به نام خدا

مقدمه مؤلف

این کتاب حاصل ۱۸ سال تدریس در دانشگاهها و مراکز آموزش عالی کشور می‌باشد. این کتاب براساس آخرین تغییرات سرفصل‌ها توسط وزارت علوم نگارش شده است. مطالعهٔ مراجع اصلی درس معماری کامپیوتر، کاری بسیار دشوار و زمان‌بر است ولی با مطالعهٔ این کتاب شما بی‌نیاز از مطالعهٔ هر منبع دیگری می‌شوید. نویسنده کتاب با تجربهٔ بسیار زیاد در زمینهٔ کنکور کارشناسی ارشد و دکتری این کتاب را بسیار کامل و جامع می‌داند. ویرایش جدید، مناسب برای داوطلبان کنکور کارشناسی ارشد و دکتری و همچنین دانشجویانی است که در دانشگاه این درس را می‌گذرانند. این کتاب در ۹ فصل تنظیم شده است و در پایان هر فصل تست‌های کنکورهای رسمی و تألیفی با پاسخ‌های تشریحی گردآوری شده‌اند. در پایان کتاب، ضمیمه‌ای آمده است که شامل مسائل تشریحی و کنکورهای رسمی سال ۹۳ تا ۹۹ است.

دوستان عزیزی در آماده‌سازی این کتاب مرا یاری کردند که از همه آنها سپاسگزارم. اگر پیشنهاد یا انتقادی در مورد این کتاب یا سایر کتاب‌های تألیف اینجانب داشتید لطفاً مطرح کنید.

هادی یوسفی – پاییز ۱۳۹۹
hadi_yusefi@ yahoo.com
۰۹۱۲۱۷۸۸۵۳۴

فهرست مطالب

فصل ۱ اعداد

۱) مقدمه

در درس مدار منطقی با اعداد آشنا شدیم و تبدیل مبنا و مکمل‌ها را آموختیم. سعی می‌کنیم خلاصه‌ای از مطالب مهم که در درس معماری کامپیوتر نیاز داریم را ارائه دهیم.

نمایش عدد a در مبنای r به شکل $a=(a_{n-1}a_{n-2}...a_1a_0/a_{-1}a_{-2}...a_{-m})_r$ است که n تا رقم صحیح $(a_{n-1},..,a_1,a_0)$ و m تا رقم اعشار $(a_{-m},..,a_{-2},a_{-1})$ دارد. ارقامِ $a_i\ (-m\le i\le n-1)$ در بازهٔ ۰ تا $r-1$ هستند. مثلاً در مبنای $r=2$ (binary) ارقام مجاز ۰ و ۱ هستند یا در مبنای $r=10$ (decimal) ارقام مجاز ۰ و ۱ و ۲و ... و ۹ هستند. برای تبدیل عدد a از مبنای دلخواه r به مبنای ۱۰ به هر رقم ارزش می‌دهیم (ارزش رقم a_i برابر r^i است) و سپس ارقام را در ارزش‌ها ضرب کرده و مقادیر بدست آمده را جمع می‌کنیم. پس a در مبنای ۱۰ برابر $a=\sum_{i=-m}^{n-1}a_ir^i$ است. به چند نمونه توجه کنید:

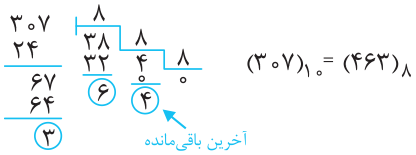
$$(A۴/C)_{۱۶}=A۴/C_{hex}=A\times۱۶^{۱}+۴\times۱۶^{۰}+C\times۱۶^{-۱}=۱۶۰+۴+\frac{۱۲}{۱۴}=(۱۶۴/۷۵)_{۱۰}$$

(در مبنای ۱۶ از ارقام ۰ تا ۱۵ استفاده می‌شود که ۱۰ و ۱۱ و ۱۲ و ۱۳ و ۱۴ و ۱۵ به ترتیب با A و B و C و D و E و F نمایش داده می‌شوند)

$$(۱۰۱۱۰۰/۱۰۱)_{۲}=۳۲+۸+۴+\frac{۱}{۲}+\frac{۱}{۸}=(۴۴/۶۲۵)_{۱۰}$$

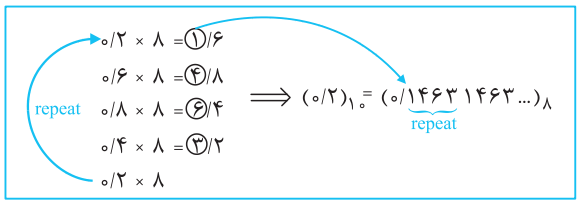
برای تبدیل یک عدد از مبنای ۱۰ به مبنای r، قسمت صحیح را به r تقسیم کرده و سپس خارج قسمت تقسیم را مجدداً به r تقسیم می‌کنیم تا زمانیکه خارج قسمت صفر شود و سپس

باقیمانده‌ها را یادداشت می‌کنیم، فقط توجه کنید آخرین باقی‌مانده با ارزش‌ترین رقم است. مثلاً بـه تبدیل عدد $(۳۰۷)_{۱۰}$ به مبنای ۸ دقت کنید:



$(۳۰۷)_{۱۰} = (۴۶۳)_{۸}$

برای تبدیل قسمت اعشار به مبنای r، قسمت اعشار را متوالیاً در r ضرب می‌کنیم و پس از هر عمل ضرب، قسمت صحیح حاصلضرب را یادداشت می‌کنیم. عمل ضرب تا زمانی انجام می‌شود کـه یا قسمت اعشار به صفر برسد و یا به تکرار منجر شود. به تبدیل $(۰/۲)_{۱۰}$ به مبنای ۸ توجه کنید:



در درس منطقی با مکمل‌ها آشنا شدیم. کاربرد مکمـل‌هـا نمـایش اعـداد منفـی و انجـام عمـل تفریق است. در مبنای r دو نوع مکمل‌گیری تعریـف مـی‌شـود: مکمـل r و مکمـل $r-۱$. مـثلاً در مبنای ۲، مکمل ۱ و مکمل ۲ معنی دارد. یا در مبنای ۱۰، مکمل ۹ و مکمل ۱۰ تعریـف مـی‌شـود. برای یافتن مکمل $r-۱$ عدد a در مبنای r، کافی است همهٔ ارقام a را از $r-۱$ کـم کنـیم. مـثلاً مکمل ۹ عدد $a = (۷۳۰/۴)_{۱۰}$ برابر $[a]_{۹} = (۲۶۹/۵)_{۱۰}$ است (همـهٔ ارقـام را از ۹ کـم کـردیم). یـا مکمل ۷ عدد $a = (۷۳۰)_{۸}$ برابـر $[a]_{۷} = (۰۴۷)_{۸}$ مـی‌باشـد. یـا مکمـل ۱ عـدد $(۱۰۰۱۰)_{۲}$ برابـر $(۰۱۱۰۱)_{۲}$ است (همهٔ ارقام را از ۱ کم کردیم) پس در واقع مکمل ۱ کردن در مبنای ۲ یعنـی not کردن. برای یافتن مکمل r عدد a در مبنای r می‌توان ابتدا مکمل $r-۱$ عـدد a را محاسـبه کـرده سپس رقم سمت راست آن را یک واحد اضافه کنیم. یا روش سریعتر برای یافتن مکمـل r آن اسـت که صفرهای سمت راست عدد را در صورت وجود تغییر ندهیم، اولین رقم غیرصفر (از سمت راست) را از r کم کنیم و سایر ارقـام را از $r-۱$ کـم کنـیم. مـثلاً مکمـل ۱۰ عـدد $a = (۷۳۰/۴)_{۱۰}$ برابـر $[a]_{۱۰} = (۲۶۹/۶)_{۱۰}$ است (۴ را از ۱۰ و سایر ارقام را از ۹ کم کردیم). مکمل ۸ عـدد $a = (۷۳۰)_{۸}$ برابر $[a]_{۸} = (۰۵۰)_{۸}$ است. مکمـل ۲ عـدد $a = (۱۰۰۱۰)_{۲}$ برابـر $[a]_{۲} = (۰۱۱۱۰)_{۲}$ اسـت. یعنـی برای یافتن مکمل ۲ عدد a در مبنای ۲، صفرهای سمت راست و اولین ۱ (سـمت راسـت) را تغییـر نمی‌دهیم و سایر بیت‌ها را مکمل (not) می‌کنیم.

۲) نمایش اعداد علامت‌دار در مبنای ۲

برای نمایش اعداد صحیح علامت‌دار در مبنای ۲ روش‌های زیر مطرح شده است:

- روش علامت مقدار (sign magnitude)
- روش مکمل ۱ (ones complement)
- روش مکمل ۲ (twos complement)
- روش بایاس (bias) یا افزونه (excess)

در روش علامت مقدار برای یافتن قرینهٔ یک عدد، بیت سمت چپ (MSB) آن عدد را عوض می‌کنیم. مثلاً با ۴ بیت ۳+ = ۰۰۱۱ و ۳− = ۱۰۱۱. در روش مکمل یک برای یافتن قرینهٔ یک عدد، آن عدد را مکمل یک می‌کنیم یعنی همهٔ بیت‌ها را عوض می‌کنیم مثلاً ۳+ = ۰۰۱۱ و ۳− = ۱۱۰۰. در روش مکمل دو برای یافتن قرینهٔ یک عدد، آن عدد را مکمل ۲ می‌کنیم مثلاً ۳+ = ۰۰۱۱ و ۳− = ۱۱۰۱. در این سه روش اعداد مثبت به یک شکل نمایش داده می‌شوند و تفاوت، در نمایش اعداد منفی است، البته بیت سمت چپ اعداد منفی در این سه روش، یک است. مثلاً ۱۱۰۱ عددی منفی است. در روش علامت مقدار ۵− = (۱۰۱)− = ۱۱۰۱، در روش مکمل یک ۲− = (۰۰۱۰)− = ۱۱۰۱ و در روش مکمل دو، ۳− = (۰۰۱۱)− = ۱۱۰۱ می‌باشد. در روش بایاس، کوچکترین عدد را به شکل ۰۰۰۰ و بزرگترین عدد را به شکل ۱۱۱۱ نمایش می‌دهند یعنی اعداد را به شکل بی‌علامت نمایش می‌دهند و برای تشخیص مقدار واقعی آنها باید مقداری به اسم bias را از مقدار ظاهری عدد، کم کنیم. با n بیت مقدار bias یا برابر $۲^{n-۱}$ یا $۲^{n-۱}-۱$ می‌باشد. پس با ۴ بیت مقدار bias یا ۸ یا ۷ می‌باشد پس ۰۰۰۰ که ظاهراً صفر است، واقعاً یا ۸− = ۰−۸ یا ۷− = ۷−۰ است و ۱۱۱۱ که ظاهراً برابر ۱۵ است واقعاً یا ۷+ = ۸−۱۵ یا ۸+ = ۷−۱۵ می‌باشد. البته در مسائل گفته می‌شود که bias چه مقداری است. جدول زیر با ۴ بیت همهٔ اعداد را در سیستم‌های علامت‌دار نشان می‌دهد.

مقداردهدهی واقعی	سیستم علامت مقدار	سیستم مکمل یک	سیستم مکمل دو	سیستم بایاس ۸	سیستم بایاس ۷
+۸	–	–	–	–	۱۱۱۱
+۷	۰۱۱۱	۰۱۱۱	۰۱۱۱	۱۱۱۱	۱۱۱۰
+۶	۰۱۱۰	۰۱۱۰	۰۱۱۰	۱۱۱۰	۱۱۰۱
+۵	۰۱۰۱	۰۱۰۱	۰۱۰۱	۱۱۰۱	۱۱۰۰
+۴	۰۱۰۰	۰۱۰۰	۰۱۰۰	۱۱۰۰	۱۰۱۱
+۳	۰۰۱۱	۰۰۱۱	۰۰۱۱	۱۰۱۱	۱۰۱۰
+۲	۰۰۱۰	۰۰۱۰	۰۰۱۰	۱۰۱۰	۱۰۰۱
+۱	۰۰۰۱	۰۰۰۱	۰۰۰۱	۱۰۰۱	۱۰۰۰
+۰	۰۰۰۰	۰۰۰۰	۰۰۰۰	۱۰۰۰	۰۱۱۱
−۰	۱۰۰۰	۱۱۱۱	–	–	–
−۱	۱۰۰۱	۱۱۱۰	۱۱۱۱	۰۱۱۱	۰۱۱۰
−۲	۱۰۱۰	۱۱۰۱	۱۱۱۰	۰۱۱۰	۰۱۰۱
−۳	۱۰۱۱	۱۱۰۰	۱۱۰۱	۰۱۰۱	۰۱۰۰
−۴	۱۱۰۰	۱۰۱۱	۱۱۰۰	۰۱۰۰	۰۰۱۱
−۵	۱۱۰۱	۱۰۱۰	۱۰۱۱	۰۰۱۱	۰۰۱۰
−۶	۱۱۱۰	۱۰۰۱	۱۰۱۰	۰۰۱۰	۰۰۰۱
−۷	۱۱۱۱	۱۰۰۰	۱۰۰۱	۰۰۰۱	۰۰۰۰
−۸	–	–	۱۰۰۰	۰۰۰۰	–

برای تبدیل عدد باینریِ $a=(a_{n-1}a_{n-2}...a_1a_0)_2$ به مبنای ۱۰ یعنی برای یافتن مقدار واقعی a در سیستم‌های مختلف می‌توان از فرمول‌های زیر کمک گرفت:

$$\text{unsigned}: a = a_{n-1}\times 2^{n-1} + a_{n-2}\times 2^{n-2} + ... + a_1\times 2^1 + a_0\times 2^0 = \sum_{i=0}^{n-1} a_i (2)^i$$

$$\text{sign magnitude}: a = (-1)^{a_{n-1}} \times \left[a_{n-2}\times 2^{n-2} + ... + a_1\times 2^1 + a_0\times 2^0 \right]$$

$$\text{ones complement}: a = a_{n-1}\times -(2^{n-1}-1) + a_{n-2}\times 2^{n-2} + ... + a_1\times 2^1 + a_0\times 2^0$$

$$\text{twos complement}: a = a_{n-1}\times -2^{n-1} + a_{n-2}\times 2^{n-2} + ... + a_1\times 2^1 + a_0\times 2^0$$

$$\text{bias system}: a = \sum_{i=0}^{n-1} a_i (2)^i - \text{bias}$$

در سیستم‌های بی‌علامت، مکمل ۱ و مکمل ۲ همهٔ بیت‌ها دارای ارزش هستند و فقط ارزش بیت سمت چپ متفاوت است. ارزش بیت سمت چپ در بی‌علامت 2^{n-1} در سیستم مکمل دو -2^{n-1} و در سیستم مکمل یک $-(2^{n-1}-1)$ می‌باشد. در سیستم علامت مقدار، بیت سمت چپ بی‌ارزش است و فقط علامت را نشان می‌دهد. در سیستم بایاس مقدار عدد را با این فرض که بی‌علامت است می‌یابیم و سپس مقدار bias را از آن کم می‌کنیم.

مثال ۱: مقدار عدد ۸ بیتی $a=(10100101)_2$ در سیستم‌های مختلف برابر است با:

$$\text{unsigned}: a=(\overset{2^7}{1}\ \overset{2^6}{0}\ \overset{2^5}{1}\ \overset{2^4}{0}\ \overset{2^3}{0}\ \overset{2^2}{1}\ \overset{2^1}{0}\ \overset{2^0}{1})_2=128+32+4+1=165$$

$$\text{sign magnitude}: a=(\overset{-}{1}\ \overset{2^6}{0}\ \overset{2^5}{1}\ \overset{2^4}{0}\ \overset{2^3}{0}\ \overset{2^2}{1}\ \overset{2^1}{0}\ \overset{2^0}{1})_2=-(32+4+1)=-37$$

$$\text{ones complement}: a=(\overset{-(2^7-1)}{1}\ \overset{2^6}{0}\ \overset{2^5}{1}\ \overset{2^4}{0}\ \overset{2^3}{0}\ \overset{2^2}{1}\ \overset{2^1}{0}\ \overset{2^0}{1})_2=-127+32+4+1=-90$$

$$\text{twos complement}: a=(\overset{-2^7}{1}\ \overset{2^6}{0}\ \overset{2^5}{1}\ \overset{2^4}{0}\ \overset{2^3}{0}\ \overset{2^2}{1}\ \overset{2^1}{0}\ \overset{2^0}{1})_2=-128+32+4+1=-91$$

$$\text{bias } 2^{8-1}: a=165-2^{8-1}=165-128=+37$$

$$\text{bias } 2^{8-1}-1: a=165-(2^{8-1}-1)=165-127=+38$$

نکته

۱- با توجه به فرمول‌های گفته شده با n بیت بزرگترین و کوچکترین (Range) عدد قابل نمایش در سیستم‌های مختلف طبق جدول زیر است:

system	Max	Min
unsigned	$111...1=2^n-1$	$000..0=0$
sign magnitude	$011...1=+(2^{n-1}-1)$	$111..1=-(2^{n-1}-1)$
ones complement	$011..1=+(2^{n-1}-1)$	$100..0=-(2^{n-1}-1)$
twos complement	$011..1=+(2^{n-1}-1)$	$100..0=-2^{n-1}$
bias 2^{n-1}	$111..1=+(2^{n-1}-1)$	$000..0=-2^{n-1}$
bias $2^{n-1}-1$	$111..1=+2^{n-1}$	$000..0=-(2^{n-1}-1)$

۲- n بیت دارای 2^n حالت است پس با هر سیستمی حداکثر 2^n عدد قابل نمایش است. سیستم‌های علامت مقدار و مکمل یک 2^n-1 عدد نمایش می‌دهند که علت آن وجود صفر مثبت و صفر منفی در این دو سیستم است و این یکی از معایب این دو سیستم است. در سیستم علامت مقدار $+0=000..0$ و $-0=100..0$ و در سیستم مکمل یک $+0=000..0$ و $-0=111..1$ می‌باشند. ولی سایر سیستم‌ها با n بیت دقیقاً 2^n عدد مختلف را نمایش می‌دهند.

۳- فلگ‌ها و جمع و تفریق سیستم‌های مختلف

یک رجیستر یا ثبات وضعیت (status register)، رجیستر فلگ یا رجیستر کد شرط، مجموعه‌ای از تعدادی فلگ یا بیت‌های وضعیت است. در برخی ماشین‌ها نام این رجیستر، FLAGS یا PSW (Program Status Word) است. متداول‌ترین بیت‌های وضعیت عبارتند از Z

و S (یا N) و C و V. این فلگ‌ها پس از انجام برخی عملیات محاسباتی و منطقی مقداردهی می‌شوند. فلگ Z یا ZF (Zero flag) وقتی یک می‌شود که حاصل عملیات صفر باشد. فلگ C (carry flag) وقتی یک می‌شود که از باارزش‌ترین بیت (Most Significant Bit : MSB) رقم نقلی خارج شود یا رقم قرضی تولید شود. فلگ S(Sign) یا N(Negative) وقتی یک می‌شود که حاصل عملیات منفی باشد یعنی بیت سمت چپ حاصل، یک باشد. فلگ V یا OF (Overflow flag) وقتی یک می‌شود که حاصل عملیات سرریز شود یعنی در بازهٔ مجاز نباشد. جمع و تفریق و وضعیت فلگ‌ها را در سیستم‌های مختلف بررسی می‌کنیم.

سیستم مکمل ۲: در اکثر ماشین‌ها برای نمایش اعداد علامت‌دار از این سیستم استفاده میشود و معمولاً هر وقت گفته می‌شود عدد علامت‌دار، منظور سیستم مکمل ۲ است. همانطور که دیدیم برای اعداد n بیتی در این سیستم، ارزش بیت سمت چپ $-۲^{n-۱}$ است.

مثال ۲: مقدار چند عدد با تعداد بیت‌های مختلف را در سیستم مکمل ۲ محاسبه می‌کنیم:

۱۱۱۱ = −۸ + ۴ + ۲ + ۱ = −۱	۱۰۰۱ = −۸ + ۱ = −۷
۱۱۱۱۱۱۱۱ = −۱۲۸ + ۶۴ + ۳۲ + ۱۶ + ۸ + ۴ + ۲ + ۱ = −۱	۱۱۰۰۱ = −۱۶ + ۸ + ۱ = −۷
۰۱۱۰ = ۴ + ۲ = +۶	۱۱۱۰۰۱ = −۳۲ + ۱۶ + ۸ + ۱ = −۷

نکته

در سیستم مکمل ۲، اگر بیت سمت چپ، تکرار شود (اصطلاحاً sign extend شود) مقدار عدد، عوض نمی‌شود. یعنی برای تبدیل یک عدد مثلاً ۴ بیتی به ۸ بیتی باید بیت سمت چپ عدد را تکرار کنیم. مثلاً ۱۰۰۱ با ۱۱۱۱۱۰۰۱ برابر است یا ۰۱۱۰ با ۰۰۰۰۰۱۱۰ برابر است.

برای انجام عمل جمع در این سیستم چه اعداد مثبت باشند چه منفی، مهم نیست و اعداد را به طور عادی با هم جمع می‌کنیم.

مثال ۳: فرض کنید ماشین ۴ بیتی است. جمع‌های زیر را در سیستم مکمل ۲ انجام داده و وضعیت فلگ‌ها ار مشخص کرده‌ایم:

توجه کنید حاصل در ۴ بیت ذخیره می‌شود و بیت پنجم که رقم نقلی است، صرفنظر می‌شود. بیت چهارم نیز فلگ S است. همچنین حاصل در بازهٔ مجاز است و سرریز ندارد.

```
  ۱
  ۱ ۱ ۰ ۱   =-۳
+ ۱ ۱ ۰ ۰   =-۴
-------------
① ۱ ۰ ۰ ۱   =-۷
```

C=۱ , S=۱ , Z=۰ , V=۰

۱ ۱ ۱ ۱ ۰ ۰ ۱ =−۷ + ۰ ۱ ۱ ۱ =+۷ ① ۰ ۰ ۰ ۰ =۰ C=۱ , S=۰ , Z=۱ , V=۰	۱ ۱ ۱ ۰ ۱ =−۳ + ۰ ۱ ۰ ۰ =+۴ ① ۰ ۰ ۰ ۱ =+۱ C=۱ , S=۰ , Z=۰ , V=۰

حاصل جمع دو عدد منفی، یک عدد مثبت شده که غلط است و سرریز می‌باشد. در واقع حاصل جمع ۷- با ۶- برابر ۱۳- می‌باشد که با ۴ بیت قابل نمایش نیست.

۱
۱ ۰ ۰ ۱ =−۷
+ ۱ ۰ ۱ ۰ =−۶
① ۰ ۰ ۱ ۱ =+۳
C=۱ , S=۰ , Z=۰ , V=۱

حاصل جمع دو عدد مثبت، منفی شده که غلط است و سرریز می‌باشد. در واقع حاصل جمع ۵+ با ۴+ باید ۹+ شود که با ۴ بیت قابل نمایش نیست.

۱
۰ ۱ ۰ ۱ =+۵
+ ۰ ۱ ۰ ۰ =+۴
۱ ۰ ۰ ۱ =−۷
C=۰ , S=۱ , Z=۰ , V=۱

نکته

برای تشخیص سرریز در سیستم مکمل ۲ دو روش وجود دارد:

۱) اگر جمع دو عدد منفی، مثبت شود یا جمع دو عدد مثبت، منفی شود آنگاه سرریز است و برعکس. توجه کنید جمع عدد مثبت با عدد منفی، امکان سرریز ندارد.

۲) اگر رقم نقلی وارد شده به بیت سمت چپ با رقم نقلی خارج شده از آن (که همان فلگ C است) یکسان نباشد، سرریز است.

تست: دو عدد n بیتی $A=a_{n-1}a_{n-2}...a_0$ و $B=b_{n-1}b_{n-2}...b_0$ در سیستم مکمل ۲ با هم جمع شده‌اند. اگر رقم نقلی وارد شده به بیت سمت راست c_0 (معمولاً صفر است)، و رقم نقلی خارج شده از بیت سمت چپ c_n باشد، کدام گزینه به درستی فلگ V را نشان می‌دهد.

۱) $V=\bar{a}_{n-1}.\bar{b}_{n-1}.s_{n-1}+a_{n-1}.b_{n-1}.\bar{s}_{n-1}$
۲) $V=c_n\oplus c_{n-1}$
۳) $V=\bar{a}_{n-1}.\bar{b}_{n-1}.c_{n-1}+a_{n-1}.b_{n-1}.\bar{c}_{n-1}$
۴) هر سه

پاسخ: اگر جمع دو عدد منفی $(a_{n-1}=b_{n-1}=1)$ مثبت شود $(s_{n-1}=0)$ یا جمع دو عدد مثبت $(a_{n-1}=b_{n-1}=0)$ منفی شود $(s_{n-1}=1)$ آنگاه سرریز است و برعکس پس:

c_n	c_{n-1}	c_{n-2}		c_1	c_0
	a_{n-1}	a_{n-2}	...	a_1	a_0
+	b_{n-1}	b_{n-2}	...	b_1	b_0
	s_{n-1}	s_{n-2}		s_1	s_0

$$V=a_{n-1}.b_{n-1}.\bar{s}_{n-1}+\bar{a}_{n-1}.\bar{b}_{n-1}.s_{n-1}$$

اگر رقم نقلی وارد شده به بیت سمت چپ (c_{n-1}) با رقم نقلی خارج شده از آن (c_n) برابر نباشد آنگاه سرریز است و برعکس پس: $V = c_n \oplus c_{n-1}$

بررسی کنید هرگاه $a_{n-1} = b_{n-1} = 1$ و $c_{n-1} = 0$ یا $a_{n-1} = b_{n-1} = 0$ و $c_{n-1} = 1$ نیز سرریز است و برعکس پس

$$V = a_{n-1}.b_{n-1}.\bar{c}_{n-1} + \bar{a}_{n-1}.\bar{b}_{n-1}.c_{n-1}$$

■

برای انجام تفریق $A - B$ در سیستم مکمل ۲ دو روش مطرح است:

روش ۱: عدد اول را با مکمل ۲ عدد دوم جمع کنیم:

$$A - B = A + (\text{مکمل دو } B) = A + \bar{B} + 1$$

روش ۲: همانند تفریق قلم و کاغذ دو عدد را با قرض گرفتن تفریق کنیم.

مثال ۴: تفریق ۴ بیتی $(-6) - (-3) = 1101 - 1010$ را با دو روش انجام داده و وضعیت فلگ‌ها را مشخص کنید.

پاسخ: روش جمع:

$$\begin{array}{cccc} & 1 & & 1 \\ & 1 & 1 & 0 & 1 \\ + & 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 1 \end{array}$$

$1101 - 1010 = 1101 + 0101 + 1 = 0011$

$C = 1, Z = 0, S = 0, V = 0$

روش قرض: از آنجایی که چپ‌ترین بیت، نیاز به قرض ندارد پس $C = 0$. در واقع C نشان‌دهنده قرض است.

$$\begin{array}{cccc} & 0 & 10 & \\ 1 & \not{1} & \not{0} & 1 \\ - 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 \end{array}$$

$C=0 , Z=0 , S=0 , V=0$

مثال ۵: تفریق ۴ بیتی $(-6) - (-3) = 1010 - 1101$ را با دو روش انجام داده وضعیت فلگ‌ها را مشخص کنید.

پاسخ: روش جمع:

$1010 - 1101 = 1010 + 0010 + 1 = 1101$

$C = 0, Z = 0 , S = 1 , V = 0$

روش قرض:

$$\begin{array}{cccc} & 10 & & \\ C=1 \rightarrow & \not{1} & 10 & 10 \\ & \not{1} & \not{0} & \not{1} & \not{0} \\ - 1 & 1 & 0 & 1 \\ \hline 1 & 1 & 0 & 1 \end{array}$$

$C=1 , Z=0 , S=1 , V=0$

توجه: پرچم یا فلگ C در دو روش گفته شده برای تفریق، مکمل یکدیگر است. حتی برخی ماشین‌ها تفریق را با روش جمع انجام می‌دهند و فلگ C حاصل را مکمل کرده و شبیه روش قرض اعلام می‌کنند. سایر فلگ‌ها در دو روش یکسان است.

نکته

پس از انجام عمل تفریق $A-B$ با هر روشی، وضعیت فلگ‌ها طبق جدول زیر است:

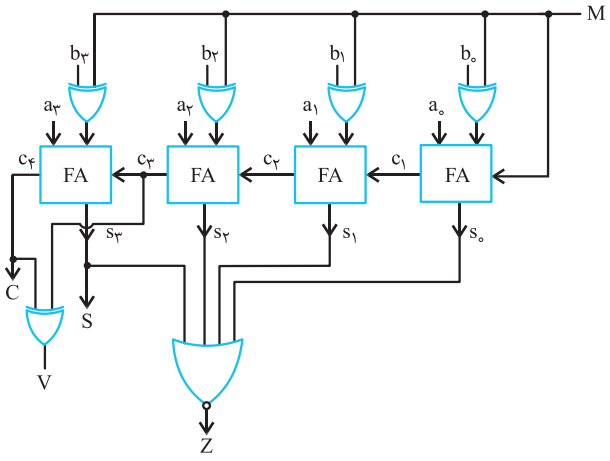
شرط	وضعیت فلگ‌ها
$A=B$	$Z=1$
$A\neq B$	$Z=0$
$A\geq B$	$[(S=0,V=0)\text{ or }(S=1,V=1)]\Leftrightarrow S=V\Leftrightarrow S\oplus V=0$
$A<B$	$[(S=1,V=0)\text{ or }(S=0,V=1)]\Leftrightarrow S\neq V\Leftrightarrow S\oplus V=1$

در حالت $A\geq B$ انتظار داریم که حاصل مثبت باشد $(S=0)$ ولی به شرطی که سرریز نباشد $(V=0)$. اگر سرریز باشد $(V=1)$ آنگاه حاصل منفی می‌شود $(S=1)$.

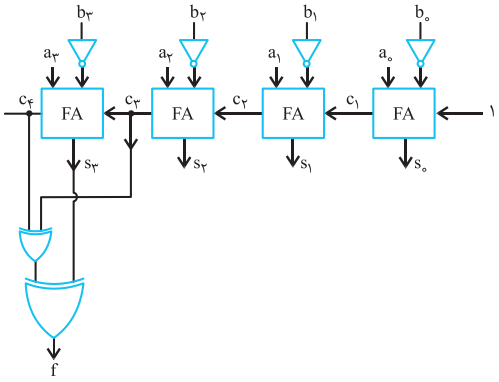
در حالت $A<B$ انتظار داریم حاصل منفی باشد $(S=1)$ به شرطی که سرریز نباشد $(V=0)$ حال اگر سرریز باشد $(V=1)$ حاصل مثبت می‌شود $(S=0)$.

شروط گفته شده در جدول، لازم و کافی هستند. در حالت $A<B$ می‌توان گفت $Z=0$. راجع به فلگ C نمی‌توان اظهارنظر کرد.

مثال ۶: سخت‌افزار جمع و تفریق ۴ بیتی $A\pm B$ به همراه تولید پرچم‌ها به شکل زیر است. در این شکل $A=a_3a_2a_1a_0$ و $B=b_3b_2b_1b_0$. هرگاه $M=0$ آنگاه $A+B$ و اگر $M=1$ آنگاه $A-B=A+\overline{B}+1$ انجام می‌شود.



مثال ۷: در شکل زیر اعداد $A = a_3a_2a_1a_0$ و $B = b_3b_2b_1b_0$ در سیستم مکمل ۲ هستند. اگر خروجی f یک شود، نشان‌دهنده چیست؟



پاسخ: در این شکل عمل $A + \overline{B} + ۱ = A - B$ انجام می‌شود و $f = S \oplus V$ که یک شده است یعنی $S \neq V$ پس $A < B$.

سیستم مکمل یک: هیچ ماشینی از این سیستم استفاده نمی‌کند. یادآوری می‌کنیم ارزش بیت‌ها در این سیستم همانند سیستم بی‌علامت است، با این تفاوت که ارزش بیت سمت چپ $-(۲^{n-۱} - ۱)$ است (برای عدد n بیتی). به عنوان مثال به تبدیلات زیر توجه کنید:

$۱۰۰۱ = -۷ + ۱ = -۶$

$۱۱۰۰۱ = -۱۵ + ۸ + ۱ = -۶$

$۱۱۱۰۰۱ = -۳۱ + ۱۶ + ۸ + ۱ = -۶$

$۱۱۱۱ = -۷ + ۴ + ۲ + ۱ = -۰$

$۱۱۱۱۱ = -۱۵ + ۸ + ۴ + ۲ + ۱ = -۰$

$۰۱۱۰ = ۴ + ۲ = +۶$

همانند سیستم مکمل ۲، اگر بیت سمت چپ گسترش یابد و تکرر شود، مقدار عدد عوض نمی‌شود. در این سیستم برای جمع دو عدد، آنها را به طور عادی جمع می‌کنیم و رقم نقلی خروجی (C) را با حاصل، جمع می‌کنیم. مثلاً به جمع‌های ۴ بیتی زیر توجه کنید:

۱ ۰ ۰ ۱ =-۶	۱ ۱ ۱ ۰ ۱ =-۲
+ ۰ ۰ ۱ ۰ =+۲	+ ۰ ۱ ۱ ۰ =+۶
۱ ۰ ۱ ۱ =-۴	C=① ۰ ۰ ۱ ۱
C=۰ , Z=۰ , S=۱ , V=۰	۱
	۰ ۱ ۰ ۰ =+۴
	C=۱ , Z=۰ , S=۰ , V=۰

برای تفریق $A-B$ در سیستم مکمل یک، عدد اول را با مکمل یک (NOT) عـدد دوم جمـع کرده و حاصل اگر رقم نقلی تولید کرد $(C=۱)$ باید با این رقم نقلی جمع شود. مثلاً به تفریق‌های ۴ بیتی زیر توجه کنید:

$(-۶)-(-۲)=۱۰۰۱-۱۱۰۱=۱۰۰۱+۰۰۱۰=۱۰۱۱=-۴$

$(-۲)-(-۶)=۱۱۰۱-۱۰۰۱=۱۱۰۱+۰۱۱۰=\textcircled{۱}۰۰۱۱+۱=۰۱۰۰=+۴$

نکات مربوط به فلگ‌ها و تشخیص سرریز در این سیستم همانند سیستم مکمل ۲ است.

سیستم بی‌علامت

همانطور که گفته شد، بزرگترین عدد n بیتی بی‌علامت، $۲^n-۱$ است. حاصل جمـع دو عـدد n بیتی حداکثر $n+۱$ بیتی می‌شود پس اگر دو عدد n بیتـی را جمـع کنـیم و حاصـل را در n بیـت ذخیره کنیم امکان سرریز وجود دارد و در صورتی سرریز است که از بیت nام رقم نقلی خارج شود یعنی $V=C$. مثلاً با ۴ بیت، جمع دو عدد ۹ و ۷ سرریز است:

$$\begin{array}{r} \textcircled{۱}\textcircled{۱}\textcircled{۱} \\ ۹=۱\ ۰\ ۰\ ۱ \\ +\ ۷=۰\ ۱\ ۱\ ۱ \\ \hline \textcircled{۱}\ ۰\ ۰\ ۰\ ۰ \end{array}$$

$C=۱ \longrightarrow V=۱$

برای تفریق دو عدد n بیتی بی‌علامت می‌توان دو روش پیشنهاد داد:

۱) عدد اول را با مکمل ۲ عدد دوم جمع کنیم یعنی:

$$A-B=A+\overline{B}+۱$$

مثلاً با ۴ بیت:

$۹-۳=۱۰۰۱-۰۰۱۱=۱۰۰۱+۱۱۰۰+۱=\textcircled{۱}۰۱۱۰=۶$

$C=۱$

با این روش اگر $C=۱$ یعنی $A\geq B$ و برعکس. و اگر $C=۰$ یعنی $A<B$ و برعکس.

۲) با روش قرض گرفتن عمل تفریق انجام شود. در این صورت فلـگ C نشـان‌دهنـده قـرض اسـت یعنی اگر $A<B$ آنگاه $C=۱$ و برعکس. و اگر $A\geq B$ آنگاه $C=۰$ و برعکس مثلاً

$۹-۳=۱۰۰۱-۰۰۱۱$

$$\begin{array}{r} \textcircled{۰}\textcircled{۱}\textcircled{۱۰} \\ -\ \not{۱}\ \not{۰}\ \not{۰}\ ۱ \\ \hline ۰\ ۰\ ۱\ ۱ \\ \hline ۰\ ۱\ ۱\ ۰ \end{array}$$

$C=۰$ چون بیت سمت چپ نیاز به قرض ندارد یعنی $A\geq B$ $(۹\geq ۳)$.

جمع و تفریق سیستم علامت مقدار

برخلاف سیستم‌های مکمل ۱ و ۲، جمع و تفریق سیستم علامت مقدار پیچیده است. فرض کنید a و b دو عدد n بیتی سیستم علامت مقدار هستند که A_S و B_S علامت این دو عدد و A و B ارزش این دو عدد است:

$a=$ | A_S | A | $\quad b=$ | B_S | B |

می‌خواهیم عمل $a \pm b \leftarrow a$ را انجام دهیم، عملِ بین a و b را OP می‌نامیم که OP می‌تواند جمع یا تفریق باشد. حالات مختلف را باید بررسی کنیم:

حالت ۱: $OP = +$ و $A_S \oplus B_S = ۰$: یعنی دو عدد هم علامت هستند و عمل، جمع است در این صورت علامت حاصل A_S است و عمل $A \leftarrow A + B$ انجام می‌شود. اگر رقم نقلی تولید شود یعنی سرریز $(V = C)$.

حالت ۲: $OP = -$ و $A_S \oplus B_S = ۱$: یعنی دو عدد مختلف‌العلامه هستند و عمل، تفریق است:

$(+A)-(-B)$ یا $(-A)-(+B)$

این حالت مشابه حالت ۱ می‌باشد.

حالت ۳: $OP = +$ و $A_S \oplus B_S = ۱$: یعنی دو عدد مختلف‌العلامه و عمل، جمع است.

$(-A)+(+B)$ یا $(+A)+(-B)$

در این حالت $A \leftarrow A - B = A + \overline{B} + ۱$ انجام می‌شود و اگر $C = ۱$ آنگاه $A \geq B$ و حاصل درست است و علامت حاصل A_S نیز درست است ولی اگر $C = ۰$ آنگاه $A < B$ یعنی حاصل باید مکمل ۲ شود $A \leftarrow \overline{A} + ۱$ و علامت حاصل عوض شود $A_S \leftarrow \overline{A}_S$.

حالت ۴: $OP = -$ و $A_S \oplus B_S = ۰$: یعنی دو عدد هم علامت هستند و عمل، تفریق است. مشابه حالت ۳ می‌باشد.

مثال ۸: با ۵ بیت عمل $(-۱۳)+(+۹)$ را انجام می‌دهیم:

$a = ۰۱۰۰۱ = +۹ \quad , \quad b = ۱۱۱۰۱ = -۱۳$

$A_S = ۰ \ , \ A = ۱۰۰۱ \ , \ B_S = ۱ \ , \ B = ۱۱۰۱ \ , \ OP = +$ (حالت ۳)

$A \leftarrow A - B \Rightarrow A \leftarrow ۱۰۰۱ - ۱۱۰۱ = ۱۰۰۱ + ۰۰۱۱ = ۱۱۰۰$

چون $C = ۰$ پس حاصل باید مکمل ۲ شود: $A \leftarrow \overline{A} + ۱ = ۰۱۰۰$ و علامت A_S عوض شود $A_S \leftarrow \overline{A}_S = ۱$ بنابراین نتیجه $a = ۱۰۱۰۰ = -۴$ می‌باشد.

توجه: به تست‌های ۱ تا ۱۵ پاسخ دهید.

سیستم بایاس یا افزونه: همانطور که دیدیم در این سیستم برای ذخیرهٔ عدد x، آن را با bias جمع کرده و $x + bias$ ذخیره می‌کنیم. مثلاً با ۴ بیت ۰۰۰۰ تا ۱۱۱۱ نشان‌دهنده $bias - ۰$ تا $bias - ۱۵$ هستند. مقادیر متداول برای bias با n بیت عبارتند از $۲^{n-۱}$ و $۲^{n-۱} - ۱$. پس با ۴ بیت $bias = ۸$ یا $bias = ۷$. پس اگر $bias = ۸$ آنگاه ۰۰۰۰ یعنی ۸− و ۱۱۱۱ یعنی ۷+ = ۸ − ۱۵. اگر $bias = ۷$ آنگاه ۰۰۰۰ یعنی ۷− و ۱۱۱۱ یعنی ۸+ = ۷ − ۱۵. جمع و تفریق در این سیستم همانند سیستم بی‌علامت است ولی پس از جمع باید حاصل منهای bias شود و پس از تفریق باید حاصل با bias جمع شود. اگر $bias = ۲^{n-۱}$ باشد جمع یا تفریق با مقدار bias به این معنی است که بیت سمت چپ حاصل را عوض کنیم. مثلاً ۵+ = ۱۱۰۱ حال اگر $bias = ۸$ را از این عدد کم کنیم ۳− = ۰۱۰۱ حاصل می‌شود یا ۴− = ۰۱۰۰ حال اگر $bias = ۸$ را به این عدد اضافه کنیم ۴+ = ۱۱۰۰ حاصل می‌شود. ضرب و تقسیم در سیستم بایاس دشوار است برای همین تنها کاربرد این سیستم، نمایش توان اعداد ممیز شناور است که هیچگاه نیاز به ضرب و تقسیم ندارند.

۴- روش‌های ضرب مبنای ۲

می‌خواهیم ضرب دو عدد در مبنای ۲ در سیستم‌های مختلف بررسی کنیم. در عمل ضرب $B \times Q$ به عدد اول (B) ضرب شونده یا مضروب (multiplicand) و به عدد دوم (Q) ضرب‌کننده یا مضروب فیه (multiplier) گویند.

ضرب اعداد بی‌علامت: همانند ضرب اعداد مبنای ۱۰ می‌باشد، یعنی ارقام ضرب‌کننده را در ضرب‌شونده ضرب می‌کنیم و هر بار حاصل‌ضرب جزئی را شیفت به چپ داده و با حاصل قبلی جمع می‌کنیم.

```
      ۱ ۱ ۰ ۱   (multiplicand ۱۳)
×     ۱ ۰ ۱ ۱   (multiplier   ۱۱)
-------------
        ① ۱ ۱ ۰ ۱
    + ① ۱ ۱ ۰ ۱ ۰   (partial products)
      ① ۰ ۰ ۰ ۰ ۰ ۰   حاصلضرب‌های جزئی
  ① ۱ ۱ ۰ ۱ ۰ ۰ ۰
-------------
  ۱ ۰ ۰ ۰ ۱ ۱ ۱ ۱   (حاصل نهایی product)
```

در مبنای ۲، عمل ضرب بسیار ساده است. بیت‌های ضرب‌کننده را از سمت راست بررسی می‌کنیم و هر بیت که ۱ بود، ضرب‌شونده را به چپ شیفت داده می‌نویسیم و اگر بیت ضرب‌کننده

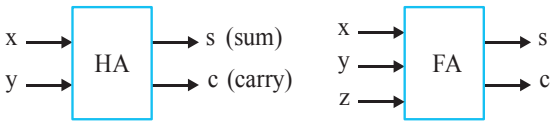
صفر بود، صفر می‌نویسیم. علت اینکه شیفت چپ می‌دهیم، ارزش مکانی بیت‌های ضرب‌کننده است:

$$۱۱۰۱\times۱۰۱۱=۱۱۰۱\times(۲^۳+۲^۱+۲^۰)=۱۱۰۱\times۲^۳+۱۱۰۱\times۲^۱+۱۱۰۱\times۲^۰$$
$$=۱۱۰۱۰۰۰+۱۱۰۱۰+۱۱۰۱=۱۰۰۰۱۱۱۱$$

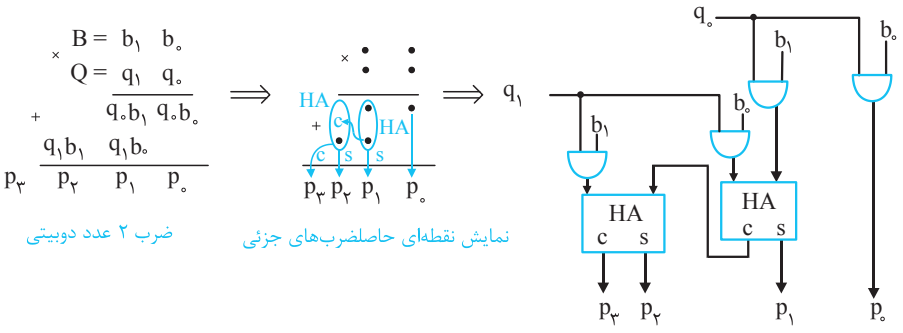
حاصلضرب دو عدد m و n بیتی حداکثر $m+n$ بیتی است زیرا ماکزیمم عدد m و n و $m+n$ بیتی به ترتیب $۲^m-۱$ و $۲^n-۱$ و $۲^{m+n}-۱$ است و به راحتی می‌توان نشان داد $(۲^m-۱)(۲^n-۱)\geq ۲^{m+n}-۱$ یعنی حاصل ضرب دو عدد m و n بیتی از $m+n$ بیت بیشتر نمی‌شود.

اگر برای جمع حاصلضرب‌های جزئی از مدارات ترکیبی استفاده کنیم اصطلاحاً روش ضرب را آرایه‌ای (array multiplier) گویند. برای ضرب آرایه‌ای مدل‌های مختلفی وجود دارد.

می‌توان با کمک گیت‌های AND و تعدادی HA (نیم جمع‌کننده Half Adder) و FA (تمام جمع‌کننده Full Adder) عمل ضرب را به صورت ترکیبی انجام داد. یادآوری می‌کنیم که HA دو بیت را جمع کرده حاصل جمع و رقم نقلی تولید می‌کند. FA سه بیت را جمع می‌کند، حاصل جمع و رقم نقلی تولید می‌کند:



به عنوان مثال برای ضرب دو عدد ۲ بیتی به ۴ گیت AND و ۲ عدد HA نیاز است:

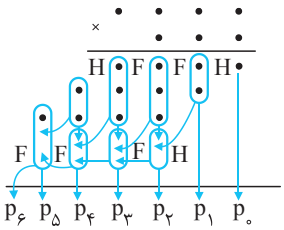


ضرب ۲ عدد دوبیتی

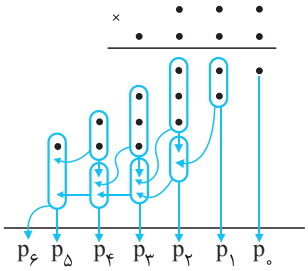
نمایش نقطه‌ای حاصلضرب‌های جزئی

پیاده‌سازی ضرب به صورت ترکیبی

بررسی کنید برای ضرب دو عدد m و n بیتی با روش فوق، نیاز به $m\times n$ گیت AND و $\min\{m,n\}$ نیم جمع‌کننده و $m\times n-(m+n)$ تمام جمع‌کننده است. شکل زیر، ضرب ۴×۳ و ۳×۴ را به صورت نقطه‌ای نشان داده است. در هر دو حالت ۱۲ گیت AND و ۳ عدد HA و ۵ عدد FA استفاده شده است:



ضرب ۴×۳

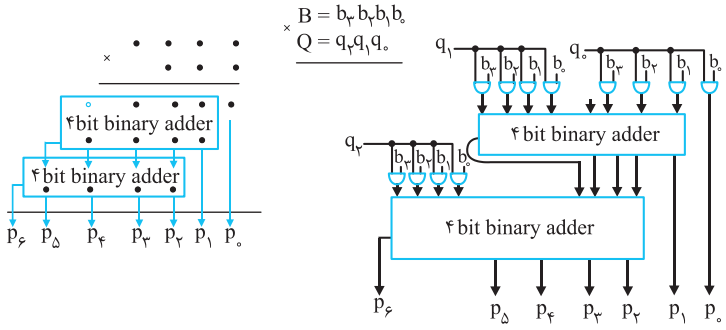


ضرب ۳×۴

نتیجه: ضرب $n\times n$ نیاز به $n^۲$ گیت AND و n عدد HA و $n^۲-۲n$ عدد FA دارد.

نتیجه: اگر HA وجود نداشت آنگاه ضرب $n\times n$ نیاز به $n^۲$ گیت AND و $n^۲-n$ عدد FA دارد.

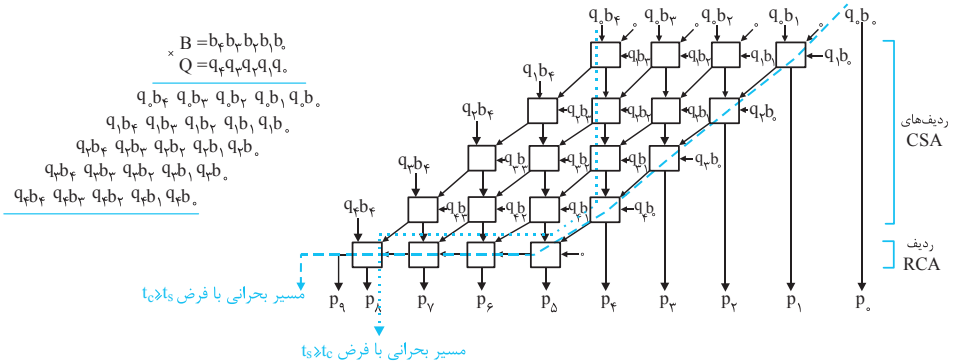
حاصلضرب‌های جزئی را می‌توان با کمک جمع‌کنندهٔ دودویی چندبیتی نیز با هم جمع کرد. در این صورت برای ضرب آرایه‌ای $m_{bit}\times n_{bit}$ نیاز به $m\times n$ گیت AND و $n-۱$ جمع‌کنندهٔ m بیتی است. شکل زیر ضرب ۴×۳ را نشان می‌دهد:



ضرب ۴×۳ - با ۱۲ گیت AND و دو جمع کننده ۴ بیتی

تاکنون ضرب آرایه‌ای را با دو مدل بررسی کردیم. مدل سوم ضرب آرایه‌ای $m\times n$ به این صورت است که $m\times n$ عدد گیت AND استفاده کنیم و برای جمع حاصلضرب‌های جزئی n ردیف قرار دهیم که در هر ردیف $m-۱$ عدد FA باشد طوری که در هر ردیف تمام جمع‌کننده‌ها به یکدیگر رقم نقلی ندهند البته به جز ردیف آخر. اصطلاحاً به ردیف‌هایی که تمام جمع‌کننده‌ها به

هم وابسته نیستند، CSA(Carry Save Adder) گویند و به ردیف آخر که تمام جمع‌کننده‌ها بـه یکدیگر رقم نقلی می‌دهند RCA(Ripple Carry Adder) گویند. شکل زیر ضرب ۵×۵ را با این مدل نشان می‌دهد. همانطور که مشاهده می‌کنید ۵ ردیـف کـه در هـر ردیـف ۴ عـدد FA اسـت. گیت‌های AND نشان داده نشده‌اند.

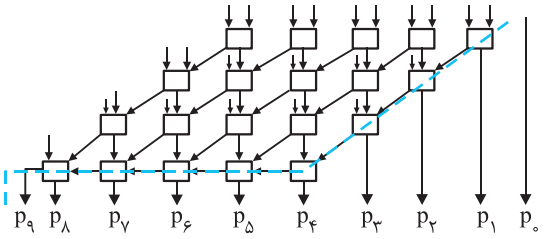


روش CSA برای ضرب آرایه‌ای از نظر تأخیر بهترین است. اگـر تـأخیر تولیـد sum و carry را در FA به ترتیب t_s و t_c فرض کنیم آنگاه اگر $t_c \geq t_s$، تأخیر شـکل فـوق برابـر $t_{AND} + ۸t_c$ است که t_{AND} تأخیر گیت AND است. دقت کنیـد بـرای محاسـبه تـأخیر بایـد مسـیر بحرانـی یعنی مسیری که بیشترین تأخیر را دارد مشـخص کنـیم در شـکل فـوق مسـیر بحرانـی بـا فـرض $t_c \geq t_s$ به صورت ..... نشـان داده شـده اسـت. اگـر $t_s \geq t_c$ آنگـاه تـأخیر شـکل فـوق برابـر $t_{AND} + ۴t_s + ۴t_c$ است که مسیر بحرانی ایـن حالـت بـه صـورت – – – – – نشـان داده شـده است.

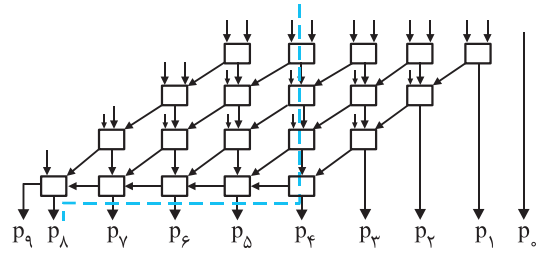
به طور کلی تاخیر ضرب آرایه‌ای $m \times n$ که به صورت $n-۱$ ردیف CSA و یک ردیف RCA که در هر ردیف $m-۱$ عدد FA است، با فرضیات مختلف برابر است با:

$t_s \leq t_c$	$t_s \geq t_c$	
$t_{AND} + (m+n-۲)t_c$	$t_{AND} + n.t_s + (m-۲)t_c$	$m > n$
$t_{AND} + (m-n-۲)t_c$	$t_{AND} + (m-۱)t_s + (n-۱)t_c$	$m \leq n$

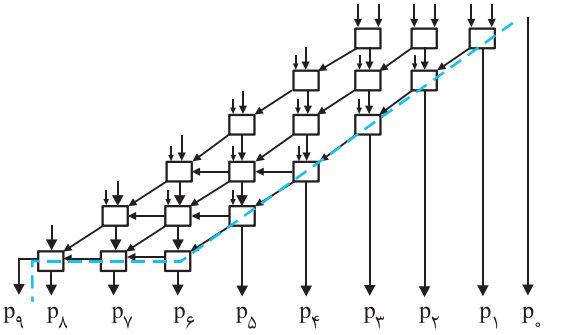
برای درک بهتر درستی این روابط در شکل‌های زیر ضرب ۴×۶ و ۶×۴ انجام شده و با فرضیات مختلف مسیر بحرانی نشان داده شده است:



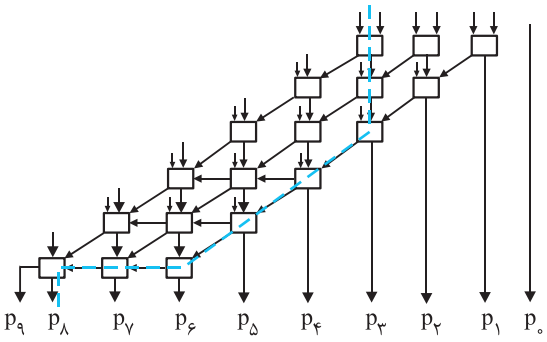
ضرب ۴×۶، مسیر بحرانی با فرض $t_S<t_C$،
تاخیر برابر $t_{AND}+۸t_C$



ضرب ۴×۶، مسیر بحرانی با فرض $t_S>t_C$،
تاخیر برابر $t_{AND}+۴t_S+۴t_C$

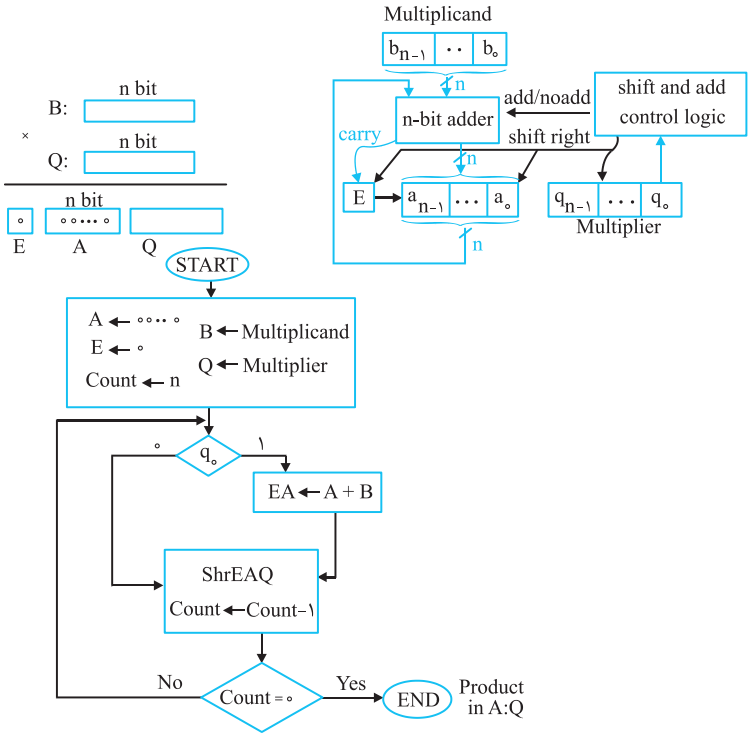


ضرب ۶×۴، مسیر بحرانی با فرض $t_S<t_C$،
تاخیر برابر $t_{AND}+۸t_C$



ضرب ۶×۴، مسیر بحرانی با فرض $t_S>t_C$،
تاخیر برابر $t_{AND}+۳t_S+۵t_C$

تاکنون ضرب آرایه‌ای را بررسی کردیم که از نظر سرعت بهترین ولی از نظر سخت‌افزار پرهزینه‌ترین روش است. در ماشین‌ها ضرب به صورت ترتیبی انجام می‌شود به این صورت که برای ضرب $n \times n$ فقط یک جمع‌کنندهٔ n بیتی (و نه $n-1$ جمع‌کنندهٔ n بیتی) استفاده می‌شود و حاصلضرب‌های جزئی به ترتیب و با کلاک با هم جمع می‌شوند. فرض کنید می‌خواهیم ضرب دو عدد n بیتی B و Q را انجام دهیم و حاصل را در جفت ثبات $A:Q$ قرار دهیم یعنی $A:Q \leftarrow B \times Q$. ثبات‌های B و Q و A هر کدام n بیتی هستند و مقدار اولیهٔ A صفر است. دقت کنید پس از پایان عمل ضرب، ضرب‌کننده (Q) از بین رفته و نیمهٔ پایین حاصل ضرب در آن قرار می‌گیرد. روش ضرب ترتیبی یا اصطلاحاً روش Add & shift به این صورت است که n بار بیت سمت Q را بررسی می‌کنیم (نام این بیت q_0 است) اگر $q_0 = 1$ آنگاه $EA \leftarrow A + B$ (یعنی A و B جمع شده حاصل در A و رقم نقلی در E قرار می‌گیرد. E یک فلیپ فلاپ با مقدار اولیهٔ صفر است) و سپس ShrEAQ (یعنی EAQ را با هم به راست شیفت می‌دهیم). ولی اگر $q_0 = 0$ آنگاه فقط عمل ShrEAQ انجام می‌شود. علت اینکه EAQ را شیفت راست می‌دهیم این است که حاصل ضرب‌های جزئی به درستی و مشابه همان روش قلم و کاغذ با هم جمع شوند. در ضمن با هر شیفت راست، Q بیت سمت راستش را از دست می‌دهد.



سخت افزار و فلوچارت ضرب روش add & shift برای اعداد بی علامت
$A:Q \leftarrow B \times Q$

مثال ۹: فرض کنید ثبات‌ها ۴ بیتی هستند. با روش add&shift حاصلضرب ۱۱×۱۳ را بدست آورید.

پاسخ: $(1101)_2 = 13$ و $(1011)_2 = 11$ پس $B = 1101$ و $Q = 1011$ (بیت سمت راست Q یعنی $q_0 = 1$) و $A = 0000$ و $E = 0$ و $count = 4$ پس باید ۴ بار عملیات را انجام دهیم:

		E	A	Q	
۱) $q_0 = 1 \Rightarrow$	$EA \leftarrow A + B$	۰	۱۱۰۱	۱۰۱۱	
	ShrEAQ	۰	۰۱۱۰	۱۱۰۱	q_0
۲) $q_0 = 1 \Rightarrow$	$EA \leftarrow A + B$	۱	۰۰۱۱	۱۱۰۱	
	ShrEAQ	۰	۱۰۰۱	۱۱۱۰	q_0
۳) $q_0 = 0 \Rightarrow$	shr EAQ	۰	۰۱۰۰	۱۱۱۱	q_0
۴) $q_0 = 1 \Rightarrow$	$EA \leftarrow A + B$	۱	۰۰۰۱	۱۱۱۱	
	shrEAQ	۰	۱۰۰۰	۱۱۱۱	

حاصل ضرب

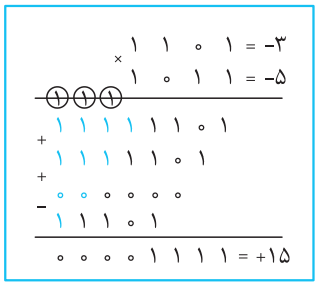
پس نتیجه $AQ = (10001111)_2 = 143$ است. در این مثال ۴ عمل شیفت و ۳ عمل جمع انجام شد.

نتیجه: روش Add & shift به تعداد بیت‌های ضرب‌کننده عمل شیفت و به تعداد یک‌های ضرب‌کننده عمل جمع انجام می‌دهد. پس اگر شیفت و جمع در دوکلاک مجزا انجام شوند، این روش به تعداد بیت‌های ضرب‌کننده بعلاوهٔ یک‌های ضرب‌کننده، کلاک نیاز دارد. البته می‌توان سخت‌افزار را طوری پیاده‌سازی کرد که جمع و شیفت در یک کلاک انجام شود که در این صورت تعداد کلاک این روش به تعداد بیت‌های ضرب‌کننده است.

ضرب در سیستم علامت مقدار: در این سیستم کافی است علامت دو عدد را XOR کرده که علامت حاصل بدست آید و سپس ارزش‌ها را مشابه روش‌های ضرب دو عدد بی‌علامت درهم ضرب کنیم.

ضرب در سیستم مکمل ۲: برای ضرب دو عدد سیستم مکمل ۲ می‌توان همانند سیستم بی‌علامت عمل کرد با این تفاوت که بیت سمت چپ حاصلضرب‌های جزئی را گسترش دهیم و همچنین آخرین حاصلضرب جزئی را منها کنیم. به ضرب $(-5)\times(-3)$ با ۴ بیت توجه کنید:

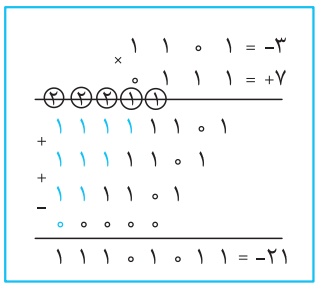
ضرب دو عدد ۴ بیتی حداکثر ۸ بیتی است پس حاصلضرب‌های جزئی را تا ۸ بیت گسترش داده‌ایم. توجه کنید آخرین حاصلضرب جزئی تفریق شده است البته می‌توان آن را مکمل ۲ کرده و همانند سایر حاصلضرب‌های جزئی، جمع کرد.



علت اینکه آخرین حاصلضرب جزئی تفریق می‌شود این است که بیت سمت چپ ضرب‌کننده ارزش منفی دارد:

$$۱۱۰۱\times۱۰۱۱=۱۱۰۱\times(-۲^{۳}+۲^{۱}+۲^{۰})=-۱۱۰۱\times۲^{۳}+۱۱۰۱\times۲^{۱}+۱۱۰۱\times۲^{۰}$$

اگر ضرب‌کننده مثبت باشد، آخرین حاصلضرب جزئی صفر می‌شود و جمع یا تفریق کردن آن تفاوتی ندارد به ضرب $(+۷)\times(-۳)$ با ۴ بیت توجه کنید:



آخرین حاصلضرب جزئی صفر است که می‌توان جمع یا تفریق کرد. دقت کنید ۱+۱+۱+۱ برابر ۴ یعنی ۰ sum ← ۱۰ carry می‌باشد که ۱۰ یعنی ۲ واحد نقلی است. یا ۱+۱+۱+۱+۱ برابر ۵ یعنی ۱ sum ← ۱۰ carry می‌باشد.

می‌توان روش ضرب گفته شده را به صورت ترتیبی Add & shift پیاده‌سازی کرد فقط باید از شیفت راست حسابی (arithmetic) استفاده شود (ashr) یعنی از سمت چپ، بیت علامت وارد شود (بیت سمت چپ تکرار شود) و همچنین فلیپ فلاپ E نیاز نیست و از ارقام نقلی صرفنظر می‌شود. در ضمن آخرین حاصلضرب جزئی باید تفریق شود.

مثال ۱۰: فرض کنید ثبات‌ها ۴ بیتی هستند با روش ترتیبی حاصلضرب $(-۵)\times(-۳)$ را بیابید.

پاسخ: $(۱۱۰۱)_{۲}=-۳$ و $(۱۰۱۱)_{۲}=-۵$ پس $B=۱۱۰۱$ و $Q=۱۰۱۱$ $(q_{۰}=۱)$ و $A=۰۰۰۰$ و باید ۴ بار عملیات انجام شود و بار چهارم عمل تفریق انجام دهیم:

		A	Q
۱) $q_{۰}=۱\Rightarrow$	$A\leftarrow A+B$	۱۱۰۱	۱۰۱۱
	ashr AQ	۱۱۱۰	۱۱۰۱ ← $q_{۰}$
۲) $q_{۰}=۱\Rightarrow$	$A\leftarrow A+B$	۱۰۱۱	۱۱۰۱
	ashrAQ	۱۱۰۱	۱۱۱۰ ← $q_{۰}$
۳) $q_{۰}=۰\Rightarrow$	ashr AQ	۱۱۱۰	۱۱۱۱ ← $q_{۰}$
۴) $q_{۰}=۱\Rightarrow$	$A\leftarrow A-B$	۰۰۰۱	۱۱۱۱
	ashrAQ	۰۰۰۰	۱۱۱۱

پس حاصل $AQ=۰۰۰۰۱۱۱۱$ یعنی ۱۵+ می‌باشد.

لازم به ذکر است برای ضرب $B\times Q$ در سیستم مکمل ۲ می‌توان پیشنهادات دیگری نیز داد. یک پیشنهاد این است که هم ضرب شونده و هم ضرب‌کننده را مثبت کرده و ضرب را با شیوهٔ ضرب بی‌علامت انجام دهیم و حاصل را در صورت لزوم قرینه (مکمل ۲) کنیم. پیشنهاد دیگر این که اگر Q منفی بود آن را مکمل ۲ کرده که مثبت شود سپس ضرب را با گسترش علامت انجام دهیم و همه حاصلضرب‌های جزئی را جمع کنیم و در نهایت، حاصل را نیز مکمل ۲ کنیم. پیشنهاد بهتر این است که اگر Q منفی بود، هم B و هم Q را مکمل ۲ کنیم و سپس ضرب را با گسترش علامت انجام دهیم و همهٔ حاصلضرب‌های جزئی را جمع کنیم و حاصل بدست آمده درست است.

در سیستم مکمل ۲ به جز پیشنهادات و روش گفته شده، روشی موسوم به **بوث** (booth) وجود دارد. روش بوث براین اساس استوار است که یک‌های متوالی Q از مرتبه ۲^m تا مرتبه ۲^k را می‌توان به صورت $۲^{k+۱}-۲^m$ نوشت. مثلاً:

$$۱۴=۰\,\overset{۲^۴}{\textcircled{۰}}\,\overset{۲^۳}{\textcircled{۱}}\,۱\,\overset{۲^۱}{\textcircled{۱}}\,۰=۲^۴-۲^۱=۰\,\overset{۲^۴}{\textcircled{۱}}\,۰۰۰۰-۰۰۰۰\,\overset{۲^۱}{\textcircled{۱}}\,۰$$

به تساوی‌های زیر دقت کنید:

$$۰۱۱۰۱۱۱۰=۰۱۱۰۰۰۰۰+۰۰۰۰۱۱۱۰=(۲^۷-۲^۵)+(۲^۴-۲^۱)=-۲^۱+۲^۴-۲^۵+۲^۷$$

$$۱۱۰۰۱۱=۱۱۰۰۰۰+۰۰۰۰۱۱=(-۲^۴)+(۲^۲-۲^۰)=-۲^۰+۲^۲-۲^۴$$

به طور کلی برای تبدیل یک عدد سیستم مکمل ۲ به مبنای ۱۰ می‌توان از سمت راست عدد حرکت کرد و هر جا از ۰ به ۱ رسیدیم، ارزش آن ۱ را با علامت منفی بنویسیم و هرجا از ۱ به ۰ رسیدیم، ارزش آن ۰ را با علامت مثبت بنویسیم. البته ابتدا یک صفر بی‌ارزش سمت راست عدد قرار دهیم:

$$۱\,۱\,۰\,۰\,۱\,۱=۱\,\underset{-}{۱}\,۰\,\underset{+}{۰}\,۱\,\underset{-}{۱}\,\boxed{۰}\overset{\leftarrow}{=}-۲^۰+۲^۲-۲^۴$$

$$۱\,\underset{-}{۱}\,\underset{+}{۰}\,۱\,۱\,\underset{-}{۱}\,۰\,\boxed{۰}\overset{\leftarrow}{=}-۲^۱+۲^۴-۲^۵$$

$$\underset{+}{۰}\,۱\,\underset{-}{۱}\,\underset{+}{۰}\,۱\,\boxed{۰}\overset{\leftarrow}{=}-۲^۰+۲^۱-۲^۲+۲^۴$$

حال به بیان روش بوث می‌پردازیم. فرض کنید می‌خواهیم عمل $B\times ۰۱۱۱۰$ را انجام دهیم، تاکنون این عمل را به صورت زیر انجام می‌دادیم:

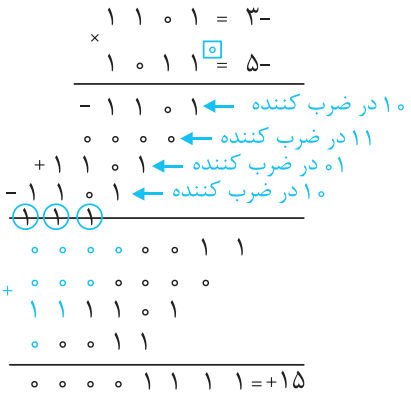
$$B\times ۰۱۱۱۰=B\times(۲^۳+۲^۲+۲^۱)=B\times ۲^۳+B\times ۲^۲+B\times ۲^۱$$

ولی روش بوث، ضرب‌کننده را به شکلی که بحث شد، بسط می‌دهد:

$$B\times ۰۱۱۱۰=B\times(-۲^۱+۲^۴)=-B\times ۲^۱+B\times ۲^۴$$

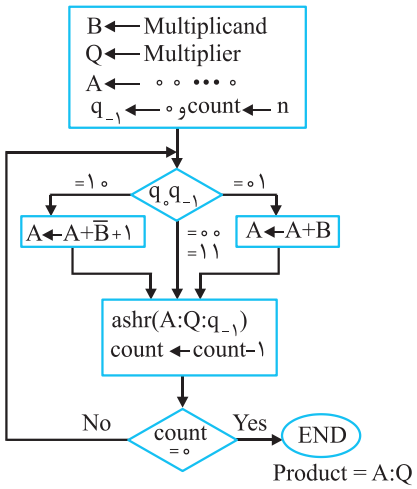
مثال ۱۱: ضرب ۵− × ۳− را با ۴ بیت با روش بوث انجام دهید.

پاسخ: سمت راست ضرب‌کننده یک صفر بی‌ارزش قرار می‌دهیم، سپس از سمت راست ضرب‌کننده حرکت می‌کنیم اگر الگوی ۱۰ دیدیم، ضرب شونده را با علامت منفی می‌نویسیم، اگر الگوی ۰۱ دیدیم ضرب شونده را با علامت مثبت می‌نویسیم. اگر الگوی ۰۰ یا ۱۱ دیدیم، صفر می‌نویسیم. البته با هر حرکت به سمت چپ، شیفت نیز می‌دهیم. سپس حاصلضرب‌های جزئی که دارای علامت منفی هستند را مکمل ۲ کرده و همهٔ حاصلضرب‌های جزئی را گسترش علامت می‌دهیم و همهٔ حاصلضرب‌های جزئی را جمع می‌کنیم.



■

حال می‌خواهیم ضرب بوث را به صورت ترتیبی پیاده‌سازی کنیم. ضرب شونده B، ضرب‌کننده Q و حاصل $A:Q$ می‌باشد یعنی $A:Q \leftarrow B \times Q$. ثبات‌ها (رجیسترها) همگی n بیتی هستند. نیازی به فلیپ فلاپ E برای نگهداری رقم نقلی نیست زیرا در عملیات سیستم مکمل ۲ از رقم نقلی صرفنظر می‌شود. بیت سمت راست Q را q_0 می‌نامیم و یک بیت به اسم q_{-1} سمت راست Q با مقدار اولیهٔ صفر تعریف می‌کنیم. الگوریتم به این صورت است که n بار (به تعداد بیت‌های Q) مقدار q_0q_{-1} را بررسی می‌کنیم، اگر ۱۰ باشد باید عمل تفریق یعنی $A \leftarrow A - B$ انجام شود، اگر ۰۱ باشد باید عمل جمع یعنی $A \leftarrow A + B$ انجام شود، اگر ۰۰ یا ۱۱ باشد هیچ عملی نباید انجام شود. پس از انجام جمع یا تفریق یا هیچ عمل باید A و Q و q_{-1} شیفت حسابی به راست $(\text{ashr } A:Q:q_{-1})$ داده شوند. فلوچارت روش بوث به شکل روبرو است:



مثال ۱۲: حاصلضرب ۵− × ۳− را با روش ترتیبی بوث و ثبات‌های ۴ بیتی بیابید.

پاسخ: $B = ۱۱۰۱$ و $Q = ۱۰۱۱$ $(q_0 = ۱)$ و $q_{-1} = ۰$ و $A = ۰۰۰۰$ و ۴ بار باید عملیات انجام شود: (مکمل دوی B برابر $\bar{B} + ۱ = ۰۰۱۱$ می‌باشد)

		A	Q	q_{-1}
۱) $q_0q_{-1}=10$	$A\leftarrow A+\overline{B}+1$	۰۰۱۱	۱۰۱۱	۰
	ashr AQq_{-1}	۰۰۰۱	۱۱۰۱	۱
۲) $q_0q_{-1}=11$	ashrAQq_{-1}	۰۰۰۰	۱۱۱۰	۱
۳) $q_0q_{-1}=01$	$A\leftarrow A+B$	۱۱۰۱	۱۱۱۰	۱
	ashrAQq_{-1}	۱۱۱۰	۱۱۱۱	۰
۴) $q_0q_{-1}=10$	$A\leftarrow A+\overline{B}+1$	۰۰۰۱	۱۱۱۱	۰
	ashrAQq_{-1}	۰۰۰۰	۱۱۱۱	۱

در نتیجه حاصل $+15=00001111=AQ$ می‌باشد.

نکته

۱- تعداد شیفت، تعداد جمع، تعداد تفریق در ضرب ترتیبی بوث به ترتیب برابر تعداد بیت‌ها، تعداد ۰۱ها، تعداد ۱۰ها در ضرب‌کننده است. فقط یادمان باشد ابتدا یک صفر بی‌ارزش سمت راست ضرب‌کننده قرار دهیم.

۲- اگر عمل تفریق با جمع انجام شود آنگاه هر عمل تفریق خود شامل یک عمل مکمل‌گیری و یک عمل جمع است (توجه کنید $A+\overline{B}+1$ شامل یک عمل جمع است) در این صورت تعداد عمل مکمل‌گیری برابر تعداد ۱۰ها و تعداد عمل جمع برابر تعداد ۱۰ها بعلاوه تعداد ۰۱ها در ضرب‌کننده است.

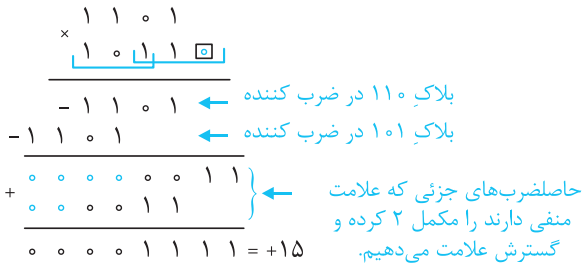
۳- از ضرب بوث می‌توان برای ضرب اعداد بی‌علامت نیز استفاده کرد مشروط بر اینکه بیت سمت چپ ضرب‌کننده حتماً صفر باشد و اگر چنین نبود یک صفر سمت چپ ضرب‌کننده قرار دهیم.

تسریع ضرب بوث: روش بوث هر بار ۲ بیت از ضرب‌کننده (Q) را بررسی می‌کند. ولی می‌توان با بررسی ۳ بیت از ضرب‌کننده تعداد جمع و تفریق‌ها را به نصف کاهش داد. روش سریع به این صورت است که ابتدا یک صفر بی‌ارزش سمت راست ضرب‌کننده قرار دهید سپس بیت‌های ضرب‌کننده را از سمت راست به بلاک‌های ۳ بیتی تقسیم کنید طوری که هر بلاک با بلاک قبلی فقط در یک بیت مشترک باشد مثلاً اگر ضرب‌کننده ۱۰۱۱ می‌باشد، بلاک‌های ۳ بیتی به صورت $\boxed{1}\boxed{011}\boxed{0}$ می‌باشند. سپس براساس این سه بیت طبق جدول مقابل یک عمل مناسب انجام دهید. فقط هر بار ضرب شونده (B) را ۲ واحد به چپ شیفت داده و با حاصل ضرب‌های جزئی قبلی جمع کنید.

علت	عملی که باید انجام شود	مقدار ۳ بیت بلاک
رشته‌ای از صفرها نیاز به عملی ندارد	None	۰ ۰ ۰
همانند بوث معمولی ۰۱ نیاز به جمع دارد	+B	۰ ۰ ۱
۱۰ یعنی B− و ۰۱ بعدی یعنی ۲B+ نتیجه B+	+B	۰ ۱ ۰
۱۱ یعنی هیچ و ۰۱ بعدی یعنی ۲B+ نتیجه ۲B+	+۲B	۰ ۱ ۱
۰۰ یعنی هیچ و ۱۰ بعدی یعنی ۲B− نتیجه ۲B−	−۲B	۱ ۰ ۰
۰۱ یعنی B+ و ۱۰ بعدی یعنی ۲B− نتیجه B−	−B	۱ ۰ ۱
۱۰ یعنی B− و ۱۱ بعدی یعنی هیچ نتیجه B−	−B	۱ ۱ ۰
رشته‌ای از یک‌ها نیاز به عملی ندارد.	None	۱ ۱ ۱

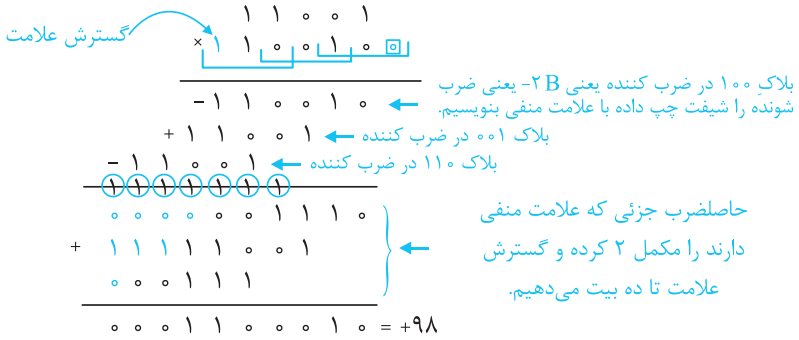
مثال ۱۲: حاصل ۵− × ۳− را با روش تسریع یافته بوث و ثبات‌های ۴ بیتی بیابید.

پاسخ:



توجه کنید اگر تعداد بیت ضرب‌کننده طوری بود که آخرین بلاک ۳ بیتی نشد، ضرب‌کننده را گسترش علامت دهید که همهٔ بلاک‌ها ۳ بیتی شوند.

مثال ۱۳: حاصلضرب ۱۴− × ۷− را با روش تسریع یافته بوث و ثبات‌های ۵ بیتی بیابید.



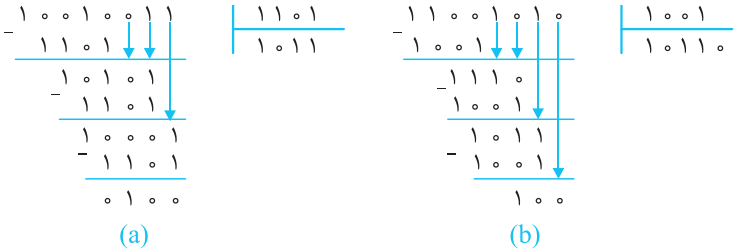
۵- روش‌های تقسیم مبنای ۲

فرض کنید A و B و Q رجیسترهای n بیتی هستند. مقسوم A : Q، مقسوم‌علیه B است یعنی می‌خواهیم A : Q را به B تقسیم کنیم. خارج قسمت در Q و باقی‌مانده در A قرار خواهند گرفت:

$$\begin{array}{l|l} A:Q & B \\ \vdots & Q \\ \hline A & \end{array}$$

پس در واقع می‌خواهیم ۲n بیت را به n بیت تقسیم کنیم و خارج قسمت و باقی‌مانده را در n بیت ذخیره کنیم.

تقسیم بی‌علامت: در عمل تقسیم، بیت‌های مقسوم را از سمت چپ یکی‌یکی جدا می‌کنیم تا وقتی که مقدار عددی تعداد بیتی که جدا کرده‌ایم از مقسوم‌علیه بیشتر یا مساوی باشد که در این صورت در خارج قسمت ۱ قرار می‌دهیم و مقسوم‌علیه را از تعداد بیتی که از مقسوم جدا کرده‌ایم کم می‌کنیم. از این به بعد به ازای هر بیت که از مقسوم در نظر می‌گیریم، یک بیت در خارج قسمت قرار می‌دهیم به تقسیم‌های ۸ بیت به ۴ بیت زیر دقت کنید.



در تقسیم (a) خارج قسمت ۴ بیتی است ولی در قسمت (b) خارج قسمت ۵ بیتی شده است. پس اگر رجیسترها در این دو مثال ۴ بیتی باشند، قسمت (b) سرریز است.

نکته

۱- در تقسیم ۲n بیت به n بیت اگر خارج قسمت در n بیت ذخیره شود در این صورت اگر n بیت با ارزش مقسوم از مقسوم‌علیه بزرگتر یا مساوی باشد آنگاه سرریز است و برعکس.

۲- در تقسیم m بیت به n بیت اگر خارج قسمت در k بیت ذخیره شود در این صورت اگر $m-k$ بیت با ارزش مقسوم از مقسوم علیه بیشتر یا مساوی باشد سرریز است و برعکس.

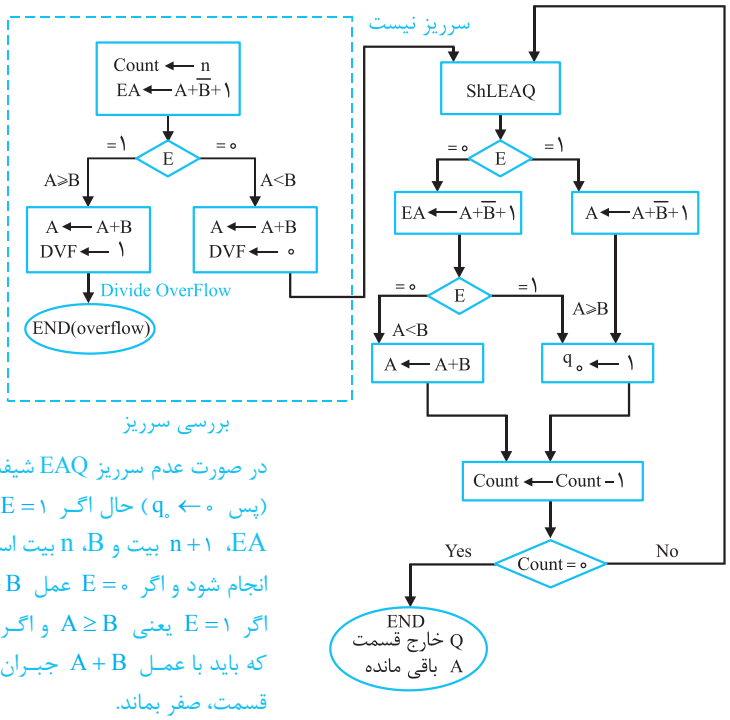
۳- در تقسیم A : Q به B که خارج قسمت در Q قرار خواهد گرفت اگر $A \geq B$ آنگاه سرریز است و برعکس.

پیاده‌سازی تقسیم سه روش دارد: مقایسه، جبرانی یا بازیابی و غیرجبرانی.

روش مقایسه (Comparison): این روش شبیه تقسیم قلم و کاغذ است. n بار A با B مقایسه می‌شود اگر $A \geq B$ آنگاه $A \leftarrow A-B$ و در خارج قسمت ۱ قرار می‌گیرد و اگر $A < B$ آنگاه در خارج قسمت صفر قرار می‌گیرد، و در هر مرحله $A:Q$ شیفت چپ داده می‌شود در واقع به این معنی که یک بیت از Q جدا می‌شود.

روش جبرانی (restoring): در این روش مقایسه انجام نمی‌شود بلکه با انجام تفریق $A \leftarrow A-B$ مشخص می‌شود که $A \geq B$ یا $A < B$. اگر $A \geq B$ که تفریق به درستی انجام شده و در خارج قسمت، یک قرار می‌گیرد ولی اگر $A < B$ تفریق باید جبران شود یعنی عمل جمع با B انجام شود که همان مقدار قبلی A بازیابی شود و در خارج قسمت صفر قرار گیرد.

فلوچارت زیر تقسیم $A:Q$ به B را با روش جبرانی به همراه بررسی سرریز نشان می‌دهد.



در صورت عدم سرریز EAQ شیفت چپ داده می‌شوند (پس $q_0 \leftarrow 0$) حال اگر $E=1$ حتماً $EA > B$ زیرا EA، $n+1$ بیت و B، n بیت است پس باید $A-B$ انجام شود و اگر $E=0$ عمل $A-B$ انجام می‌شود حال اگر $E=1$ یعنی $A \geq B$ و اگر $E=0$ یعنی $A < B$ که باید با عمل $A+B$ جبران صورت گیرد و خارج قسمت، صفر بماند.

مثال ۱۴: تقسیم ۱۵۷ به ۱۲ را با کمک ثبات‌های ۴ بیتی و روش جبرانی انجام دهید.

پاسخ: $۱۵۷ = ۱۲۸+۱۶+۸+۴+۱ = ۱۰۰۱۱۱۰۱$ و $۱۲ = ۱۱۰۰$ پس $A = ۱۰۰۱$ و $Q = ۱۱۰۱$ و $B = ۱۱۰۰$ و $E = ۰$ (نیمهٔ بالای مقسوم A و نیمهٔ پایین مقسوم Q است) چون $A < B$ پس

سرریز نیست و ما در فلوچارت سرریز را بررسی نمی‌کنیم. باید ۴ بار عملیات انجام شود:
$(\overline{B}+۱=۰۱۰۰)$

		E	A	Q
۱) shL EAQ	$\Rightarrow$	۱	۰۰۱۱	۱۰۱۰
$E=۱\Rightarrow \begin{cases} A \leftarrow A+\overline{B}+۱ \\ q_۰ \leftarrow ۱ \end{cases}$	$\Rightarrow$	۱	۰۱۱۱	۱۰۱۰
	$\Rightarrow$	۱	۰۱۱۱	۱۰۱۱
۲) shL EAQ	$\Rightarrow$	۰	۱۱۱۱	۰۱۱۰
$EA \leftarrow A+\overline{B}+۱$	$\Rightarrow$	۱	۰۰۱۱	۰۱۱۰
$E=۱\Rightarrow q_۰ \leftarrow ۱$	$\Rightarrow$	۱	۰۰۱۱	۰۱۱۱
۳) shL EAQ	$\Rightarrow$	۰	۰۱۱۰	۱۱۱۰
$EA \leftarrow A+\overline{B}+۱$	$\Rightarrow$	۰	۱۰۱۰	۱۱۱۰
$E=۰\Rightarrow A \leftarrow A+B$	$\Rightarrow$	۰	۰۱۱۰	۱۱۱۰
۴) shL EAQ	$\Rightarrow$	۰	۱۱۰۱	۱۱۰۰
$EA \leftarrow A+\overline{B}+۱$	$\Rightarrow$	۱	۰۰۰۱	۱۱۰۰
$E=۱\Rightarrow q_۰ \leftarrow ۱$	$\Rightarrow$	۱	۰۰۰۱	۱۱۰۱

پس خارج قسمت $Q=۱۱۰۱$ یعنی ۱۳ و باقی‌مانده $A=۰۰۰۱$ یعنی ۱ است.

روش غیرجبرانی (nonrestoring): در این روش اگر پس از تفریق به این نتیجه برسیم که $A<B$ برخلاف روش جبرانی، B جمع نمی‌شود و در عوض حاصل تفریق به چپ شیفت داده می‌شود و سپس با B جمع می‌گردد. برای اینکه ببینیم چگونه این عمل صحیح است، حالتی را در نظر بگیرید که در آن $A<B$ است. در روش جبرانی، حاصل $A-B$ با B جمع می‌شود یعنی $A-B+B$ که A بازیافت شود، در مرحلهٔ بعدی، این عدد به چپ شیفت داده می‌شود یعنی در ۲ ضرب می‌شود و دوباره B از آن کم می‌شود، نتیجه این عملیات $۲(A-B+B)-B=۲A-B$ است. این نتیجه در روش غیرجبرانی به این شکل بدست می‌آید که پس از محاسبهٔ $A-B$ با اینکه $A<B$ ولی جبران صورت نمی‌گیرد (با B جمع نمی‌شود) بلکه در مرحلهٔ بعدی حاصل به چپ شیفت داده می‌شود و حال با B جمع می‌شود که نتیجهٔ کار $۲(A-B)+B=۲A-B$ است که برابر با همان نتیجه قبلی است. بنابراین در روش غیرجبرانی اگر مقدار قبلی $q_۰$ برابر ۱ باشد B تفریق می‌گردد و اگر مقدار قبلی $q_۰$ برابر ۰ باشد B جمع می‌شود و جبران لازم نیست. به این ترتیب یک مرحله، یعنی جمع کردن مقسوم‌علیه اگر A کوچکتر از B باشد، صرفه‌جویی می‌شود ولی نیاز به کنترل خاصی برای بخاطر سپردن نتیجهٔ قبلی دارد. اولین باری که مقسوم شیفت

می‌یابد، B باید تفریق شود همچنین اگر آخرین بیت خارج قسمت ۰ باشد، باقی‌مانده جزئی باید بازیابی شود تا باقیمانده نهایی صحیح باشد.

نکته

اگر ثباتها n بیتی بدون علامت باشند و فرض کنیم مقداردهی اولیه انجام شده است و مشکل سرریز نداریم آنگاه:

روش	پیچیدگی
مقایسه	n مقایسه، n شیفت و به تعداد یک‌های خارج قسمت تفریق (متوسط $\frac{n}{۲}$)
جبرانی	n شیفت، n تفریق و به تعداد صفرهای خارج قسمت جمع (متوسط $\frac{n}{۲}$)
غیرجبرانی	n شیفت، به تعداد یک‌های خارج قسمت تفریق (متوسط $\frac{n}{۲}$) و به تعداد صفرهای خارج قسمت جمع (متوسط $\frac{n}{۲}$)

تقسیم سیستم علامت مقدار: کافی است علامت مقسوم و مقسوم‌علیه را XOR کرده که علامت حاصل بدست آید و سپس ارزش‌ها را همانند تقسیم بی‌علامت، عمل کنیم.

تقسیم سیستم مکمل ۲

فرض کنید مقسوم در $A:Q$ و مقسوم‌علیه در B است و سیستم مکمل ۲ است. مراحل تقسیم:

۱- ShL AQ

۲- اگر A و B هم علامت هستند عمل $A \leftarrow A-B$ و در غیر این صورت عمل $A \leftarrow A+B$ انجام شود.

۳- اگر علامت A بعد از عمل ۲ عوض نشد یا $A=۰$ آنگاه $Q_۰=۱$ یعنی بیت سمت راست خارج قسمت ۱ می‌شود. ولی اگر بعد از عمل ۲ علامت A عوض شد و $A\neq ۰$ آنگاه $Q_۰ \leftarrow ۰$ و عمل ۲ را جبران کن (یعنی اگر در عمل ۲ جمع انجام شده آنگاه تفریق کن و اگر تفریق انجام شده آنگاه جمع کن)

۴- به تعداد بیت‌های Q عملیات ۱ و ۲ و ۳ انجام شود.

۵- باقیمانده A است. اگر مقسوم و مقسوم‌علیه هم علامت بودند، خارج قسمت Q است در غیر این صورت، خارج قسمت، مکمل دو Q است.

مثال ۱۵: با ثباتهای ۴ بیتی تقسیم $(+۳) \div (-۷)$ را انجام می‌دهیم.

$AQ = ۱۱۱۱۱۰۰۱$, $B = ۰۰۱۱$

	A	Q
ShL A Q:	۱۱۱۱	۰۰۱۰
$A \leftarrow A + B$:	۰۰۱۰	۰۰۱۰
restore:	۱۱۱۱	۰۰۱۰
ShL AQ:	۱۱۱۰	۰۱۰۰
$A \leftarrow A + B$:	۰۰۰۱	۰۱۰۰
restore:	۱۱۱۰	۰۱۰۰
ShL AQ:	۱۱۰۰	۱۰۰۰
$A \leftarrow A + B$:	۱۱۱۱	۱۰۰۰
set $Q_۰ \leftarrow ۱$:	۱۱۱۱	۱۰۰۱
ShL AQ:	۱۱۱۱	۰۰۱۰
$A \leftarrow A + B$:	۰۰۱۰	۰۰۱۰

restore: ۱۱۱۱ ۰۰۱۰ $\Rightarrow$ $A = (۱۱۱۱)_۲ = -۱$, $\overline{Q} + ۱ = (۱۱۱۰)_۲ = -۲$

$(-۷ \text{ div } ۳ = -۲ \text{ , } -۷ \text{ mod } ۳ = -۱)$

توجه: به تست‌های ۱۶ تا ۳۳ پاسخ دهید.

۶- نمایش اعداد اعشاری

برای نمایش اعداد اعشاری می‌توان از نمایش ممیز ثابت (fixed point) و ممیز شناور (floating point) استفاده کرد. در سیستم ممیز ثابت، تعداد ارقام قسمت صحیح و قسمت اعشار ثابت و مشخص است. مثلاً در مبنای ۱۰ اعداد ۳ رقمی ممیز ثابتِ مثبت که دو رقم اعشار دارند دارای حداقل ۰/۰۰ و حداکثر ۹/۹۹ می‌باشند یعنی بازهٔ (Range) این نمایش [۰/۰۰...۹/۹۹] است. دقت (precision) فاصلهٔ بین دو عدد متوالی قابل نمایش است. با نمایش فوق دقت ۰/۰۱ (یک صدم) است (مثلاً فاصلهٔ ۰/۰۰ با عدد بعدی‌اش یعنی ۰/۰۱). حداکثر خطا (error) برابر نصف فاصلهٔ دو عدد متوالی قابل نمایش است. پس حداکثر خطا برای نمایش فوق برابر $\frac{۰/۰۱}{۲} = ۰/۰۰۵$ است. نمایش ممیز ثابت برای اعداد اعشاری در هیچ ماشینی استفاده نمی‌شود و همهٔ ماشین‌ها از نمایش ممیز شناور استفاده می‌کنند. فرمت‌های متفاوتی برای ممیز شناور وجود دارد که دو نمونه را بررسی می‌کنیم.

ممیز شناور فرمت ۱: عدد اعشاری به صورت | S | E | M | ذخیره می‌شود که فیلدهای S(sign)، E(Exponent) و M(Significand or Mantissa) به ترتیب علامت، توان یا نما و مانتیس هستند. عددی که با این فرمت ذخیره می‌شود، مقدارش از رابطهٔ $(-۱)^S \times (۰/M)_B \times B^E$ بدست می‌آید که B(Base) به طور ضمنی مشخص می‌شود و ما فرض می‌کنیم $B=۲$، ولی در برخی ماشین‌ها ممکن است $B=۱۶$ یا هر مقدار دیگری باشد. مانتیس معمولاً به صورت هنجار شده یا نرمال ذخیره می‌شود به این معنی که با ارزش‌ترین رقم مانتیس مخالف صفر است پس اگر $B=۲$ آنگاه با ارزش‌ترین (چپ‌ترین) رقم مانتیس برابر ۱ است. علت نرمال‌سازی این است که برای هر عدد یک نمایش منحصر به فرد ایجاد شود. مثلاً در مبنای ۱۰ عدد ۰/۰۵ را می‌توان به شکل‌های $۰/۵\times۱۰^{-۱}$، $۰/۰۵\times۱۰^{۰}$، ۵×۱۰^{-۲} و ... نمایش داد که همگی طبق فرمت گفته شده برای ممیز شناور هستند ولی فقط اولی نرمال است. همهٔ اعداد را می‌توان نرمال کرد و کافیست مانتیس به درستی شیفت داده شود. تنها عددی که نرمال نمی‌شود صفر است. برای نمایش صفر در کامپیوتر فیلدهای E و M تماماً صفر می‌شوند. مانتیس نرمال در مبنای ۲ به شکل $(۰/۱...)_۲$ است پس مقدار مانتیس نرمال در بازهٔ $[۰/۵,۱)$ می‌باشد. نما، خود یک عدد علامت‌دار است و می‌تواند با هر سیستمی ذخیره شود ولی معمولاً با سیستم بایاس ذخیره می‌شود. اگر نما e بیتی باشد مقدار بایاس برابر $bias=۲^{e-۱}$ است (در فرمت‌های دیگر که خواهیم دید بایاس $۲^{e-۱}-۱$ می‌باشد). مزیت بایاس این است که اولاً مقایسهٔ نماها ساده‌تر می‌شود و ثانیاً کوچکترین نما به شکل تماماً صفر نمایش داده می‌شود و این باعث می‌شود که عدد صفر تماماً صفر شود زیرا در کامپیوترها کوچکترین نما را به صفر تخصیص می‌دهند.

مثال ۱۶: فرض کنید نما ۴ بیتی بایاس شده است عدد ممیز شناور $a=۰۱۱۰۱۱۰۰۰۱۱۱۰۰۰۰$ را به مبنای ۱۰ تبدیل کنید.

پاسخ: با توجه به اینکه نما ۴ بیتی است پس $S=۰$ و $E=۱۱۰۱$ و $M=۱۰۰۰۱۱۱۰۰۰۰$ و چون نما ۴ بیتی است پس $bias=۲^۳=۸$ می‌باشد بنابراین نما برابر $۵+=۸-۱۳$ می‌باشد پس:

$$a=+(۰/۱۰۰۰۱۱۱۰۰۰۰)_۲\times۲^{+۵}=(۲^{-۱}+۲^{-۵}+۲^{-۶}+۲^{-۷})\times۲^{+۵}=۲^{+۴}+۲^{۰}+۲^{-۱}+۲^{-۲}=(۱۷/۷۵)_{۱۰}$$

مثال ۱۷: عدد ۲/۸۷۵- را با فرض نمای ۴ بیتی بایاس شده و مانتیس نرمال، در قالب ۱۶ بیت به صورت hex نمایش دهید.

پاسخ: ابتد عدد داده شده را به مبنای ۲ تبدیل می‌کنیم سپس با شیفت نرمال می‌کنیم:

$$-۲/۸۷۵=-(۱۰/۱۱۱)_۲ \xrightarrow{\text{۲ واحد شیفت راست}} -(۰/۱۰۱۱۱)_۲\times۲^{+۲}$$

با توجه به صورت سوال، مانتیس یازده بیتی است بنابراین $M=۱۰۱۱۱۰۰۰۰۰۰$ و نما ۲+ است که باید با ۸ جمع شود و سپس ذخیره شود پس $E=۱۰۱۰$ بنابراین نمایش عدد ۲/۸۷۵− با فرمت ممیز شناور گفته شده به شکل زیر است:

$$\left(\underbrace{۱}_{S}\ \underbrace{۱۰۱۰}_{E}\ \underbrace{۱۰۱۱۱۰۰۰۰۰۰}_{M}\right)_{۲}=D5C0_{hex}$$

مثال ۱۸: با فرضیات مثال قبل **الف)** بزرگترین و کوچکترین عدد مثبت قابل نمایش را بیابید. **ب)** دقت این سیستم را محاسبه کنید.

پاسخ: الف) کوچک‌ترین عدد مثبت وقتی حاصل می‌شود که $S=۰$ (عدد مثبت) و E و M مینیمم باشند. با توجه به اینکه نما بایاس است پس $E_{min}=۰۰۰۰$ و با توجه به اینکه مانتیس نرمال است پس $M_{min}=۱۰۰۰۰۰۰۰۰۰$ پس کوچکترین عدد مثبت برابر است با:

$$+۰/۱۰۰۰۰۰۰۰۰۰۰\times۲^{۰۰۰۰}=+۲^{-۱}\times۲^{۰-۸}=+۲^{-۹}$$

به همین ترتیب بزرگترین عدد مثبت برابر است با:

$$+Max=\underbrace{۰}_{S}\ \underbrace{۱۱۱۱}_{E}\ \underbrace{۱۱۱۱۱۱۱۱۱۱۱}_{M}=+۰/۱۱۱۱۱۱۱۱۱۱۱\times۲^{۱۱۱۱}=+(۱-۲^{-۱۱})\times۲^{۱۵-۸}$$

$$=+(۱-۲^{-۱۱})\times۲^{+۷}$$

خوب است بدانیم که n تا یک بعد از ممیز در مبنای ۲ یعنی $۰/\underbrace{۱۱\ldots۱}_{n\ \text{تا}}$ برابر $۱-۲^{-n}$ می‌باشد.

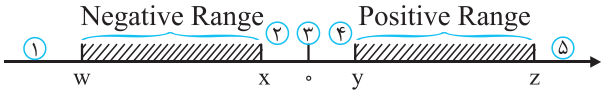
ب) دقت یعنی فاصله بین اعداد متوالی قابل نمایش. در سیستم ممیز شناور دقت ثابت نیست و هرچه به سمت $+Max$ پیش می‌رویم فاصلهٔ بین اعداد متوالی بیشتر می‌شود (چرا؟) و دقت نمایش بدتر می‌شود. پس ما ریزترین و درشت‌ترین دقت را می‌یابیم. ریزترین (بهترین) دقت فاصلهٔ بین مینیمم مثبت و عدد بعدی‌اش است و درشت‌ترین (بدترین) دقت فاصلهٔ بین ماکزیمم مثبت و عدد قبلی‌اش است. پس ریزترین دقت برابر $۲^{-۱۹}=۲^{-۱۱}\times۲^{-۸}$ و درشت‌ترین دقت برابر $۲^{-۴}=۲^{-۱۱}\times۲^{+۷}$ می‌باشد.

نکته

اگر $B=۲$ آنگاه چپ‌ترین رقم مانتیس نرمال همواره برابر ۱ است بنابراین می‌توان آن را ذخیره نکرد و مخفی (hidden) کرد یعنی به طور ضمنی (implicit) آن را در ثبات ذخیره نکرد ولی در محاسبات وارد کرد. دقت کنید اگر $B>۲$ آنگاه امکان مخفی‌سازی وجود ندارد.

مثال ۱۹: فرض کنید سیستم ممیز شناور ۳۲ بیتی و مانتیس نرمال باشد و بیت مخفی داشته باشیم. نما ۸ بیتی بایاس شده است. بازهٔ اعداد مثبت و منفی قابل نمایش را بیابید.

پاسخ: با توجه به محور اعداد، w کوچکترین عدد منفی، x بزرگترین عدد منفی، y کوچکترین عدد مثبت و z بزرگترین عدد مثبت است که محاسبه می‌کنیم:



$$y = +۰/۱۰۰\ldots۰ \times ۲^{۰۰۰۰۰۰۰۰-bias} = ۲^{-۱} \times ۲^{-۱۲۸} = ۲^{-۱۲۹}$$

$$z = +۰/\underbrace{۱۱۱..۱}_{۲۴} \times ۲^{۱۱۱۱۱۱۱۱-bias} = (۱-۲^{-۲۴}) \times ۲^{+۱۲۷}$$

۲۴ تا یک (۱ مخفی هم در نظر گرفتیم)

به همین ترتیب $x = -y = -۲^{-۱۲۹}$ و $w = -z = -(۱-۲^{-۲۴}) \times ۲^{+۱۲۷}$.

لازم به ذکر است روی محور اعداد ۵ ناحیهٔ دیگر با شماره مشخص شده‌اند که عبارتند از:

۱) اعداد منفی کوچکتر از w که قابل نمایش نیستند (negative overflow)

۲) اعداد منفی بزرگتر از x (negative underflow) که قابل نمایش نیستند و معمولاً به صفر تقریب زده می‌شوند. ولی در برخی استانداردها می‌توانند نمایش داده شوند.

۳) صفر که به صورت E و M تماماً صفر نمایش داده می‌شود

۴) اعداد مثبت کوچکتر از y(Positvie underflow)

۵) اعداد مثبت بزرگتر از z(positive overflow)

نکته

۱- برای افزایش دامنه اعداد قابل نمایش می‌توان ارقام نما را افزایش داد که در این صورت مانتیس کوچک می‌شود و دقت نمایش کاهش می‌یابد. یا می‌توان پایه (B) را افزایش داد مثلاً پایهٔ $B = ۲$ را به $B = ۱۶$ تبدیل کرد که در این صورت دقت اعداد کوچک بهتر می‌شود و دقت اعداد بزرگ بدتر می‌شود.

۲- علت ذخیرهٔ ممیز شناور به صورت SEM، این است که دو عدد ممیز شناور را به راحتی بتوان با هم مقایسه کرد. زیرا در مقایسه، مهم‌ترین فیلد S یعنی علامت است اگر علامت دو عدد متفاوت باشد کار تمام است. در صورت مساوی بودن علامت‌ها E باید بررسی شود و در صورت یکسان بودن نماها M بررسی می‌شود.

جمع و تفریق ممیز شناور: باید نمای دو عدد یکسان باشد. پس ابتدا نماها با هم مقایسه می‌شوند و اگر یکسان نباشند، نمای کوچکتر به نمای بزرگتر تبدیل می‌شود و مانتیس آن به اندازهٔ اختلاف نماها به سمت راست شیفت داده می‌شود که ممکن است تعدادی از ارقام سمت راست را از دست بدهیم. پس از یکسان‌سازی نماها، آن را برای نمای حاصل انتخاب می‌کنیم و سپس مانتیس‌ها را با هم جمع یا تفریق می‌کنیم و در نهایت اگر مانتیس حاصل نرمال نباشد با شیفت آن را نرمال می‌کنیم.

مثال ۲۰: با فرض اینکه نما و مانتیس هر دو ۴ بیتی هستند به جمع و تفریق‌های زیر توجه کنید:

a) $۰/۱۱۰۱\times۲^{۱۰۱۰}+۰/۱۰۱۰\times۲^{۱۰۰۱}=۰/۱۱۰۱\times۲^{۱۰۱۰}+\underbrace{۰/۰۱۰۱\times۲^{۱۰۱۰}}_{\text{نما یک واحد زیاد شد و مانتیس یک واحد شیفت راست داده شد}}$

$=\underbrace{۱/۰۰۱۰}_{\text{در مانتیس سرریز اتفاق افتاده که با شیفت به راست درست می‌شود}}\times۲^{۱۰۱۰}=۰/۱۰۰۱\times۲^{۱۰۱۱}$

b) $۰/۱۱۱۰\times۲^{۱۰۰۱}-۰/۱۱۰۰\times۲^{۱۰۰۱}=\underbrace{۰/۰۰۱۰\times۲^{۱۰۰۱}}_{\text{با ۲ شیفت چپ نرمال می‌شود}}=۰/۱۰۰۰\times۲^{۰۱۱۱}$

در برخی منابع این حالت که بیت‌های باارزش مانتیس صفر می‌شود و نیاز به شیفت چپ دارد را زیرریز (underflow) در مانتیس گویند.

ضرب ممیز شناور: مانتیس‌ها را درهم ضرب و نماها را جمع می‌کنیم. چون نماها بایاس شده هستند باید از جمع نماها مقدار bias را کم کنیم که با توجه به اینکه $\text{bias}=۲^{e-۱}$، کم کردن بایاس از نما به این معنی است که بیت سمت چپ نمای حاصل را عوض کنیم. هنگام ضرب مانتیس‌ها ممکن است زیرریز در مانتیس رخ دهد که با شیفت چپ اصلاح می‌شود. توجه کنید که امکان سرریز در مانتیس وجود ندارد.

مثال ۲۱: با فرض اینکه نما و مانتیس ۴ بیتی هستند به ضرب مقابل توجه کنید:

$(۰/۱۰۰۱\times۲^{۱۰۱۰})\times(۰/۱۰۰۰\times۲^{۰۱۰۱})=۰/۱۰۰۱\times۰/۱۰۰۰\times۲^{۱۰۱۰+۰۱۰۱-\text{bias}}=۰/۰۱۰۰۱۰۰۰$

$\times۲^{۱۱۱۱-۸}\xrightarrow{\text{مانتیس ۴ بیتی}}\underbrace{۰/۰۱۰۰}_{\text{زیرریز در مانتیس}}\times۲^{۰۱۱۱}\xrightarrow{\text{شیفت چپ}}۰/۱۰۰۰\times۲^{۰۱۱۰}$

در برخی ماشین‌ها، بیت‌های موسوم به محافظ (guard) و گردسازی (round) وجود دارد که ۲ بیت اضافه بر ظرفیت ذخیره‌سازی را نگه می‌دارند و هنگام شیفت و گرد کردن استفاده می‌شوند. در این مثال حاصلضرب مانتیس‌ها $۰/۰۱۰۰۱۰۰۰$ (guard, round) است که اگر بیت‌های محافظ و گردسازی وجود داشته باشند، $۰/۰۱۰۰۱۰$ می‌شود و در نهایت حاصل نهایی $۰/۱۰۰۱\times۲^{۰۱۱۰}$ می‌شود.

تقسیم ممیز شناور: اگر مقسوم‌علیه صفر باشد که اعلام تقسیم به صفر می‌شود و در غیر این صورت مانتیس‌ها را به هم تقسیم و نماها را از هم کم می‌کنیم و به نمای حاصل مقدار بایاس را اضافه می‌کنیم یعنی بیت سمت چپ نمای حاصل را عوض می‌کنیم.

در تقسیم مانتیس‌ها امکان سرریز وجود دارد که البته مشکلی ایجاد نمی‌کند و با شیفت به راست درست می‌شود ولی می‌توان قبل از تقسیم مانتیس‌ها، مانتیس مقسوم را یک واحد به راست شیفت دهیم (به این عمل هم ردیف کردن مقسوم گویند) و نمای آن را یک واحد افزایش دهیم و سپس عمل تقسیم را انجام دهیم که دیگر سرریز در مانتیس رخ نمی‌دهد.

مثال ۲۲: با نما و مانتیس ۴ بیتی، تقسیم مقابل را انجام می‌دهیم:

$$(۰/۱۰۱۱\times۲^{۱۱۰۱})\div(۰/۱۰۰۱\times۲^{۱۰۰۱}) \xrightarrow{\text{بدون هم ردیف کردن مقسوم}} \frac{۰/۱۰۱۱}{۰/۱۰۰۱}\times۲^{۱۱۰۱-۱۰۰۱+\text{bias}}$$

$$۰/۱۰۰۱\times۲^{۱۱۰۱} \xrightarrow{\text{مانتیس ۴ بیتی}} ۰/۱۰۰۱۱\times۲^{۱۱۰۱} \xrightarrow{\text{شیفت راست}} \underbrace{۱/۰۰۱۱}_{\text{سرریز در مانتیس}}\times۲^{۱۱۰۰} = ۱/۰۰۱۱\times۲^{۰۱۰۰+۸}$$

حال همین تقسیم را با هم ردیف کردن مقسوم انجام می‌دهیم:

$$(۰/۱۰۱۱\times۲^{۱۱۰۱})\div(۰/۱۰۰۱\times۲^{۱۰۰۱}) = (۰/۰۱۰۱۱\times۲^{۱۱۱۰})\div(۰/۱۰۰۱\times۲^{۱۰۰۱})$$

$$= \frac{۰/۰۱۰۱۱}{۰/۱۰۰۱}\times۲^{۱۱۱۰-۱۰۰۱+\text{bias}} = ۰/۱۰۰۱\times۲^{۱۱۰۱}$$

نکته

در تمام عملیات ممیز شناور، امکان زیریز و سرریز در نما هست. سرریز در نما یعنی نمای حاصل از ماکزیمم نمای قابل ذخیره بزرگتر شده است که خطای سرریز است و حاصل قابل ذخیره نیست. زیرریز در نما یعنی نمای حاصل از مینیمم نمای قابل ذخیره کوچکتر شده است که در این صورت حاصل بسیار ریز است و می‌توان آن را به صفر تقریب زد ولی برخی استانداردها چنین اعدادی را به صورت غیرنرمال ذخیره می‌کنند.

ممیز شناور فرمت ۲: همانند فرمت قبل، عدد اعشاری به شکل | S | E | M | ذخیره می‌شود. با این تفاوت‌ها که برای نمای e بیتی، مقدار بایاس $۲^{e-۱}$ نیست بلکه $۲^{e-۱}-۱$ است و مانتیس نرمال به شکل ۰/۱... نیست بلکه به شکل ۱/... می‌باشد که معروف به نماد علمی است بنابراین مانتیس در بازهٔ $[۱,۲)$ است.

مثال ۲۳: فرض کنید عدد اعشاری ۳۲ بیتی که نما ۸ بیتی بایاس شده و مانتیس نرمال دارای بیت مخفی، طبق فرمت ۲، کوچکترین و بزرگترین عدد مثبت را بیابید.

پاسخ: با نمای ۸ بیتی مقدار بایاس برابر $\text{bias} = ۲^{۷}-۱ = ۱۲۷$ است.