جزوه درس مدار منطقی

علیرضا سلطانی نشان

دانشجوی نرم افزار

ترم سوم

فهرست مطالب

۱ اعداد - مبناها - مکمل ها - کدها

کلا اگه ما بخوایم هر عددی را نمایش بدهیم یا آنرا بنویسیم، این اعداد میتواند مبنا های مختلفی داشته باشد. در حالت کلی عددی مثل a را میتوانیم اینگونه نمایش دهیم:

- $a = a_{n-1}a_{n-2}...a_2a_1a_0.a_{-1}a_{-2}a_{-3}...a_{-m}$

در حقیقت هر کدام از این اعداد دارای ارزش مکانی هستند، که در زیر به درستی نشان داده شده است :

- $a = \overset{r^{n-1}}{a_{n-1}}, \overset{r^{n-2}}{a_{n-2}}, ...\overset{r^2}{a_2}, \overset{r^1}{a_1}, \overset{r^0}{a_0}, .\overset{r^{-1}}{a_{-1}}, \overset{r^{-2}}{a_{-2}}, \overset{r^{-m}}{a_{-m}}$

که بعد از اعشار (۰) دارای n رقم عدد صحیح و قبل آن n رقم عدد اعشاری داریم.
نکته مهمی که در اینجا بایستی یادآوری شود آنست که هر عدد به توان منفی برابر است با:

- $r^{-m} = \frac{1}{r^m}$

معرفی انواع مبنا ها:

۱. $0...1$ = مبنای دو Binary

۲. $0...9$ = مبنای ۱۰ Decimal

۳. $0...7$ = مبنای هشت Octal

۴. $0...9, A, B, C, D, E, F$ = مبنای ۱۶ Decimal Hexa

نکته: زمانی که در مبناها، به مبنا های بزرگی میرسیم، ما از ۰ تا ۹ را به رسمیت میدانیم و برای نمایش اعداد بعد از آن از حروف انگلیسی استفاده میکنیم.

اگر عددی در مبنای r باشد و بخواهیم آنرا به مبنای ۱۰ تبدیل کنیم کافیست هر رقم را در ارزش مکانی خودش ضرب کرده و حاصل را باهم جمع کنیم.

۱.۰.۱ تصاعد هندسی

$$(r-1)_{r^1}(r-1)_{r^0}.(r-1)_{r^{-1}}(r-1)_{r^{-m}} = (r-1)\frac{r^n-1}{r-1} = r^n-1$$

بطوری که r مبنا و n تعداد ارقام است، بطور مثال، در مبنای دسیمال که سیستم دهدهی است، برای بدست آوردن بزرگ ترین عدد مجموعه از $r-1$ که میشود ۹، یعنی ۹ بزرگ ترین عدد مجموعه مبنای دسیمال است، و اگر به تعداد دو رقم آنرا در نظر بگیریم، یعنی ۹۹، در جایگاه مربوطه بررسی خواهیم کرد، یعنی $99 = (9*10^1) + (10^0*9)$ ودر مقابل تصاعد هندسی داریم، $10^2-1=99$.

تمرین: تبدیل مبناهای زیر را انجام دهید:

در هنگام انجام تبدیل مبنا های زیر، قصد یادآوری مطالب گذشته است، اما در بین آنها مطابی هم ذکر شده است که نکاتی را در بر دارند و بایستی به عنوان مطلب جدید آنرا در نظر داشت:

نکات

برای تبدیل مبنا از ۱۰ به هر مبنای دیگری باید ۱۰ را در آن مبنا پی در پی تقسیم کنیم که در نهایت به با استفاده از باقی مانده ها و مقسوم

علیه آخر تقیسم از راست به چپ نتایج تقسیم را کنار هم بگذاریم.

در هنگام تبدیل مبنا، جلوترین و اخرین ارزش مکانی عدد را MSB یا $MostSignificantBit$ و کمترین و اخرین ارزش مکانی در عدد را LSB یا $LeastSignificantBit$ گفته میشود. مانند: $(327.2)_{10}$ که عدد ۳ MSB و ۷ LSB .

۱.۱ تبدیل مبنا از ۲ به ۱۰ یا بلعکس

در تبدیل مبنا از ۲ به ۱۰ میبایست به صورت عادی با استفاده از ارزش مکانی هر عدد 2^0 و 2^1 و... یا به صورت خودمان ۱، ۲، ۴، ۸، ۱۶، ۳۲، ۶۴،... و در نهایت هر قسمتی که بیت روشن یا ۱ داشت را نتیجه را با هم جمع میکنیم.

۲.۱ تبدیل مبنا از ۲ به ۱۰

تمرین:

- $(11001)_{2\to10} = (1*1) + (1*8) + (1*16) = 1 + 8 + 16 = 25$
- $(1111001)_{2\to10} = (1*1)+(1*8)+(1*16)+(1*32)+(1*64) = 1 + 8 + 16 + 32 + 64 = (121)_{10}$
- $(100011111)_{2\to10} = (1*1)+(1*2)+(1*4)+(1*8)+1*16)+(1*256) = (281)_{10}$

۳.۱ تبدیل مبنا از ۱۰ به دو

همانطور که در بالاتر گفته شد در تبدیل مبنا از ۱۰ به ۲ یا به هر مبنای دیگر میتوانیم هب راحتی از تقسیم پی در پی ۱۰ در مبانی مورد نظر با استفاده از باقی مانده ها به جواب نهایی برسیم.
تمرین:

- $(34)_{10\to2} = $ $(34)_{10\to2} = 100010$

۴.۱ تبدیل مبنا از ۱۰ به هشت و برعکس

- $(972)_{10\to8} = $ $(972)_{10\to8} = (1637)_8$
- $(1637)_{8\to10} = (7*8^0)+(3*8^1)+(6*8^2)+(1*8^3) = (927)_{10}$

Calculate	باقی مانده
۳۴ / ۲ = ۱۷	۰
۱۷ / ۲ = ۸	۱
۸ / ۲ = ۴	۰
۴ / ۲ = ۲	۰
۲ / ۲ = ۱	۰
۱ / ۲ =	۱

Calculate	باقی مانده
۹۷۲ / ۸ = ۱۱۵	۷
۱۱۵ / ۸ = ۱۴	۳
۱۴ / ۸ = ۱	۶
۱ / ۸ = ۰	۱

۵.۱ تبدیل اعداد صحیح از مبنای ۱۰ به ۱۶ و برعکس

برای تبدیل مبنا از ۱۰ به ۱۶ درست مثل قبل از تقسیم پیا پی ۱۰ در ۱۶ استفاده خواهیم کرد:

- $(954)_{10 \to 16}$ $(954)_{10 \to 16} = (3BA)_{16}$

Calculate	مانده باقی
۹۵۴ / ۱۶ = ۵۹	۱۰→ A
۵۹ / ۱۶ = ۳	۱۱→ B
۳ / ۱۶ = ۰	۳

- $(3BA)_{16\to10} = (10*16^0)+(11*16^1)+(3*16^2) = 954$

۶.۱ تبدیل اعشاری دهدهی به دودویی و برعکس

مهم ترین بخش تبدیل مبناها زمانیست که شما با مبنایی اعشاری رو به رو میشوید، در حالت تبدیل مبنا از دودویی به سیستم دهدهی، قسمتی که به صورت صحیح است را طبق معمول محاسبه می کنیم، و آن قسمتی که بعد از اعشار قرار دارد، به صورت ... $2^{-1},2^{-2},2^{-3}$ محاسبه خوهیم کرد و در نهایت نتیجه قسمت صحیح را به نتیجه قسمت اعشاری قرار خواهیم داد:

۱. $(1110.01)_{2\to10} =$

$(1110)_{2\to10} = 14$

$(.01)_{2\to10} = (0*2^{-1})+(1*2^{-2}) = \frac{1}{4}$

$\rightarrow 14 + \frac{1}{4} = \frac{56}{4} + \frac{1}{4} = \frac{57}{4} = 14.25$

۲. $(10111001.0101)_{2\rightarrow 10} =$

$(10111001)_{2\rightarrow 10} = 185$

$(0.0101)_{2\rightarrow 10} = \frac{1}{4} + \frac{1}{16} = \frac{4}{16} + \frac{1}{16} = \frac{5}{16} = 0.3125$

$\rightarrow 185 + 0.3125 = 185.3125$

حلا نوبت به انجام تبدیل مبنا ۱۰ به ۲ میرسد، در این بخش، طبق معمول آن قسمتی که سمت صحیح عدد قرار دارد را به صورت عادی از ۱۰ به ۲ تبدیل میکنیم، اما آن بخشی که به صورت اعشاری است را باید کمی توجه کنیم، قسمت اعشاری را ضرب ۲ میکنیم، و حاصل آن را بررسی میکنیم و قسمت صحیح حاصل را به عنوان نتیجه تقسیم اول در نظر میگیرم، و آن بخش اعشاری را (فقط اعشاری) دو باره ضرب در دو میکنیم، انقدر این ضرب را انجام میدهیم که قسمت اعشاری به صفر برسد، ممکن است در طی این عملیات تعداد ضرب کردن به ۲ زیاد شود، در این مواقع تا هشت مرحله ضرب کردن کافی است (قاعده پرشدن حافظه ۸ بیت). لازم به ذکر است که در تمامی تبدیل های سیستم دهدهی به هر مبنای دیگیری، مانند باینری یا اکتال یا سه سه ای، چهارچهاری و غیره دقیقا همان عملیاتی که گفته شد صورت میگیرد، هم بخش تقسیم پیاپی در قسمت صحیح، هم ضرب متناوب در قسمت اعشاری.

تمرین:

۱. $(12.25)_{10\to2} =$

$(12)_{10\to2} = (1100)_2$

$(0.25)_{10\to2} =$

۲۵.۰ * ۲ = ۵.۰	۰
۵.۰ * ۲ = ۰.۱	۱

بعد از اینکه عدد اعشار حاصل دومین ضرب برابر با صفر شد دیگر نیازی به ضرب کردن نیست و از همان دو عدد نتیجه ۰ و ۱ استفاده میکنیم و در نتیجه خواهیم داشت:

$\to (1100.01)$

۲. $(15.361)_{10\to2} =$

$(15)_{10\to2} = (1111)_2$

$(0.361)_{10\to2} =$

$\to (1111.01011100)$

۰	۳۶۱.۰ * ۲ = ۷۲۲.۰
۱	۷۲۲.۰ * ۲ = ۴۴۴.۱
۰	۴۴۴.۰ * ۲ = ۸۸۸.۰
۱	۸۸۸.۰ * ۲ = ۷۷۶.۱
۱	۷۷۶.۰ * ۲ = ۵۵۲.۱
۱	۵۵۲.۰ * ۲ = ۱۰۴.۱
۰	۱۰۴.۰ * ۲ = ۲۰۸.۰
۰	۲۰۸.۰ * ۲ = ۴۱۶.۰

۷.۱ تبدیل اعداد مبنای دو به مبنای هشت و برعکس

برای تبدیل مبنای دو به هشت فقط کافی است از سمت راست به چپ سه بیت سه بیت جدا کنیم و براساس ریتم ۱، ۲، ۴ آنها را باهم جمع کنیم، نکته ای که در این میان باید توجه کنیم ان است که اگر تعداد ارقام مضربی از سه نباشد، بایستی از سمت چپ، عدد صفر اضافه کنیم.

مثال/تمرین:

۱. $(11001)_{2\to 8} =$

$(011'001)_{2\to8} = (31)_8$

۲. $(1110001)_{2\to8} =$

$(001'110'001)_{2\to8} = (161)_8$

۳. $(100111)_{2\to8} =$

$(100'111)_{2\to8} = (47)_8$

تبدیل مبنا اعشاری از دو به هشت:

این نوع تبدیل مبناهم بایستی در نظر داشته باشیم که بهتر به دوقسمت صحیح و اعشاری تقسیم می شود، در قسمت اعشاری اگه تعداد ارقام مضربی از ۳ بود میتوان به صورت سه تا سه از راست به چپ ازجدایی را انجام داد، در غیر این صورت باید از سمت چپ به تعداد لازم ۰ اضافه کنیم. در قسمت اعشاری هم همین قاعده صادق است، با این تفاوت که اگه تعداد ارقام سمت اعشار مضربی از ۳ نبود این بار بایسیتی از سمت راست به تعداد لازم صفر وارد کنیم.

۱. $(10011.1101)_{2\to8} =$

$(010'011)_{2\to8} = (23)_8$

$(0.110'100)_{2\to8} = (64)_8$

$\to (23.64)_8$

تبدیل برعکس ۸ به ۲ هم دقیقا به همین صورت است.

۱. $(10101.0111)_{2\to8} =$
$(010'101)_{2\to8} = (25)_8$
$(011'100)_{2\to8} = (34)_8$
$\to (25.34)_8$

۲. $(25.34)_{2\to8} =$
$(010'101)_{2\to8} = (25)_8$
$(011'100)_{2\to8} = (34)_8$
$\to (10101.0111)_2$

۳. $(55.67)_{2\to8} =$
$(101'101)_{2\to8} = (55)_8$
$(110'111)_{2\to8} = (67)_8$
$\to (101101.110111)_2$

۸.۱ تبدیل مبنا ۲ به ۱۶ و برعکس

در تبدیل مبنا از ۲ به ۱۶ هم دقیقا مانند مبنای ۸ است که، بطوری که باید تعداد ارقام ضریبی از ۴ باشند و در هنگام اعشاری شدن هم باید

برای جبران کسری صفر در سمت چپ برای عدد صحیح و در سمت راست برای عدد اعشاری.

۱. $(1111101)_{2\to16} =$
$(0111'1101)_{2\to16} = (7D)_{16}$

۲. $(1011101100)_{2\to16} =$
$(0010'1110'1100)_{2\to16} = (2EC)_{16}$

۳. Decimal ۱۶ to ۲ $(1111101.0110)_{2\to16} =$
$(0111'1101)_{2\to16} = (7D)_{16}$
$(.0110)_{2\to16} = 6$
$\to (7D.6)_{16}$

۴. ۲ to ۱۶ $(F25.03)_{16\to2} =$
$(F25)_{16\to2} = (1111'0010'0101)_2$
$(.03)_{16\to2} = (0000'0011)_2$
$\to (111100100101.00000011)_2$

۹.۱ تبدیل اعداد مبنای ۸ به ۱۶ و برعکس

۱. $(A36)_{16\to8} =$

$(1010'0011'0110)_{16\to2} = (101000110110)_2$

$(101'000'110'110)_{2\to8} = (5066)_8$

۲. $(753)_{8\to16} =$

$(111'101'011)_{8\to2} = (111101011)_2$

$(0001'1110'1011)_{2\to16} = (1EB)_{16}$

Extra

۱۰.۱ متمم ها یا (Complements)

متمم ها در کامپیوتر های دیجیتال برای ساده کردن عمل تفریق و یا عملیات منطقی به کار میروند. ساده سازی عملیات منجر به پیاده سازی مدارات ساده تر میگردد. در هر مبنایی مانند r، دو نوع متتم وجود دارد: یکی متمم مبنا و دیگری متمم کاهش یافته. فرم اول به نام متتم r و دومی به متمم $r-1$ مرسوم است. وقتی که مقدار مبنا(یا پایه) را جایگذین کنیم، برای اعداد دودویی، متمم ها ی۲ و ۱ و برای دهدهی، متمم های ۱۰ و ۹ را خواهیم داشت.

دو نوع متتم برای هر عدد در مبنای r وجود دارد:

- متمم مبنا یا مکمل r
- متمم مبنای کاهش یافته یا مکمل $r-1$

اگر عدد N در مبنای r شامل n رقم باشد:

- متمم مبنا $r^n - N$
- متمم مبنای کاهش یافته $(r^n - 1) - N$

نکته بسیار مهم: r^n در هر مبنایی برابر است با $1 + n(0)$

برای مثال در 16^3 درست است که برابر با ۴۰۹۶ میشود اما این عدد بدست آمده در واقع در مبنای ۱۰ است نه در مبنای ۱۶، بهمین خاطر بایستی عدد حاصله را در مبنای ۱۶ تبدیل کنیم، که با توجه به قاعده بالا ما خواهیم داشت یک عدد ۱ به همراه تعداد ارقام صفر یعنی ۱۰۰۰
$2^3 = 1000$ $10^3 = 1000_{10}$ $8^3 = 1000_8$ $16^3 = (1000)_{16}$
اما، باید توجه داشته باشیم که هر کدام از اعداد بدست آمده بالا در حقیقت آنچیزی که نشان میدهد، نیست، فقط عدد ۱۰۰۰ در مبنای ۱۰ است، که حقیقتا برابر این مقدار است، بقیه حالت ها در حالت

ماکسیموم خود قرار دارند، یعنی ۱۰۰۰ در مبنای ۸ برابر با ۷۷۷ است.
و به یاد داشته باشید که در بدست اوردن مبنایی مانند ۸ و هر مبنایی به غیر از ۲ و ۱۰ (برای مثال در اینجا هشت آورده شده)، مقدار ۷۷۷ درواقع مقدار متمم کاهش یافته را به شما بر میگرداند.
و همچنین در این مبنا یعنی مبنای۸، همانطور که گفته شد با ۷۷۷ شما مکمل کاهش یافته را بدست خواهید آورد، برای بدست آوردن مکمل یا متمم ۸ بایستی در نظر داشته باشید اگر عدد مبنای هشت شما در انتها دارای صفر بود، مانند ۱۲۰، ۴۰۳۰۰، یا هر عددی که به تعداد صفر ختم شده باشد، در جواب این مسئله با تعداد عدد ۷ در اخر عدد مواجه میشوید، برای داشتن متمم مبنای ۸ همین تعداد ۷ پایانی را کافیست برابر با صفر قرار دهیم و یک عدد به عدد بعد از آن اضافه کنیم
لازم به ذکر است که در هر مبنایی به غیر از ۱۰ و دو در بدست آوردن متمم ها، شما همیشه متمم کاهش یافته را بدست می آورید!
مثال های مهم برای این نکته:

$(123)_8 = 8^3 - 123 = 777 - 123 = 654 reduce. 654 + 1 = 655 radix$

$(1230)_8 = 8^3 - 1230 = 7777 - 1230 =$

$6547 reduce. 6547. 6540 + (4 + 1) \rightarrow 6550$

این مورد تنها در مبنای ۸ نیست بلکه در مبناهای طبیعی دیگر، مانند ۳، ۴، ۵، ... نیز وجود دارد.

نتیجه گیری نهایی اگر، مبنای ۱۰ دادن برای مکمل و مکمل کاهش، از قاعده بومی $r^n - n$ و $(r^n - 1) - n$ استفاده میکنیم که مبنا به توان تعداد ارقام برابر با همان ۱ با تعدادی صفر در جلوش است. در مبنای دو هم که طبق تعریف نیازی به دوباره کاری نیست، و فقط به قاعده ای که در چند خط بالا گفته شد دقت کنید.

متمم اعداد

متمم ۱۰ عدد ۵۴۶۷۰۰ برابر است با

$10^6 - 546700 = 453300$

متمم ۹ عدد ۵۴۶۷۰۰ برابر است با

$(10^6 - 1) - 546700 = 453299$

روش سریع دیگر آن است که اگر به ما مبنا را دادند میتوانیم با یکی کم کردن از آن به مبنای کاهش یافته آن برسیم، و برعکس، اگر متمم مبنای کاهش یافته را به ما دهند میتوانیم با یکی اضافه کردن به آن به متمم مبنا برسیم!

متمم ۲ عدد ۱۱۰۱۱۰۰ برابر است با:

۱۰۸ = ۱۱۰۱۱۰۰

$2^7 - 108 = (20)_{10} = (10100)_2$

متمم ۱ عدد ۱۱۰۱۱۰۰ برابر است با:

$(2^7 - 1) - 108 = (19)_{10 \to 2} = (0010011)_2$

متمم اعداد در مبنای ۲ به سریع ترین روش

نکات

متمم۱: تمام ارقال NOT میشود. متمم ۲: تمام صفر های سمت راست تا زمانی که به اولین یک از سمت راست برسیم همان طور به شکل اول خود باقی می ماند، اما از آن به بعد بیت های بعدی NOT میشود.

متمم ۱ عدد ۰۰۱۱۰۱۱ برابر است با: ۱۱۰۰۱۰۰

متمم ۲ عدد ۱۱۰۱۱۰۰ برابر است با: ۰۰۱۰۱۰۰

این همه گفتیم، پس در سیستم ۱۶ تایی چطوره؟

برای بدست آوردن مکمل ۱۶ و مکمل کاهشی آن (۱۵) بایستی بدانیم که ماکسیموم برای n رقم مثل $16^3 = (1000)_{16}$ اخرین درجه برای مقدار ۱۰۰۰، ۱۵ ۱۵ ۱۵ است که میبایستی این عدد را از مبنای ۱۶ کم کنیم تا آنگاه مبنای کاهش یافته را بدست بیاوریم.

مثال

$$(A86)_{16} = 16^3 = 1000_{16} - A86 = 151515 - 1086 =$$

$$579 Reduce ... 579 (9 + 1 = 10 = A) = 5710 or 57A$$

تمرین

۱. مبنا ۱۰ و مکمل ۹ عدد $(256.73)_{10}$ را بنویسید.

۲. مبنا ۱۰ و مکمل ۹ عدد $(325.12)_{10}$ را بنویسید.

۳. هر دو مقدار ۹ و ۱۰ $(256.73)_{10}$

۴. هر دو مقدار ۹ و ۱۰ $(325.12)_{10}$

۵. هر دو مقدار ۷ و ۸ $(276.35)_8$

۶. هر دو مقدار ۹ و ۱۰ $(9300)_{10}$

۷. $(304000)_8$

۸. $(11110100)_2 = 00001100$

۱.۱۱ اعداد علامت دار

اعداد را میتوان به ۴ روش نمایش داد:

- روش بدون علامت:
عدد را به صورت عادی ارزش گذاری میکنیم و در نهایت نتیجه را مینویسیم

- روش مقدار علامت:
سمت چپ ترین بیت نشان دهنده علامت عدد است.

- نمایش عدد به صورت مکمل یک

- نمایش عدد بصورت مکمل ۲

نکته

- در روش مقدار علامت، بیت صفر نشان دهنده مثبت بودن و بیت یک نشان دهنده منفی بودن است.
- در مکمل یک عدد را به صورت مکمل یک نمایش میدهیم.
- در مکمل ۲ عدد را بصورت مکمل دو نمایش میدهیم.

مثال

عدد ۸ بیتی $(10010100)_2$ راب به مبنای دهدهی تبدیل کنید با فرض اینکه:

۱. این عدد در سیستم بی علامت باشد.
$(10010100)_2 = 4 + 16 + 128 = 148$

۲. این عدد در سیستم علامت مقدار باشد.
$(10010100)_2 = -(4 + 16) = -20$

۳. این عدد در سیستم مکمل ۱ باشد. $(10010100)_2 =$
$-(01101011) = -(1 + 2 + 8 + 32 + 64) = -107 =$

۴. این عدد در سیستم مکمل ۲ باشد.
$(10010100)_2 = -(01101100) = -(4+8+32+64) = -108$

مثال

عدد ۸ بیتی $(01000101)_2$ راب به مبنای دهدهی تبدیل کنید با فرض اینکه:

۱. این عدد در سیستم بی علامت باشد. $(01000101)_2 = 69$

۲. این عدد در سیستم علامت مقدار باشد. $(01000101)_2 = +69$

۳. این عدد در سیستم مکمل ۱ باشد.
$(01000101)_2 = +(10111010) = 2+8+16+32+128 = +186$

۴. این عدد در سیستم مکمل ۲ باشد.
$(01000101)_2 = +(10111011) = +187$

نکته مهم

گاهی ممکن است بخواهیم عددی را منفی اش را به صورت دودویی بنویسیم اما باید یک نکته مهمی را مورد نظر داشته باشیم، برای مثال اگر از ما بخواهند که عدد ۹ را نمایش دهیم آن را به صورت $(1001)_2$ نشان میدهیم، اما اگر از ما بخواهند که منفی این عدد را نمایش بدهیم نمی توان آنرا به این صورت $(11001)_2$ نشان داد، در حقیقت بازهم به

جواب ۹- خواهیم رسید اما در ۵ بیت معنایی ندارد، چرا که کامپیوتر همه اعداد در در تعدادی از توان های دو نمایش میدهد مانند ۴ بیت، ۸ بیت، ۱۶، ۳۲، ۶۴، ۱۲۸ و غیره. پس برای نشان دادن عدد -۹ باید به این صورت بنویسیم: $(1000, 1001)_2 = (-9)_{10}$

۱۲.۱ عملیات روی اعداد بدون علامت در مبناهای مختلف

انسان برای شمارش و انجام عملیات ریاضی (جمع و تفریق و ضرب و تقسیم) از مبنای ۱۰ استفاده میکند، دلیل این انتخاب توسط انسان، تعداد انگشت های دست او بود. جدول ضرب هم بر اساس مبنای ۱۰ نوشته شده است. امااگر ما ۸ انگشت داشتیم مجبور بودیم از مبنای ۸ استفاه کنیم. در این صورت دیگر جمع و تفریق ما بر اساس مبنای ۸ است که جواب هایی که از مبنای ۱۰ در محاسبات ریاضی بدست می آوریم در مبنای ۸ بسیار متفاوت است. یعنی ۷ x ۶ != ۴۲، ۱۲ - ۵ != ۷، ۷ + ۱ != ۸. پس میفهمیم که محاسبات در مبناهای غیر ۱۰ برای ما بسیار سخت و دشوار است. زیرا ما وقتی عملیات ساده مبناهای غیر از ۱۰ را حفظ نیستیم پس میتوانیم مسائل را به زبان خودمان یعنی مبنای ۱۰ ترجمه کنیم و در همین مبنا محاسبات را انجام دهیم و سپس به مبنای خواسته شده توسط مسئله تبدیل میکنیم. در مورد عملیات جمع و تفریق، میتوانیم از مبنای ۱۰ استفاده نکنیم و از جدول کمک بگیریم.

جمع اعداد در مبناهای غیر ۱۰

تمرین:

- $(101101)_2 + (010111)_2 =$
- $(276)_8 + (357)_8 =$
- $(276)_{10} + (357)_{10} =$
- $(2A58)_{16} + (71D0)_{16} =$
- $(2F2C)_{16} + (2FAA)_{16} =$

جمع دو عدد در مبنای ۲

$$(111101)_2 + (10111)_2 = (1010100)_2$$

در هنگام جمع دو عدد باینری باید توجه داشت که اگر جمع اول با دومی بیشتر از ۱ شد، در حقیقت عدد بدست آمده در مبنای ۱۰ است، به همین خاطر این عدد را بایستی به مبنای دو سریعا تبدیل کنیم، و با بخش بعدی به جمع بپردازیم. بعد از اینکه حاصل خود را به صورت عدد دودویی بدست اوردیم، برای بررسی آن میتوانیم، صورت اول جمع را به سیستم دهدهی و صورت دوم هم همینطور تبدیل کرده و سپس باهم جمع کنیم، و در نهایت حاصل را به مبنای ده برده و در آخر بررسی برابر خود را انجام

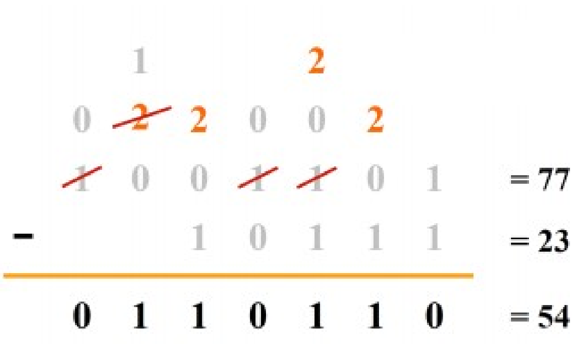
میدهیم، این بررسی نشان دهنده آن است که حاصل بدست آمده در مبنای دو چقدر امکان خطا دارد، در ادامه صحبت خواهیم کرد.

۱۳.۱ تفریق دو عدد بی علامت در مبنای ۲ به روش مستقیم (قرض گرفتن)

در این نوع محاسبه تفریق همانند تفریق سیستم دهدهی عمل میکنیم، همان طور که در سیستم دهدهی هرگاه به عدد ۰ میرسیم که بر روی عددی غیر ۰ تفریق کنیم به صفر، ده عدد قرض میدادیم و از خانه بعدی صورت یکی کم میکردیم، در تفریق مبنای دو هم، دقیقا همچنین اتفاقی رخ می دهد، شما زمانی که در صورت به عدد صفر میرسید که در عدد پایینی عددی غیر صفر قرار دارد، تا سقف دو، به عدد صفر قرض میدهید و یکی از خانه بعدی عدد کم میکنید و عمل تفریق در مبنای دو هم به آسانی صورت خواهد گرفت

مثال

$(1001101)_2 - (10111)_2 = (0110110)_2$



شکل ۱: مثالی از تفریق در مبنای دو

۱۴.۱ جمع و تفریق اعداد علامت دار

به روش های متداول قبلی قابل انجام است.
معمولا در کامپیوتر در سیستم مکمل ۲ انجام می شود.
بایستی همه عملوند ها را به سیستم مکمل ۲ ببریم
جواب نهایی در سیستم مکمل ۲ بدست خواهد آمد، باید برای خواندن آن دقت کنیم.

۱.۱۴.۱ جمع دو عدد علامت دار در سیستم مکمل ۲

جمع در سیستم مکمل ۲، بدون توجه به مثبت یا منفی بودن اعداد، آنها را زیر هم نوشته و جمع میکنیم، از رقم نقلی خروجی صرف نظر میکنیم یعنی آنرا حذف می کنیم.
مثال: حاصل عملیات جمع را در ۴ بیت در سیستم مکمل دو حساب کنید:
نکته ای که در این مسئله باید به آن توجه داشته باشید، آن است که در هنگام جمع به صورت بی علامت پیش میرویم، برای بررسی هر کدام از صورت جمع ها، آن عددی که بیت علامتش ۰ است که به صورت دهدهی عادی تبدیل میشود، در غیر این صورت بایستی اول به صورت مکمل ۲ نوشته شده و بعد به مبنای ۱۰ تبدیل شود.

۱. $(0001)_{2_{(+1)}} + (1001)_{2_{(-7)}} = (1010)_{2_{-(6)}}$

۲. $(0010)_{2_{(+2)}} + (0111)_{2_{(7)_{2+7=9}}} = (1001)_{2_{-(0111 \to -7)}}$

در مثال بالا در حقیقت Overflow رخ داده که جواب اشتبا بدست آمده است. و فلگ کری ۱ خواهد شد چرا که یک بیت اضافی دارد.

۳. $(0011)_{2_{(+3)}} + (0100)_{2_{(+4)_{+3+4=7}}} = (0111)_{2_{+(7)}}$

۴. $(1001)_{2_{(-7)}} + (0111)_{2_{(+7)_{7-7=0}}} = ([1]0000)_{2_{(0)}}$

در مثال بالا، [۱] به این خاطر حذف شده چرا که حاصل بدست آمده بیشتر از ۴ بیت میشد.

نکته:

در هنگام جمع دو عدد در مبنای دو، اگر برای مثال صورت اول و صورت دوم هر دو مثبت باشند و در نهایت در نتیجه عددی منفی را داشته باشیم، اورفلو رخ خواهد داد، برای تصحیح این مشکل فقط کافیه است که از

روش Sign Extended استفاده کنیم، که اگر ۴ بیتی باشد می شود، هشت بیت، اگر هشت بیتی باشد میشود ۱۶ بیت الی آخر. این طور می توان به جواب درست دست پیدا کرد(تکرار عدد اخر).
برای مثال، در تمرین دوم بالا، ما اورفلو داریم چرا که نتیجه بدست آمده با جمع کلی درست نیست، یا اینطور جمع دو عدد مثبت باید مثبت شود اما نتیجه منفی شده است، برای بدست آوردن نتیجه درست از Extended Sign استفاده کنیم:

$$(0000,0010)_2 + (0000,0111)_2 = (0000,1001)_2$$

در بالا بیت علامت صفر، اورو فلو صفر، فلگ صفر، صفر و کری هم صفر خواهد بود، و نتیجه بدست امده هم مثبت.

۱۵.۱ Overflow سرریز

اعداد در کامپیوتر با طول محدود و تعداد بیت های مشخص به کاربرده می شوند.اگر نیتجه محاسبات خارج از این محدوده شود و بیت های بیشتر در دسترس نبایشد، این بیت های اضافی حذف خواهند شد و نتیجه بدست آمده صحیح نخواهد بود. در این حالت می گوییم در انجام محاسبه سرریز اتفاق افتاده است.
فلگ v یا of $(Overflowflag)$: این فلگ وقتی ۱ میشود که از نتیجه محاسبات در بازه مجاز تعداد بیت نباشد، در این حالت میگوییم اورفلو یا

سرریز رخ داده است.
در ادامه مطالب حتما مثال هایی در این خصوص زده خواهد شد.

۱۶.۱ بازه مختلف ساخت اعداد در سیستم های مختلف با کمک n بیت

در سیستم بی علامت:

$0-(111...1)_2=2^n-1$

در سیستم علامت و مقدار:

$(111...1)_2=-(2^{n-1}-1)-(111...1)_2=(2^{n-1}-1)$

سیستم مکمل یک:

$(111...1)_2=-(2^{n-1}-1)-(111...1)_2=(2^{n-1}-1)$

سیستم مکمل دو:

$(111...1)_2=-(2^{n-1})-(111...1)_2=(2^{n-1}-1)$

۱۷.۱ تشخیص سرریز دو عدد بدون علامت

در جمع اعداد در سیستم بدون علامت، اگر پس از جمع دو علامت رقم، رقم اخر (نقلی و نهایی) یک شود، سرریز اتفاق خواهد افتاد.

$(1101)_2+(1100)_2=([1](1001))_2$

۱۸.۱ تشخیص سرریز در اعداد علامت دار مکمل ۲

نکته:

اگه جمع دو عدد منفی، مثبت شود یا جمع دو عدد مثبت منفی شود، سرریز رخ میدهد. دقت داشته باشید، جمع دو عدد مثبت و منفی باهم، سرریز ندارد.

اگر دو عدد A و B در سیستم مکمل ۱ و ۲ باشند، آنگاه A+B در صورتی سرریز خواهد بود که:

۱. اگر A و B هر دو مثبت باشند و نتیجه منفی را بدهند.
۲. اگر A و B منفی باشند و نتیجه مثبت را بدهند.

نتیجه:

اگر دو عدد با علامت های مختلفی داشته باشیم هیچگاه اورفلو رخ نخواهد داد.

اگر دو عدد A و B در سیستم مکمل ۱ یا ۲ باشند، آنگاه A-B در صورتی سرریز خواهد بود که:

۱. A مثبت باشد و B منفی باشد، و حاصل آنها منفی شود.
درستش:
$(A_+) - (B_-) = (+)$

۲. A منفی باشد و B عددی مثبت، حاصل آن مثبت شود
درستش:
$(A_-) - (B_+) = (-)$

دیگر نکته بر روی دیگر فلگ های سادست که مختصرا بیان میکنیم:
فلگ zero زمانی ۱ میشود که حاصل بدست آمده برابر با صفر باشد در غیر این صورت ۰ خواهد بود.
فلگ carry زمانی ۱ میشود که بیت اضافی وجود داشته باشد.
فلگ sign زمانی یک میشود که بیت علامت ما منفی باشد.
فلگ اورفلو زمانی یک میشود که حاصل بدست آمده با جمع منطقی ما (اگر عدد صورت اول منفی بود به مکمل ۲ میبریم در غیر این صورت بصورت ساده جمع را انجام میدهیم) برابر نباشد.(با توجه به قوانینی که در بالاتر توضیح داده شد)

حل چند تمرین:

۱. $(1001)_{2_{2c(-7)}} + (0111)_{2_7} = ([1]0000)_{2_{(0)}}$

carry = ۱، Zero = ۱، Of = ۰، sign =۰

۲. $(1101)_{2_{2c=(-3)}} + (1110)_{2_{2c=(-2)}} = ([1]1011)_{2_{2c=(-5)}}$

carry = ۱، Zero = ۰، Of = ۰، sign =۱

۳. $(0101)_{2_{(5)}} + (0100)_{2_4} = (1001)_{2_{2c=(0111=>-7)}}$

carry = ۰، Zero = ۰، Of = ۱، sign =۱

۴. $(1001)_{2_{2c=(-7)}} + (1010)_{2_2c=-6} = ([1]0011)_{2_{(3)}}$

carry = ۱، Zero = ۰، Of = ۱، sign =۰

۱۹.۱ تفریق دو عدد علاممت دار در سیستم مکمل ۲

در هنگام تفریق دو عدد در مکمل ۲، میتوانیم عدد A را با مکمل عدد B جمع کنیم. و این جمع مانند جمعی است که در بالاتر توضیح داده شد.

حل چند تمرین:

+ 6	00000110	− 6	11111010
+13	00001101	+13	00001101
+19	00010011	+ 7	00000111
+ 6	00000110	−6	11111010
−13	11110011	−13	11110011
− 7	11111001	−19	11101101

شکل ۲: تمرین از جمع اعداد در سیستم مکمل ۲

۱. $(1101)_2 - (1001)_2 \rightarrow (1101)_{2_{2c=(-3)}} + (0111)_7 = ([1]0100)_2$

carry = ۱، Zero = ۰، Of = ۰، sign =۰

۲. $(1100)_2 - (1010)_2 \rightarrow (1100)_{2_{2c=(-4)}} + (0110)_6 = ([1]0010)_2$

carry = ۱، Zero = ۰، Of = ۰، sign =۰

۳. $(1001)_2 - (0100)_2 \rightarrow (1001)_{2_{2c=(-7)}} + (1100)_{2c(-4)} =$
$([1]0101)_2$

carry = ۱، Zero = ۰، Of = ۱، sign =۰

مثال:

با فرض دو عدد دودویی x = ۱۰۱۰۱۰۰ و Y = ۱۰۰۰۰۱۱ تفریق های زیر را انجام دهید.

۱. $X - Y$

حل:

$(1010100)_2 - (1000011)_2 \rightarrow$

$(1010100)_{2_{2c(-44)}} + (0111101)_{2_{61}} = ([1]0010001)_{2_{17}}$

۰= sign ،۰ = Of ،۰ = Zero ،۱ = carry

۲. $Y - X$ حل:

$(1000011)_2 - (1010100)_2 \rightarrow$

$(1000011)_{2_{2c(-61)}} + (0101100)_{2_{44}} = (1101111)_{2_{-17}}$

۱= sign ،۰ = Of ،۰ = Zero ،۰ = carry

پس در هنگام تفریق دو عدد $A - B$ خواهیم داشت:

اگر $A > B$ پس عددی مثبت خواهیم داشت، و فلگ سر ریز با فلگ علامت صفر خواهد بود.

اگر $A < B$، پس عددی منفی خواهیم داشت که هم فلگ علامت و هم فلگ اورفلو ۱ خوهد بود.

اگر $A = B$، در این صورت هر دو یکسان هستند پس جواب صفر را

خواهیم داشت که فلگ Zero روشن خواهد شد.

۲۰.۱ سیستم D2B

این نوع اعداد به زبان خودمانی، فقط در نقش مبنای دو ظاهر می شوند وگرنه از نظر ماهیتی همان اعداد Decimal هستند، این اعداد زمانی مورد استفاده قرار میگیرند که برای تبدیل اعداد Decimal به آنها جایگاه اعداد برایمان مهم نیست، بلکه معنا و مفهوم آن عدد برایمان مهم است، مثلا شما شماره دانشجویی یا یک شماره تلفن یک منطقه را در نظر بگیرید، که هر کدام از اعداد نشان دهنده و به معنای خاصی هست، این اعداد هیچ لزومی ندارد که صرفا دودویی باشند، برای تبدیل اعداد Decimal به اعداد Arithmetic به چهار بیت به ازای هر عدد نیاز است، مانند تبدیل اعداد سیستم Hex. این اعداد قابل درک هستند، نسبت به اعداد باینری فضای بیشتری را میگیرند و اصلا برای محاسبات جمع و تفریق مناسب نیستند!

۱.۲۰.۱ کد کردن اعداد دهدهی

- کدهای باید به صورت دودویی باشند، زیرا در کامپیوتر همه چیز صفر و یک است.

- کدها فقط نماد یا سمبل نمایش اطلاعات را عوض میکنند و نه مفهوم آن ها را.
- یک کد دودویی n بیت، 2^n ترکیب ممکن از یک ها و صفرها را داراست.

کد ها به دو دسته تقسیم می شوند:
کدهای وزن دار: به هر مکان یک وزن اختصاص داده می شود مثل کد BCD، که دارای وزن 8421 است.
کد های بدون وزن: مثل کد افزودنی ۳، که میتوانیم با اضافه کردن ۳ عدد به کد BCD به کد Excess-۳ رسید.

در سیستم کد، $8, 4\ \bar{2}\ \bar{1}$ شما می توانید طبق وزن ها از کد BCD به کد مورد نظر خود برسید.

در سیستم کد، ۲۴۲۱ از صفر تا ۴، وزنمان را از سمت راست تعیین میکنیم، اما از خود پنج به بعد از سمت چپ به عنوان تعیین وزن استفاده خواهیم کرد.

مثال

عدد $(82)_{10}$ را به صورت، مبنای ۲، BCD، Excess-۳ ، $8\ 4\ \bar{2}\ \bar{1}$، ۲۴۲۱،

Decimal Digit	BCD 8421	Excess-3	84-2-1	2*421	Biquinary 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

شکل ۳: سایر کد های Decimal

کد کنید.

۱. $(82)_{10\to 2} = (1010010)_2$

۲. $(82)_{10} = (1000,0010)_{BCD}$

۳. $(82)_{10} = (1000,0010)_{BCD\to(excess-3)+3} =$
$(1011,0101)_{Excess-3}$

۴. $(82)_{10} = (1000,0010)_{BCD\to 84\overline{21}} = (1000,0110)_{84\overline{21}}$

۵. $(82)_{10} = (1000,0010)_{BCD\to 2421} = (1110,0010)_{2421}$

عدد $(10111001)_2$ را به صورت، مبنای ۱۰، BCD ، Excess-۳ ، $4\,\bar{2}\,\bar{1}$ 8، ۲۴۲۱، کد کنید.

۱. $(10111001)_{2\to10} = (185)_{10}$

۲. $(185)_{10} = (0001, 1000, 0101)_{BCD}$

۳. $(185)_{10} = (0001, 1000, 0101)_{BCD\to(excess-3)+3} =$
$(0100, 1011, 1000)_{Excess-3}$

۴. $(185)_{10} = (0111, 1000, 1011)_{84\overline{21}}$

۵. $(185)_{10} = (0001, 1110, 1011)_{2421}$

عدد دودویی $(10111001)_2$ را به صورت BCD بازنویسی کنید.
$(10111001)_2 = (185)_{10} = (0001, 1000, 0101)_{BCD}$

۲.۲۰.۱ جمع دوعدد BCD

برای جمع اعداد BCD فقط کافیه است که معادل Decimal آنها را با هم جمع کنیم و بعد از آن حاصل را به BCD تبدیل خواهیم کرد.

۲۱.۱ کد گری یا کد انعکاسی

زمانی پیش می آید که عدد ورودی ما مثلا 0111 که در مبنای ۱۰ برابر با هفت است به عنوان ورودی وارد شده و مثلا میخواد بشود ۸ یا 1000 در طی این تبدیل ممکن است تاخیر ها و Delay های پیش آمده باعث شود به این تبدیل چندین بیت با هم تفاوت و فاصله ایجاد شود، این باعث میشود که CPU زحمت زیادی بکشد در فرایند های تبدیل، به همین خاطر سیستم عددی به نام Gray به وجود آمد که در اثر بوجود آمدن این Delay ها فاصله بین هر عدد تنها یک بیت باشد، کد گری میتواند یک ست ۲ بیتی، ۳ بیتی ۴ بیتی، ۵بیتی و غیره باشد، که همه این ها با هم تنها یک بیت فاصله دارند، منظور از یک بیت فاصله یعنی بعد از تبدیل عدد اصلی از حالت بومی خودش به کد گری فقط و فقط یک بیت تفاوت است.

برای تبدیل کد باینری به کد گری خیلی راحت میتوانیم از طریق شباهت و تفاوت پیش برویم، عددی که بیشترین ارزش را دارد (اولین عدد از سمت چپ) یا عدد MSB را بدون هیچ دستکاری می نویسیم، بعد از آن خود آن عدد را با عدد کناری خودش بررسی میکنیم، اگر بایکدیگر مشابه بودند، عدد صفر را مینویسیم، اگر باهم تفاوت داشتند عدد یک را مینویسیم. همین فرایند را تا انتها پیش میرویم. برای مثال:
$Same = 0$

$Difference = 1$

$(110111)_{2 \to Gray} = (101100)_{Gray}$

$(111)_{2 \to Gray} = (100)_{Gray}$

$(10101101)_{2 \to Gray} = (11111011)_{Gray}$

۱.۲۱.۱ تبدیل باینری ۴ بیتی به گری ۴ بیت

Binary	Gray
۰۰۰۰	۰۰۰۰
۰۰۰۱	۰۰۰۱
۰۰۱۰	۰۰۱۱
۰۰۱۱	۰۰۱۰
۰۱۰۰	۰۱۱۰
۰۱۰۱	۰۱۱۱
۰۱۱۰	۰۱۰۱
۰۱۱۱	۰۱۰۰
۱۰۰۰	۱۱۰۰
۱۰۰۱	۱۱۰۱
۱۰۱۰	۱۱۱۱
۱۰۱۱	۱۱۱۰
۱۱۰۰	۱۰۱۰
۱۱۰۱	۱۰۱۱
۱۱۱۰	۱۰۰۱
۱۱۱۱	۱۰۰۰

۲.۲۱.۱ تبدیل کد گری به کد باینری

برای تبدیل کد گری به باینری میباست عدد MSB را نوشته (همانطوری) و بعد از همان به بعد با اعداد بعدی مقایسه میشود

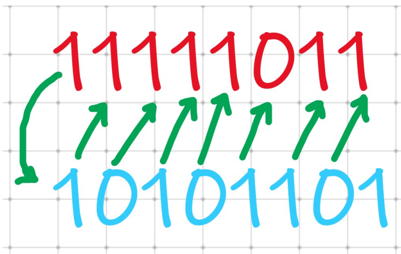
$Same = 0$

$Difference = 1$

$(101100)_{Gragy \to 2} = (110111)_2$

$(100)_{Gragy \to 2} = (111)_2$

$(11111011)_{Gragy \to 2} = (10101101)_2$



شکل ۴: Gray ۲ Binary

نکته:

- کدی که فاصله d دارد میتواند $d-1$ خطا را تشخیص دهد.
- کدی که فاصله d دارد میتواند $\lfloor \frac{d-1}{2} \rfloor$ را تحصیح کند.

۲۲.۱ ASCII Codes

از کدهای اسکلی زمانی استفاده میشود که ما بخواهیم نماد ها، کارکتر های مختلف، اعداد، نماد های زبان های مختلف و غیره را در سیستم خود نمایش بدهیم. این سیستم کدینگ در حالت کلی ۸ بیتی میباشد. در ۵۰ سال گذشته این سیستم ۶ بیتی بوده و توانایی نمایش حروف فقط بزرگ انگلیسی را داشت، اما بعد از آنکه این سیستم ۸ بیتی شد توانایی هایی که در بالاتر توضیح داده شد را دارا میباشند، در این سیستم کدینگ ۳ بیت به عنوان ستون ها و ۴ بیت به عنوان سطر ها مورد استفاده قرار گرفته، و بیت اخر که بیت مشخص کنند ماهیت کارکتر میباشد، اگر وضعیت این بیت ۰ باشد توانایی نمایش اعداد، کارکتر های کنترلی و یکسری علائم و نماد های مختلف مانند $(`BackText)$، :، ؛، "، را دارای میباشد که به این دسته ها Standard ASCKII گفته میشود، که **تعداد کرکتر های آن ۱۲۸ تا** می باشد. اما اگر این بیت هشتم ۱ باشد علاوه بر نمایش حروف انگلیسی میتواند حروف اضافه بر زبان انگلیسی را نمایش دهد مانند در زبان ترکی و آلمانی (اوملات) و غیره. که این دسته در حقیقت ASCKII

Extended نامیده میشود، که این دسته هم **دارای ۱۲۸ تا کرکتر** میباشد.
دسته بندی های کدهای الفبایی یا بطور کلی $Textual$:

Alphabet: a...z and Z ... A

Digits: ۹..۱

Special: Symbols ‘ ،” ،؛ ،: ،.

non-printable: NULL DEL،

نوع	ستون اعداد			سطر اعداد			
7	6	5	4	3	2	1	0

شکل ۵: بیت های کد اسکی

مهم ترین بخش جدول کدهای اسکی مربوط به ستون اول و دوم و ابتدای ستون سوم، مربوط به کارکتر های کنترلی میشود، حالا این کاراکتر های کترلی چیست این مهم است:

در گذشته ماشین های تله تایپی وجود داشتند که نهایت سرعت آنها ۱۰ بیت در ثانیه بود و کلا یکسری دستگاه های مکانیکالی بودند. این دستگاه های برای ارتبط با سیستم های دیگر از یکسری کرکتر ها استفاده میکردند تا بعضی از جریان ها مورد بررسی قرار بگیرد، برای مثال وقتی به این سیستم ها قرار بود یکسری کلمات و جلملات را بنویسند ارتباط

	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC_1	!	1	A	Q	a	q
0010	STX	DC_2	“	2	B	R	b	r
0011	ETX	DC_3	#	3	C	S	c	s
0100	EOT	DC_4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	O	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

شکل ۶: ASCII Codes Table

آنها مانند ارتباط ما با کامپیوتر های امروزی نبود و با تاخیری کلمات چاپ میشدند، این سیستم ها برای اینکه ببینند آیا سیستم مکانیکی توانسته کراکتر های قبلی را بنویسد که کارکتر های بعدی را ارسال کند، یا اینکه دست نگهدارد که آن سیستم مکانیکی کار قبلی اش را تمام و سپس بروی کار جدید برود، در این میان از مجموعه ای از کارکتر های کنترلی استفاده میکردند که کنترل جریان را هدایت می نمودند. بطور کلی حفظ کردن نام برخی از آنها دشوار میباشد به همین دلیل در سیستم های امروزی از کنترل ترکیبی بجای استفاده از آن کرکتر ها استفاده کردند، برای مثال کرکتر ESC معادل $ctrl + [$ است.

برای بدست اوردن نام و کلا تایپ این نوع کرکتر ها باید دقت داشته

باشیم که از سمت چپ اعداد خود را قرار بدهیم، **اولین دسته عدد که سه بیتی** است ستون را منظور دارد، **دومین دسته که چهار بیتی** است بعد از سه بیت ستون نوشته شده و قرار میگیرد و در انتها برای اینکه بررسی کنیم آیا این عدد یک Extended است یا Standard از سمت چپ بیت MSB را یکی اکستند میکنیم که تا نوع آن مشخص و ۸ بیتی شود. و بعد در انتها میتوانیم کد بانری بدست آمده را به دسیمال تبدیل کرده و معادل آن عدد را در کد اسکی بررسی کنیم. مانند:

$$1100111 \rightarrow (110, 111)_{(2)\rightarrow 10} = 103\ g$$

	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC_1	!	1	A	Q	a	q
0010	STX	DC_2	“	2	B	R	b	r
0011	ETX	DC_3	#	3	C	S	c	s
0100	EOT	DC_4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	O	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

شکل ۷: ASCII Codes Sample

یا اینکه میتوانیم بطور کلی اینگونه پیش برویم که ۳ بیت اول از ستون و ۴ بیت دوم از سطر جدول را نقطه یابی کنیم که معادل چه حرفی میشود. تصویر بالا

۲۳.۱ کد های تشخیص و تصحیح خطا

هنگام انتقال داده ها ممکن است در آنها خطایی به علت تداخل های الکترومغناطیسی، حرارت زیاد و غیره به وجود آید. میتوان کد هایی طراحی کرد که خطا را تشخیص و تحصیح کنند.
یکی از ساده ترین روش های تشخیص خطا استفاده از بیت توازن یا **Parity** است. میتوان به هر کلمه یک بیت اضافه کرد به طوری که تعداد بیت های یک آن مثلا فرد شود

۱.۲۳.۱ کدهای تشخیص خطا

سیستم توازن، یکي از ساده ترین روش ها برای تشخیص خطاست، که یک بیت توازن به اطلاعات اضافه می شود.
در حالت کلی دو نوع توازن وجود دارد، توازن زوج و توازن فرد.
توازن زوج Even Parity
یک بیت به اطلاعات اضافه میشود تا تعداد کل ۱ ها کد زوج جود.

$0110010 = [1]0110010$

$0100100 = [0]0100100$

توازن فرد Odd Parity

$0110011 = [1]0110011$

$0100101 = [0]0100101$

یک بیت به اطلاعات اضافه میشود تا تعداد کل ۱ های کد فرد شود. تا تعداد صفر ها و یک ها به تعادل و برابری برسد.

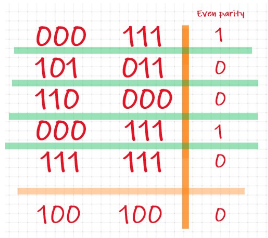
نوع دیگری از هاParity وجود دارد که به تعداد یک ها توجه می شود، یعنی اگر سیستم پریتی زوج بودیم تعداد یک ها بایستی زوج باشند، اما اگر در سیستم پریتی فرد بودیم باید تعداد یک ها یک دسته فرد باشند، برای اینکار در سیستم پریتی زوج تعداد یک ها اگر فرد بود یک، ۱ را اضافه خواهیم کرد اما اگر زوج بود ۰ را وارد میکنیم. در فرد هم همینگونست اگر سیستم فرد تا ۱ داشت ۰ اگر سیستم زوج تا ۱ داشت یک، ۱ اضافه میکنیم تا نظم زوجی یک ها را بهم بزنیم و آنها را فرد کنیم!

$EvenParity : 100011 = 1 \rightarrow 1000111$

$OddParity : 100011 = 0 \rightarrow 1000110$

۲۴.۱ Overlapping Parity or Block Parity

یک مجموعه ای از دیتا ها که بایستی به دو صورت افقی و عمودی از آنها پریتی گرفته شود.



شکل ۸: Block Parity

در صورتی که در یکی از بیت های مجموعه داده های بالا خطایی ایجاد شود میتوان براحتی دریافت که در آن قسمت، هم به صورت افقی و هم به صورت عمودی نتیجه پریتی با نتیجه قبلی برابر نخواهد بود. و برای نشان داده اشتباه در آن متقطه باید به صورت یک آرایه دو بعدی طور عمل کنیم که بعد اول در مورد سطر و بعد دوم در مورد ستون میباشد، مانند اشکال در $M_{[5][3]}$.

۲۵.۱ Parity Checksum

در این نوع پریتی ما سه نوع داریم:

۱.۲۵.۱ Single-precision checksum

در این نوع از checksum دیتا های دریافتی را با هم جمع میکنیم، و حاصل بدست آمده اگر همراه با Carry بود از آن صرف نظر خواهیم کرد.

۲.۲۵.۱ Double-precision checksum

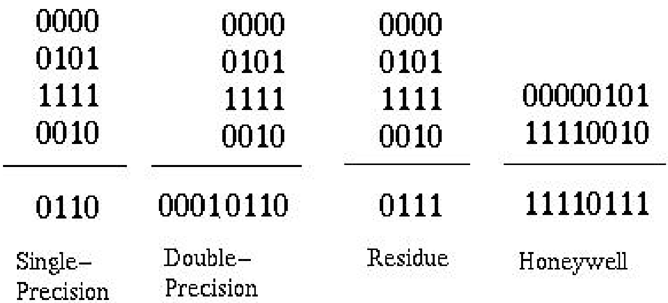
در این نوع، علاوه بر اینکه کری را به همراه حاصل مینویسیم بایستی بررسی کنیم که وجود کری آیا دسته بیت ها را در توانی از دو قرار داده است یا خیر، اگر نبود خودمان دستی این کار را انجام میدهیم

۳.۲۵.۱ Residue checksum

در این نوع از checksum، مانند Single-precision عمل میکنیم با این تفاوت که کری بدست آمده را در حاصل جمع میکنیم

۴.۲۵.۱ HoneyWell checksum

در این در صورتی که ۴ عدد دیتا را داشته باشیم دو به دو دسته ها را بهم می چسبانیم.



شکل ۹: Parity checksum

۲۶.۱ کد همینگ یا Hamming Code

بطور کلی، کد همینگ، کد تشخیص و تصحیح خطا می باشد. که در مخابرات مورد استفاده قرار میگیرد. منظور از مخابرات یعنی ارسال و دریافت اطلاعات بین دو قسمت مبدا و مقصد است. که برای اولین بار به افتخار ریچارد همینگ معرفی شد. در مفهوم، در کد همینگ برای شناسایی و تصحیح خطا دسته ای از کد ها در اطلاعات ارسال میشود، که به این کدها کد همینگ می گویند.

۱.۲۶.۱ فاصله همینگ

زمانی که ما کد تحصیح شده را در کنار کدی با بیت های غلط قرار می دهیم به اختلاف بین این دو عدد، فاصله همینگ گفته می شود.

$A = 111001 \rightarrow Correct$

$A_{uncorrect} = 010110 \rightarrow uncorrect \rightarrow distance = 5$

۲.۲۶.۱ نحوه بدست همینگ کد و تشخیص و تصحیح خطا

برای بدست آوردن کد همینگ یک عدد میبایستی به شکل زیر عمل کنیم:

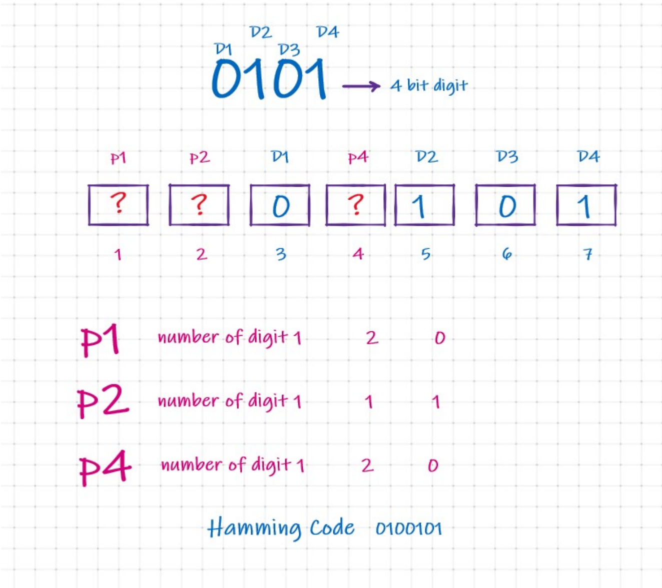
- اول باید بررسی کنیم که اعداد دریافت شده چند بیتی است، (تشخیص از تعداد ارقام عدد ارسالی)

- بعد از بدست آوردن تعداد ارقام عدد ارسالی، از سمت چپ باید کد هایی تحت عنوان Code Parity را در نظر داشته باشیم، هر کدام از این کدهای Parity توانی از دو خواهند بود. یعنی $2^0, 2^1, 2^2 Or 1, 2, 4, 8, 16, 32, etc$

- باتوجه به قاعده بالا، از سمت چپ جایگاه های مثلا ۱ و ۲ و ۴ و ۸ و غیره را برای کد های Parity نگه میداریم، و اعداد داده اصلی را لا به لای این اعداد Parity خواهیم نوشت!

با یک مثال نحوه بدست آوردن کد همینگ را متوجه خواهید شد:

کد همینگ برای عدد ۰۱۰۱ به چه شکل خواهد بود؟



شکل ۱۰: مثال بدست آوردن کد همینگ

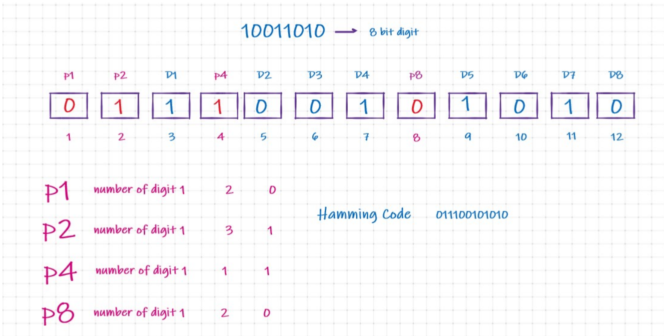
توضیح حل مثال بالا:

در این مثال ما چهار بیت داریم، و کدهای parity ۱ و ۲ و۴ خواهیم داشت، با توجه به هر عدد بلاک های خالی برای قسمت های ۱ و ۲ و ۴ خواهیم گذاشت تا با روشی مناسب اعداد مناسب آنها را بدست بیاوریم و در قسمت های خالی یا بین این پریتی ها دیتای اصلی خود را خواهیم گذاشت، بعد از تنظیم تمام جایگاه ها و قرار دادن اعداد اصلی در جایگاه مناسب خودشان، با روشی ساده میتوان اعداد پریتی را بدست آورد، اینکه شما میتوانید از جایگاه اول که ۱ است یک در میان اعداد ۱ و ۰ را در نظر بگیرید، صفرها مهم نیستند بلکه باید تعداد یک ها را داشته باشیم، **اگر** تعداد یک ها **زوج** بود آن پرتی ۰ خواهد بود اگر **فرد** بود ۱ می باشد، بعد از جایگاه اول به جایگاه دوم یعنی ۲ میریم، که میبایست همان کار اول را از دو، با تفاوت ۲ در میان انجام دهیم تا مقدار صفر یا یک آن را بدست بیاوریم. بعد به جایگاه چهارم میرویم و بایستی از عدد جایگاه ۴، ۴ در میان تعداد یک ها را مورد بررسی قرار دهیم.

نکته: گاهی ممکن است به ۸ در میان یا بیشتر از آن، هم برخورد کنید، اگر در انتها به اندازه مناسب n در میان، عددی وجود نداشته باشد، همان تعدادی که وجود دارند را میتوان به عنوان دسته آخر مورد بررسی قرار داد.

مثال:

کد همینگ برای عدد ۱۰۰۱۱۰۱۰ به چه شکل خواهد بود؟



شکل ۱۱: مثال بدست آوردن کد همینگ

۳.۲۶.۱ پیدا کردن عدد غلط و تصحیح آن

نکته دیگری هم هست!

ممکن است عددی را به شما دهند که یا اشتباه است یا درست، که شما بایستی با عمل همینگ درستی یا نادرستی آن عدد ارسالی را بررسی کنید.

مثال:

در عدد ۰۱۱۰۱۰۱ مشخص کنید که کجا خطا رخ داده است؟

در این مثال دیگر کاری به اضافه کردن جایگاه نداشته باشید با این فرض پیش بروید که تمام جایگاه ها همانی است که میدانید:

$0_{p1}1_{p2}10_{4}101$

$p1 : numberof1 : 3 \to 1$

$p2 : numberof1 : 2 \to 0$

$p4 : numberof1 : 2 \to 0$

با توجه فرایندی که در بالا صورت گرفت میتوان نیتجه گرفت که پریتی های بدست آمده با عدد نوشته شده مسئله در جایگاه های مناسب، یا درست است یا غلط، که مشخص کردیم که جایگاه اول و دوم غلط نوشته شده، این موضوغ غلط بودن این عدد را نشان نمیدهد، برای بدست اوردن اشکال در عدد بایستی جایگاه اعداد غلط مانند جایگاه یک و دو را باهم جمع کنیم که می شود ۳، آنگاه از این نتیجه میتوان دریافت که در جایگاه سوم عدد درستی درج نشده، اگر صفر است آنرا یک میکنیم، اگر یک است آنرا صفر میکنیم. نکته ای که لازم است در انتها به آن اشاره کنم آن است که در هنگام شمارش تعداد یک های، اگر جایگاهی از خودش یک داشت آنرا حساب نخواهیم کرد.

عدد درست، ۰۱۰۰۱۰۱

مثال:

عدد ۰۱۱۱۰۰۱۰۱۱۱۰ درست دریافت شده یا غلط؟ در صورت تشخیص بیت اشتباه آنرا مشخص کنید:

$$0_{p1}1_{p2}11_{p4}0010_{p8}1110$$

$$p1 : numberof1 : 4 \rightarrow 0Correct$$

$$p2 : numberof1 : 4 \rightarrow 0Uncorrect$$

$$p4 : numberof1 : 1 \rightarrow 1Correct$$

$$p8 : numberof1 : 3 \rightarrow 1Uncorrect$$

جایگاه های غلط را باهم جمع میکنیم تا جایگاهی که واقعا بیت اشتباهی دارد را تشخیص دهیم، جایگاه ۲ با ۸ که میشود ۱۰:

$$0111001011_{uncorrectbit}10$$

عدد درست ۰۱۱۱۰۰۱۰۱۰۱۰

۴.۲۶.۱ نحوه محاسبه تعداد پرتی های همینگ

یک فرمول ساده در این رابطه وجود دارد $2^r - 1 >= (n)bitnumber + r$ که در آن رقم توان، تعداد پرتی ها و در مقابل تعداد ارقام عدد دریافتی + آن تعداد پرتی

برای مثال عدد دودویی داریم که ۱۱ بیت است برای پیدا کردن این که این رقم چند تا جایگاه پرتی خواهد داشت به صورت زیر خواهد بود:

$$2^3 - 1 >= 11 + 3 false$$

$$2^4 - 1 >= 11 + 4 true$$

پس در ۱۱ بیت ۴ بیت پرتی خواهیم داشت.

۲ جبر بول - ساده سازی - EPI PI،

اساس کار مدار منطقی، جبر بول یا جبر مجموعه ها یا جبر سویچ است. جبر بول به خاطر آقای George Boole نام گذاری شده است، که برای بیان منطق انسان از آن استفاده نمود. بعد از مدتی Shannonn جبر کلیدی یا Switching Algebra را برای نمایش مدارهای کلیدی دو حالتی معرفی کرد یعنی جبر دو ارزشی.

۱.۲ Not Or And truth table

جدول را از راست به چپ بخوانید:

!A	B + A	B . A	B	A
۱	۰	۰	۰	۰
۰	۱	۰	۰	۱
۱	۱	۰	۱	۰
۰	۱	۱	۱	۱

۲.۲ XNOR Xor NOR NAND truth table

A	B	↑ NAND	↓ NOR	⊕ XOR	⊙ XNOR
۰	۰	۱	۱	۰	۱
۱	۰	۱	۰	۱	۰
۰	۱	۱	۰	۱	۰
۱	۱	۰	۰	۰	۱

راهی دیگر برای اثبات **Xor** یا **Xnor** وجود دارد:

$$XOR = (\overline{A}.B) + (\overline{B}.A) \equiv (\sim A \wedge B) \vee (A \wedge \sim B)$$
$$XNOR = (A.B) + (\overline{A}.\overline{B}) \equiv (A \wedge B) \vee (\sim A \wedge \sim B)$$

نکته:

به هر کدام از A.B ها یا حالاتی دیگر در حالت کلی **Products Term** گفته میشود.

به **Product Term** هایی که بین آنها عمل جمع یا or اتفاق افتاده است، **Sum of Product** یا **SoP** گفته می شود.

به هر Products Term که بین آنها عمل ضرب یا And صورت گرفته بر عکس SoP در حقیت PoS یا Products of Sum گفته می شود

- $\overbrace{\underbrace{(\overline{A}.B)}_{ProductsTerm} + (\overline{B}.A)}^{Sum of Products}$
- $\overbrace{\underbrace{(\overline{A}.B)}_{ProductsTerm} . (\overline{B}.A)}^{Products of Sum}$

۳.۲ Dual Low

در قانون دوئال در Product Term ها چند چیز تغییر خواهد کرد:

$$A.B \leftrightarrow A + B$$
$$0 \leftrightarrow 1$$

برای مثال:

$$f(a.b) \rightarrow f^* = a + b$$
$$f = a.1 + \overline{b}.c \rightarrow f^* = (a + 0).(\overline{b} + c)$$
$$f() = A \oplus B = (\overline{A}.B) + (A.\overline{B}) \rightarrow f^*() = (\overline{A} + B).(A + \overline{B})$$

۱.۳.۲ Self Dual

در قانون Self Dual به چیزی اشاره میکند که صحت دوئال آن با حالت عادی آن برابر است

مثال:

ثابت کنید که تابع $f = a.b + a.c + b.c$ با هم خود دگان یا سلف دوئال است:

$$f^*() = (a+b).(a+c).(b+c)$$

$f^*()$	$f()$	c	b	a
۰	۰	۰	۰	۰
۰	۰	۱	۰	۰
۰	۰	۰	۱	۰
۱	۱	۱	۱	۰
۰	۰	۰	۰	۱
۱	۱	۱	۰	۱
۱	۱	۰	۱	۱
۱	۱	۱	۱	۱

۲.۳.۲ اصل Duality

اگر دو تابع با هم سلف دوئال باشند، اگر خاصیتی برای حالت عادی تابع صدق کند برای حالت دوئال آن هم صادق میباشد.

۴.۲ خاصیت های گزاره ها

- خاصیت خودتوانی:

$$A.A = A$$

$$A + A = A$$

- خاصیت جذبی:

$$A.(A + B) = A$$

$$A + (A.B) = A$$

- خاصیت جذبی:

$$A.(A + B) = A$$

$$A + (A.B) = A$$

- خاصیت جابجایی:

$$A.B = B.A$$

$$A + B = B + A$$

- خاصیت شرکت پذیری:

$$A.(B.C) = (A.B).C$$

$$A + (B + C) = (A + B) + C$$

- خاصیت توزیع پذیری:

$$A.(B+C) = (A.B) + (A.C)$$
$$A + (B.C) = (A+B).(A+C)$$

- خاصیت نقیض نقیض:

$$\overline{\overline{A}} = A$$

- خاصیت متمم:

$$\overline{A}.A = 0$$
$$\overline{A} + A = 1$$

- خاصیت همانی:

$$A.1 = A|A+1 = 1$$
$$A.0 = 0|A+0 = A$$

- خاصیت دمورگان:

$$\overline{(A.B)} = (\overline{A} + \overline{B})$$
$$\overline{(A+B)} = (\overline{A}.\overline{B})$$