 60 \times 100$) ضرب می‌شده و این ضرب در عدد بزرگ باعث می شود که خطای کوچک گردکردن بزرگ شده و نهایتا موشک پاتریوت با تاخیری ۰٫۳۴ ثانیه‌ای پرتاب شود. از سوی دیگر موشک اسکاد در هر ثانیه تقریبا ۱۶۷۶ متر را می‌پیماید و در نتیجه در این مدت زمان بیش از ۵۰۰ متررا طی می‌کند و این فاصله خارج از برد پوشش‌داده شده توسط یک موشک پاتریوت است. درنتیجه موشک اسکاد نهایتا به هدف اصابت می‌کند.

۳. تغییراحزاب تشکیل دهنده‌ی پارلمان آلمان در سال ۱۹۹۲ در اثر خطای گردکردن. در سیستم پیچیده‌ی انتخابات آلمان بصورت است اگر حزبی کمتر از پنج درصد آراء رابدست آورد، نمی‌تواند وارد پارلمان شده و کلیه آراء آن حزب حذف شده و کرسی‌های پارلمانی مربوطه بین سایر احزاب بصورت خاصی پخش می‌شود . پس از اعلام نتایج انتخابات ۵ آوریل ۱۹۹۲ ، اعلام می‌شود که حزب سبزها توانسته پنج درصد آراء را بدست آورد اما بعد از نیمه شب یکی از اعضای کمیته‌ی انتخابات متوجه شد که حزب سبزها در واقع توانسته بوده 4.97% آراء را بدست آورد، اما برنامه ای که محاسبات را انجام می‌داده ، تنها یک رقم بعداز ممیز اعشار را پس از گرد کردن به سمت بالا نهایتا چاپ می‌کرده و به همین دلیل عدد 4.97% را به پنج درصد تبدیل کرده بود. پس از مشخص‌شدن این اشتباه حزب سبزها نتوانست وارد پارلمان شود و حزب SPD توانست اکثریت پارلمان را به خود اختصاص دهد.