بنام خداوند جان و خرد

دانشگاه صنعتی خواجه نصیرالدین طوسی

معماری کامپیوتر

جلسه هفدهم: محاسبات اعشاری ممیز شناور

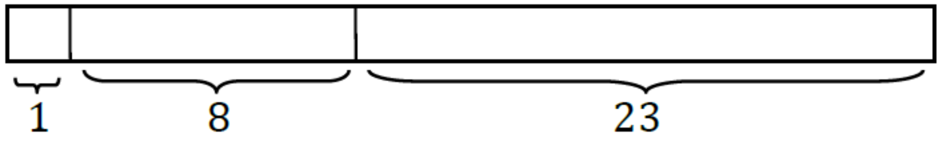
اعداد اعشاری ممیز شناور

- روال تبدیل و ذخیره اعداد به فرمت ممیز شناور:
  - عدد را هنجار (نرمال) می‌کنیم
    - همه اعداد به‌جز صفر را می‌توان هنجار کرد (نمایش صفر با کوچکترین عدد قابل نمایش)
  - نما و اعشار بدست آمده را در قالب نمایش جایگذاری می‌کنیم
    - قسمت اعشاری (مانتیس) همیشه مثبت است و به همان شکل در حافظه قرار می‌گیرد
    - قسمت نما می‌تواند مثبت یا منفی بوده و نمایش مکمل دو دارد
    - در ذخیره این دو بخش تعداد بیت تخصیص داده شده مهم است
      - امکان رخداد سرریز یا زیرریز در ذخیره نما

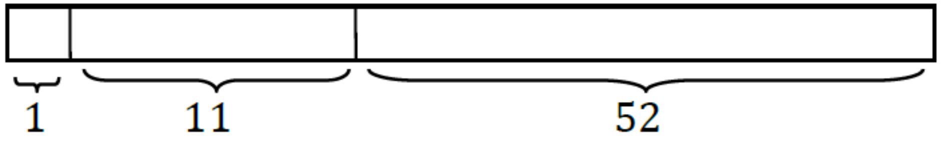
اعداد اعشاری ممیز شناور

- سایز هر بخش در فرمت ذخیره‌سازی مهم است
  - استاندارد IEEE 754 برای تنظیم فضای e و f

Single Precision(دقت ساده): 32 bits (Exponent 8 bits, Fraction 23 bits)



Double Precision(دقت مضاعف):64 bits (Exponent 11 bits, Fraction 52 bits)



نمایش اعداد در سیستم ممیز شناور

- مقدار بایاس در هر سیستم نمایش برحسب تعریف مشخص است
- استخراج عدد ممیز شناور بایاس شده:

$$(-1)^S 1.F * 2^{e\text{-}bias}$$

- تعداد اعداد اعشاری در بازه صفر تا یک بی‌نهایت است پس قادر به نمایش همه اعداد نیستیم

- **قرارداد نمایش**
  - Bias #1 = 2^{e-1}
  - Bias #2: 2^{e-1}-1: رزرو محدوده یک نما برای اعداد خاص

نمایش اعداد در سیستم ممیز شناور

- فرض کنیم طول بخش‌های fraction و نما برابر f و e بیت باشند:

  - **حداقل مقدار اعشار(F_{min}) :** صفر (تمامی ارقام برابر صفر) به‌دلیل بی‌علامت بودن این بخش
  - **حداکثر مقدار اعشار(F_{max}):** تمامی ارقام اعشار برابر یک ($1-2^{-f}$)
  - **حداقل مقدار نما(E_{min}):** باتوجه به علامت‌دار بودن این بخش برابر -2^{e-1}
  - **حداکثر مقدار نما(E_{max}):** باتوجه به علامت‌دار بودن این بخش برابر $2^{e-1}-1$

نمایش اعداد در سیستم ممیز شناور

- کمترین مقدار مثبت قابل ذخیره در سیستم ممیز شناور:

$$\varepsilon = N_{min} = 1.F_{min} * 2^{E_{min}} \rightarrow 1.0 * 2^{-2^{e-1}}$$

- عدد ε به‌عنوان کوچکترین مقدار قابل نمایش، در برخی سیستم‌ها معادل صفر لحاظ می‌شود
  - درصورتی که e=8، این عدد معادل 2^{-128} می‌باشد که خیلی کوچک است
- دومین کوچکترین عدد پس از ε چگونه بدست می‌آید؟

نمایش اعداد در سیستم ممیز شناور

- دومین کوچکترین عدد پس از ε
  - افزایش مقدار fraction به‌اندازه 2^{-f}

$$N_{min+1} = (1 + 2^{-f}) * 2^{-2^{e-1}}$$

  - تفاضل N_{min} و N_{min+1}:

$$\Delta_1 = 2^{-f} * 2^{-2^{e-1}}$$

- روال افزایش گام به گام fraction را تا انتها ادامه می‌دهیم:

$$N_{min+2^f-1} = (2 - 2^{-f}) * 2^{-2^{e-1}}$$

- فاصله هردو عدد متوالی در این‌حالت برابر Δ_1 است

نمایش اعداد در سیستم ممیز شناور

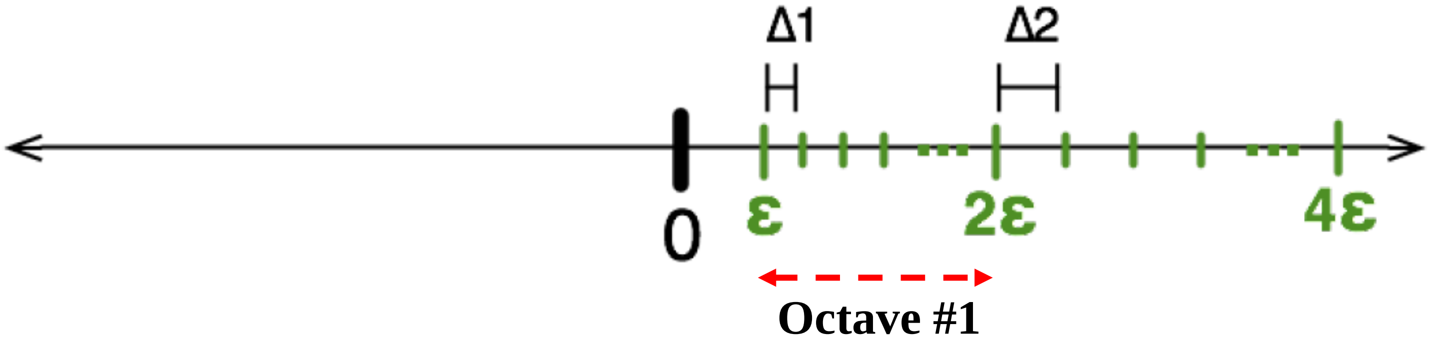
- با بیشینه شدن fraction، برای افزایش باید e را زیاد کنیم:
  - عدد بزرگتر بعدی: $N = 1.0 * 2^{-2^{e-1}+1}$
  - فاصله با عدد قبلی: $\Delta_2 = 2^{-f} * 2^{-2^{e-1}+1}$
  - به‌دلیل افزایش یک واحدی نما $\Delta_2 = 2\Delta_1$
  - در نتیجه هرچه از صفر فاصله بگیریم، فاصله اعداد بیشتر می‌شود
    - فاصله بین اعداد نمادی از دقت نمایش

نمایش اعداد در سیستم ممیز شناور

- اکتاو: سطوح نمای مختلف در نمایش ممیز شناور
  - تعداد اعداد قابل نمایش در اکتاوها یکسان است (2^f)
  - هرچه اکتاو بالاتر باشد، فاصله اعداد آن بیشتر هستند

$$\Delta_1 = 2^{-f} * 2^{-2^{e-1}}, \Delta_i = 2^{i-1} * \Delta_1$$

- اعداد قابل نمایش در سیستم ممیز شناور



نمایش اعداد در سیستم ممیز شناور

- ویژگی‌های سیستم نمایش ممیز شناور(سیستم bias #2)
  - **تعداد اعداد قابل نمایش:** $2 * 2^{f} * 2^{e} - 1$
  - **بازه اعداد قابل نمایش:** $[-N_{max}, N_{max}]$
  - **حداقل و حداکثر اعداد:** $N_{max} = (2-2^{-f}) * 2^{Emax}$, $N_{min} = -N_{max}$
  - **دقت اعداد ذخیره شده:** Δ که برحسب نما تعیین می‌شود

مقایسه سیستم ممیز ثابت و ممیز شناور

- هزینه سخت‌افزاری کمتر نمایش ممیز ثابت نسبت به ممیز شناور
- تاخیر محاسبات کمتر نمایش ممیز ثابت نسبت به ممیز شناور
- عدم انعطاف‌پذیری نمایش ممیز ثابت نسبت به ممیز شناور
- قابلت نمایش اعداد خاص مانند e، π، Nan ... در سیستم نمایش ممیز شناور

جمع و تفریق اعداد اعشاری ممیز شناور

- ابتدا لازم است اعداد هم‌نما شوند
- عدد کوچک را به نمای عدد بزرگ می‌بریم
  - برای این‌کار لازم است شیفت به راست داده و از بیت‌های کم‌ارزش صرف‌نظر می‌شود
  - مثال: $3.1415*10^{2}$, $2.344 * 10^{-6}$ و ذخیره یک رقم صحیح و چهار رقم اعشار
  - $314150000.0000*10^{-6}$, $2.344*10^{-6}$: دور ریخته شدن قسمت پرارزش براثر کمبود فضا
  - $3.1415*10^{2}$,$0.000002344*10^{2}$: دور ریخته شدن قسمت کم‌ارزش براثر کمبود فضا
  - پس روش دوم در کامپیوترها استفاده می‌شود

جمع و تفریق اعداد اعشاری ممیز شناور

- الگوریتم:
  - مقایسه نماها و هم‌ردیف کردن آن‌ها
    - اعشار عدد با نمای کوچکتر را به‌اندازه اختلاف نماها به سمت راست شیفت می‌دهیم
  - دو عدد را باهم جمع/تفریق می‌کنیم (اندازه علامت)
  - درصورتی که حاصل هنجار نباشد، آن را مطابق قالب نرمال می‌کنیم
    - یک رقم یک قبل ممیز و بقیه ارقام بعد ممیز $(-1)^{s}1.F2^{E}$

جمع و تفریق اعداد اعشاری ممیز شناور

- در حین جمع/تفریق ممکن است سرریز/زیرریز رخ دهد
  - سرریز (Overflow): قبل ممیز رقمی بزرگتر از یک وجود داشته باشد
    - قابل رخداد در جمع و ضرب
  - زیرریز (underflow): قبل از ممیز فقط رقم صفر وجود داشته باشد
    - قابل رخداد در تفریق و تقسیم

$$\begin{array}{r} 1.0011 \times 2^{10} \\ +\,1.0010 \times 2^{10} \\ \hline 10.0101 \times 2^{10} \end{array}$$

$$\begin{array}{r} 1.0011 \times 2^{10} \\ -\,1.0010 \times 2^{10} \\ \hline 0.0001 \times 2^{10} \end{array}$$

جمع و تفریق اعداد اعشاری ممیز شناور

- مثال: دو عدد $(0.5)_{10}$ و $(-0.4375)_{10}$ را در سیستم باینری ممیز شناور جمع کنید.

- **حل:**

$(0.5)_{10} = (0.1)_2$, $(-0.4375)_{10} = (-1.110*2^{-2})_2$

1) هم‌نما کردن: $1.0 * 2^{-1}$, $-0.111 * 2^{-1}$

2) جمع کردن: $(0.001)_2*2^{-1}$

3) هنجارسازی: $1.0*2^{-4}$

ضرب اعداد اعشاری ممیز شناور

- الگوریتم:
  - چک کردن صفر
  - جمع نماها با یکدیگر: $E_r = E_a + E_b - bias$
    - کم کردن بایاس با هدف آنکه یکبار در محاسبات دخیل شود و نه دوبار
  - تعیین علامت حاصل‌ضرب: $S_r = S_A \oplus S_B$
  - انجام عمل ضرب روی اعداد اعشار $X = 1.F_A * 1.F_B$
  - هنجار کردن نتیجه بدست آمده: $\mathbf{(-1)^{Sr} * X * 2^{Er}}$

ضرب اعداد اعشاری ممیز شناور

- مثال: دو عدد $(0.5)_{10}$ و $(-0.4375)_{10}$ را در سیستم باینری ممیز شناور ضرب کنید.

- حل:

$(0.5)_{10} = (0.1)_{2} = (1.0 * 2^{-1})_2$, $(-0.4375)_{10} = (-1.110*2^{-2})_2$

1) جمع نماها باهم: $E_r = -1 + (-2) = -3$

2) تعیین علامت حاصل‌ضرب: $1 \oplus 0 = 1$

3) انجام عملیات ضرب: $1.000 * 1.110 = 1.110$

4) هنجار کردن نتیجه: $- 1.110 * 2^{-3}$

تقسیم اعداد اعشاری ممیز شناور

- الگوریتم:
  - چک کردن صفر
  - تفریق نماها از یکدیگر: $E_r = E_a - E_b + bias$
    - جمع کردن بایاس با هدف آنکه یکبار در محاسبات دخیل شود و نه دوبار
  - تعیین علامت حاصل‌تقسیم: $S_r = S_A \oplus S_B$
  - انجام عمل تقسیم روی اعداد اعشار $X = 1.F_A / 1.F_B$
  - هنجار کردن نتیجه بدست آمده: $\mathbf{(-1)^{Sr} * X * 2^{Er}}$

نمایش اعداد BCD

- نمایش Binary Coded Decimal
  - هر رقم (۹-۰) در این سیستم نمایش با چهار بیت نشان داده می‌شود
  - این کدگذاری در ارتباطات سیستم‌های I/O استفاده می‌شود
  - کامپیوتر ورودی را به‌صورت BCD گرفته و سپس باینری می‌کند
    - 595: 0101 1001 0101
  - برای محاسبات می‌توان، ابتدا ورودی را باینری کرد که پیچیده و زمان‌بر است
    - پس محاسبات مستقیما روی BCD انجام می‌شود

جمع اعداد BCD

- حاصل جمع دو رقم (دو چهار بیت) در هشت بیت ذخیره می‌شود
  - چهار بیت برای یکان و چهار بیت برای دهگان
- اگر حاصل‌جمع دو رقمی شود باید با ۶ جمع شده و در هشت بیت ذخیره گردد
  - در حالت دسیمال اگر عدد دورقمی AB داشته باشیم یعنی: A*10 + B (14 = 1*10+4)
  - در حالت BCD اگر عدد دو رقمی AB داشته باشیم یعنی: A*16+B (14 = 1*16+4)
  - تفاضل این دو حالت یعنی A*6+B که در BCD استفاده می‌شود
    - در جمع دو رقم چهاربیتی بین صفر و نه، A می‌تواند صفر یا یک باشد

جمع اعداد BCD

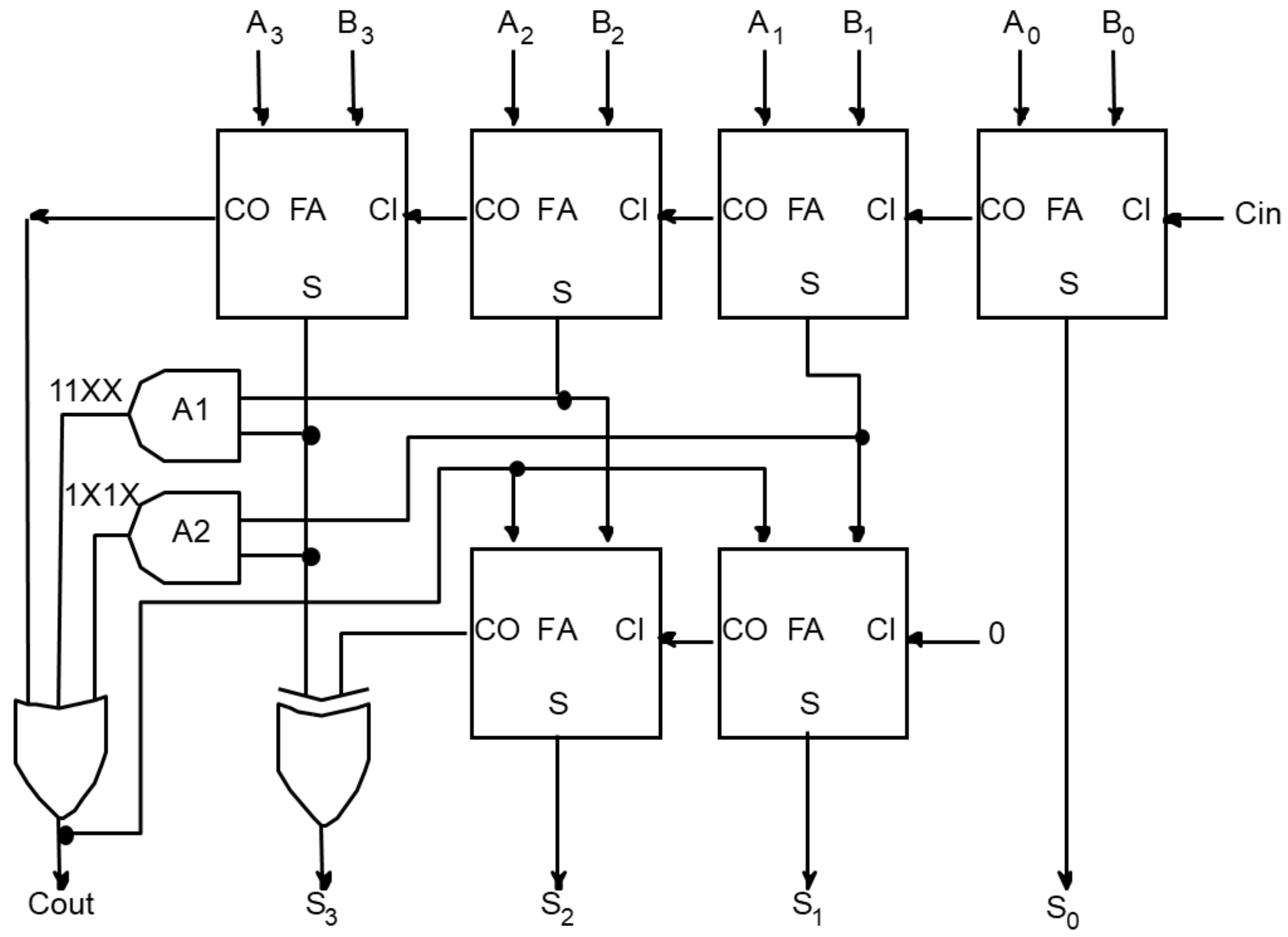
- لازم است مداری برای تشخیص دورقمی بودن حاصل‌جمع داشته باشیم
  - تابع F برای این هدف تعریف شده است
    - F=1: حاصل‌جمع دورقمی
    - F=0: حاصل‌جمع تک‌رقمی

حاصل‌جمع‌ها

	C	$S_3S_2S_1S_0$
0	0	0000
1	0	0001
2	0	0010
3	0	0011
4	0	0100
5	0	0101
6	0	0110
7	0	0111
8	0	1000
9	0	1001
10	0	1010
11	0	1011
12	0	1100
13	0	1101
14	0	1110
15	0	1111
16	1	0000
17	1	0001
18	1	0010

$$F=C+S_3S_2+S_3S_1$$

جمع اعداد BCD



تفریق اعداد BCD

- بجای A-B، A+10’s(B) را حساب می‌کنیم
- اگر حاصل جمع بیت‌نقلی تولید کند، مثبت است و بیت نقلی را درنظر نمی‌گیریم
- اگر حاصل‌جمع بیت‌نقلی تولید نکند، منفی و به‌صورت مکمل ۱۰ است
  - A-B: c=1→A>B
  - A-B: c=0→A<B

A	10's(A)
0	10
1	9
2	8
3	7
4	6
5	5
6	4
7	3
8	2
9	1
10	0