

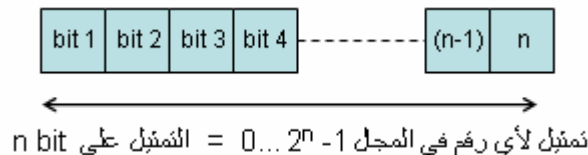
الدرس الثاني: تدوين الأعداد الصحيحة

تدوين الأعداد الصحيحة

نميز هنا بين ثلاثة أنواع من التمثيل : تمثيل الأعداد غير السالبة، وتمثيل المطال والإشارة، والتمثيل بالإتتمام إلى 2

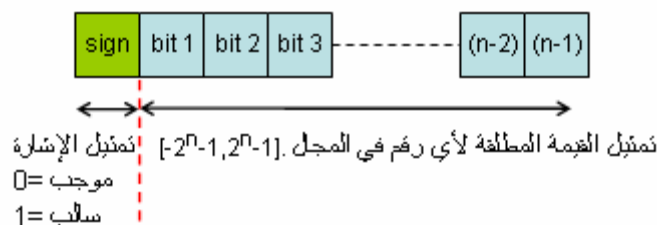
تمثيل الأعداد غير السالبة (Non-negative)

يمكن تمثيل أي عدد موجب (أو غير سالب) على n بت تمثيلاً ثنائياً مباشراً. وباستخدام هذا العدد من البتات، أي n بت، يمكن تمثيل جميع الأعداد الموجبة الواقعة في المجال من 0 إلى $2^n - 1$.



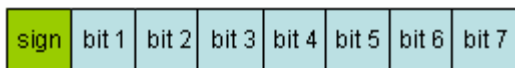
تمثيل الأعداد بالمطال والإشارة Sign/Magnitude

إذا كان العدد يحمل إشارة (أي قد يكون موجباً أو سالباً)، يستخدم لتمثيله حفلان من البتات. يمثل الحقل الأول مطال العدد (أي قيمته المطلقة) ويستخدم لهذا الغرض تمثيل الأعداد غير السالبة المذكور سابقاً. أما الحقل الثاني، وطوله بت واحد، فيفيد في تحديد إشارة العدد (موجب أم سالب). عند استخدام n بت، يمكن تمثيل الأعداد الواقعة في المجال $[-2^{n-1}, 2^{n-1}]$.



تمثيل الأعداد بالمطال والإشارة Sign/Magnitude

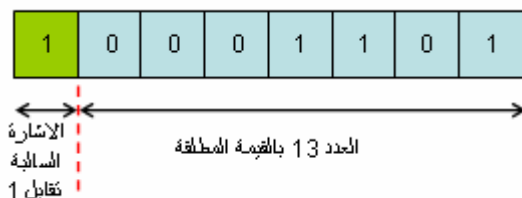
تمرين :



ما تمثيل العدد -13 بطريقة المطال / الإشارة على 8 بتات؟

الحل:

مثل أولاً القيمة المطلقة أي 13 على 7 بتات، فنحصل على : 1101 000. ولما كان العدد سالباً فإن بت الإشارة سيكون مساوياً للواحد. يصبح إذن التمثيل النهائي : 1000 1101.



تمثيل الأعداد بالإتمام إلى 2 (2-Complement)

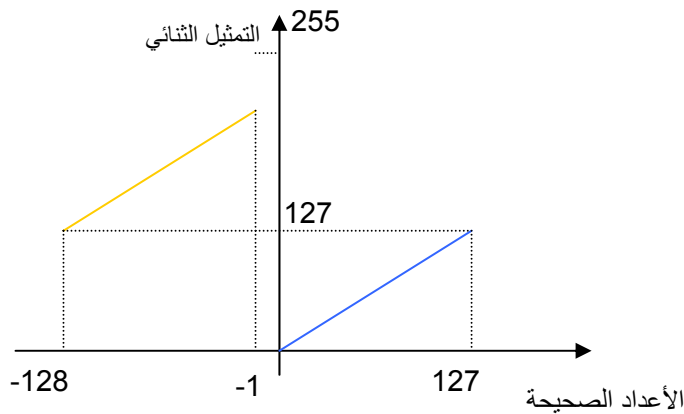
وهو التمثيل الأكثر شيوعاً في النظم الرقمية والحواسيب. يمكن بواسطته تمثيل أي عدد موجب أو سالب على النحو الآتي :

إذا كان العدد موجباً، فإن تمثيله مطابق للأعداد غير السالبة المذكور سابقاً. أما إذا كان العدد سالباً، نأخذ التمثيل الثنائي لقيمته المطلقة، ثم نحسب القيمة المتممة له، ونضيف العدد 1 إلى الناتج. يمثل التمثيل عندئذٍ ترميز العدد السالب بالإتمام إلى 2.

عند تمثيل الأعداد السالبة بالإتمام إلى 2، نأخذ البت الأخير من التمثيل فنجد أن القيمة تساوي 1، على الرغم من عدم توصيفه كبت إشارة.

يبين الشكل التالي العلاقة البيانية بين الأعداد الصحيحة (الموجبة والسالبة) والقيمة الثنائية الناتجة عن الإتمام إلى 2.

يمكن باستخدام n بت، تمثيل الأعداد الواقعة في المجال $[-2^{n-1}, 2^{n-1}-1]$.



تمثيل الأعداد بالإتمام إلى 2 (2-Complement)

تمرين :

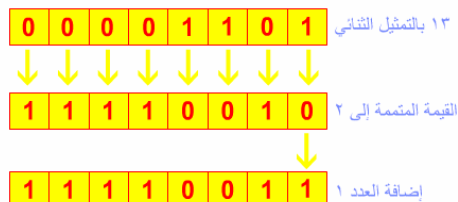
ما تمثيل العدد 13- بطريقة الإتمام إلى 2 ؟

الحل :

إن العدد سالب، لذا نحسب التمثيل الثنائي للقيمة المطلقة (أي 13) فنجد : 0000 1101.

نحسب الآن القيمة المتممة إلى 2، فنجد : 1111 0010.

نضيف العدد 1 إلى الناتج فنحصل على : 1111 0011.



-13 بالتمثيل بالإتمام إلى 2

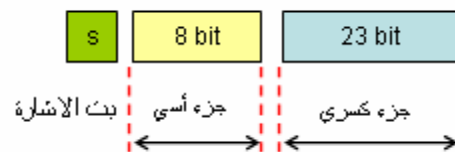
تدوين الأعداد بالفاصلة العائمة

تعريف أساسية

الأس، والأساس، والجزء الكسري : يمكن تمثيل أي عدد ينتمي إلى مجموعة الأعداد الحقيقية Real بكتابتته على النحو الآتي :

$$s \cdot B^{\text{exp}}$$

حيث : s هو الجزء الكسري Significand، و B هو الأساس Basis، و Exp هو الأس Exponent .



الاستنظام Normalization : وتعني الكتابة بدون أصفار إلى يسار الفاصلة. فبدلاً من كتابة القيمة : 0.012، نكتب : 1.2×10^{-3} .

الدقة Precision : وهو عدد البتات التي سيمثل عليها الجزء العشري.

الفجوة Gap : وهو الفرق بين قيمتين متتاليتين. وتجدر الملاحظة إلى أن هذه القيمة ليست ثابتة على كامل مجال الترميز.

المجال Range : وهو المسافة الفاصلة بين بين القيمة الصغرى والقيمة الكبرى الممكن تمثيلهما.

الفيض Overflow : يحدث الفيض إذا كانت القيمة كبيرة جداً بحيث لا يمكن تمثيلها بالتمثيل المعطى (وهذه القيمة قد تكون موجبة أو سالبة).

الفيض Underflow : يظهر الفيض إذا كانت القيمة صغيرة جداً إلى حد لا يسمح بتمثيلها (وهذه القيمة قد تكون موجبة أو سالبة).

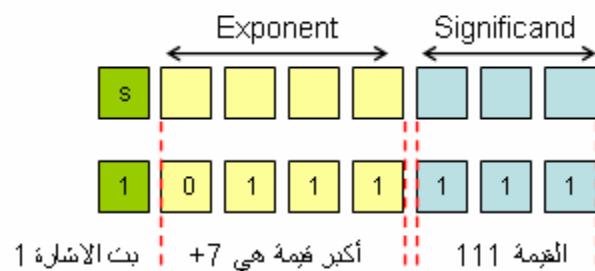
التدوير Rounding : وهو تقريب أي عدد حقيقي إلى عدد يمكن تمثيله بالترميز المعطى. وقد يكون التدوير إلى الأعلى (ceil) أو إلى الأدنى (floor) أو بإهمال الأرقام بعد الفاصلة (Truncation).

تمرين:

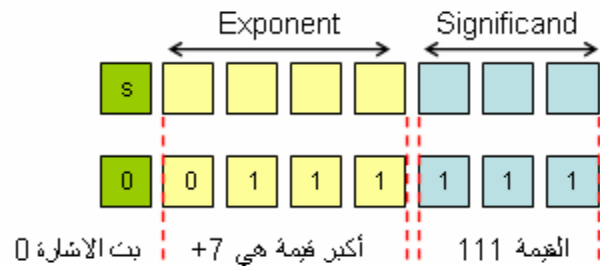
ليكن لدينا التمثيل التالي للأعداد ذات الفاصلة العائمة : 4 بتات للجزء الكسري، و4 بتات للأس. احسب القيمة الصغرى ، والعظمى ، والمجال ، وأصغر قيمة موجبة، وأكبر قيمة سالبة يمكن تمثيلها.

الحل :

القيمة الصغرى: نحصل عليها من جعل الأس مساوياً لأكبر قيمة، أي 0111 (وهي تمثل +7). أما الجزء الكسري فهو 111. وتؤخذ الإشارة السالبة للجزء الكسري (بت الإشارة = 1). وبذا تكون القيمة الصغرى هي -0.111×2^7 (أي -112).



القيمة العظمى: نحصل عليها من جعل الأس مساوياً لأكبر قيمة، أي 0111 (وهي تمثل +7). أما الجزء الكسري فهو 111. وتؤخذ الإشارة الموجبة للجزء الكسري (بت الإشارة = 0). وبذا تكون القيمة العظمى هي $+0.111 \times 2^7$ (أي +112).



يمكن الآن حساب مجال الترميز، وهو 224.

إن أصغر قيمة موجبة هي : 0.001×2^{-8} (أي 0.000488)

إن أكبر قيمة سالبة هي : -0.001×2^{-8} (أي -0.000488).

تمرين:

إذا أردنا تمثيل الرقم : 0.010111 في الترميز السابق، ما هي القيمة بعد التدوير إلى الأدنى، وإلى الأعلى، وإلى الأقرب.

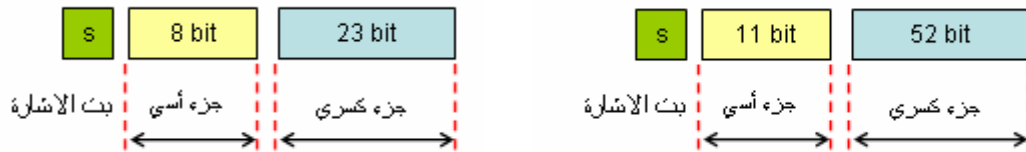
الحل :

التدوير إلى الأقرب : 0.011 - التدوير إلى الأدنى : 0.010 - التدوير إلى الأعلى : 0.011

المعيار¹ IEEE-754

نميز في هذا المعيار بين مستويين للدقة : دقة مفردة (single)، ودقة مضاعفة (double). نستخدم في الدقة المفردة 23 بتاً للجزء الكسري، و8 بتات للجزء الأسّي، وبتاً واحداً للإشارة. أما في الدقة المضاعفة، فيستخدم 52 بت للجزء الكسري، و11 للجزء الأسّي، وبت واحد للإشارة.

ملاحظة مهمة: ويضاف في الحالتين عدد تمثيل الجزء الأسّي انحياز ثابت قيمته في الحالة الأولى 127، وفي الحالة الثانية 1023.



تُمثل القيم الخاصة في هذا المعيار على النحو الآتي :

القيمة 0 : وتأخذ جميع البتات القيمة 0.

قيمة اللانهاية : وتأخذ جميع بتات الجزء الكسري القيمة 0، وبتات الأس القيمة 1، وبت الإشارة القيمة 1 أو 0.

قيمة عدم التعيين (Not a Number) ويرمز لها اختصاراً بـ NaN) : وتأخذ بتات الجزء الكسري قيمة تختلف عن الصفر 0، وبتات الأس القيمة 1، وبت الإشارة القيمة 1 أو 0.

وفي هذا المعيار، يعتمد التدوير إلى الأقرب.

تُعطى المصاغة العامة لهذا التدوين على النحو الآتي : $1.S \times 2^{\text{exp} + \text{bias}}$

¹ راجع الملف IEEE754.pdf

مثال:

ما هو التمثيل الثنائي للعدد : 19.5 في المعيار IEEE754 بالدقة المفردة ؟

الحل :

نكتب العدد وفق المصاغة النظامية للتمثيل IEEE754 :

$$19.5 = 1.21875 \times 2^4$$

العدد موجب، وبالتالي بت الإشارة يساوي : 0

الجزء الكسري يساوي : 0.2875 ، ويكتب إذن

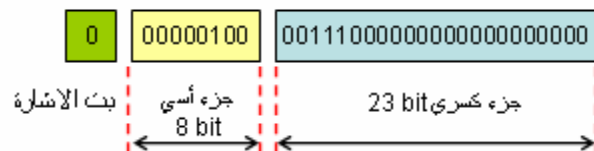
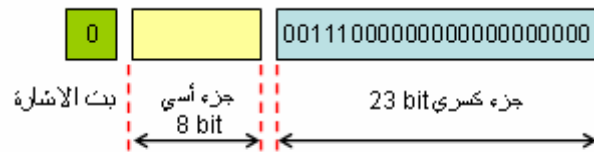
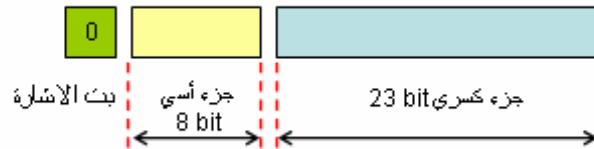
على النحو الآتي : 001110000000000000000000

الجزء الأسّي يساوي 4، يكتب إذن على النحو الآتي :

00000100

نستنتج إذن التمثيل الثنائي للعدد كما يلي :

0 - 00000100 - 001110000000000000000000 -



جمع وطرح الأعداد الصحيحة

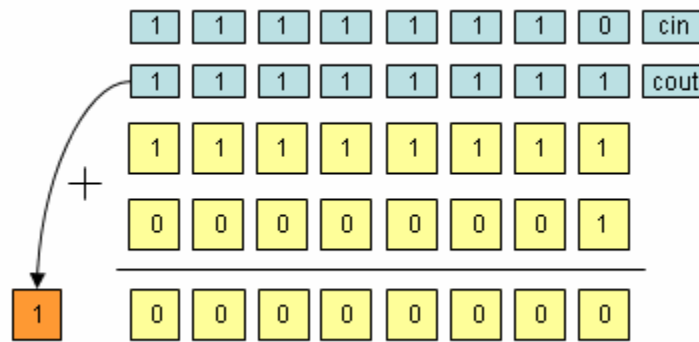
نعرض هنا مبدأ عملية الجمع للأعداد الصحيحة، مع توضيح التنفيذ العملي العتادي لها.

حالة الجمع

المبدأ : يمكن جمع الأعداد الصحيحة المرمزة بطريقة الإتمام إلى 2 جمعاً مباشراً ثنائياً. فإذا كان العددين موجبين، جمعاً جمعاً ثنائياً نظامياً. أما إذا كان العددين سالبين، وجب الانتباه إلى حالة الفيض والتي تحدث عندما يختلف بت الحمل الناتج C_{out} من عملية الجمع الأخيرة (لآخر بتين) عن قيمة بت حمل الدخل C_{in} لهما.

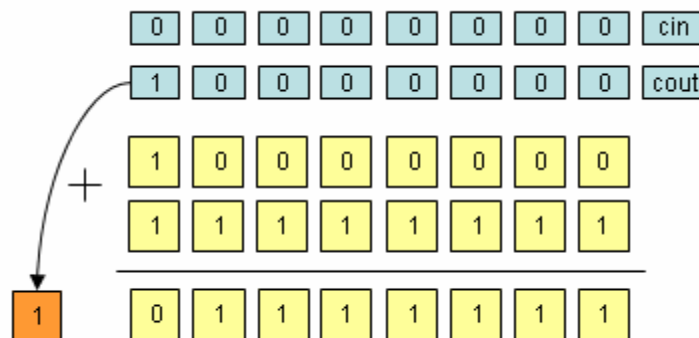
مثال:

نجمع القيمتين التاليتين : **1111 1111** و **0000 0001**. إن الناتج هو : **0000 0000**. وتكون قيمة البت $C_{out} = 1$ (لآخر بتين) و $C_{in} = 1$ (من أجل البتين الأخيرين). وإذا ، لا تحدث حالة فيض، والنتيجة صحيحة. وفي الواقع، تمثل هاتان القيمتان العدد $1+$ ، والعدد $1-$. وناتج الجمع إذن سيكون معدوماً.



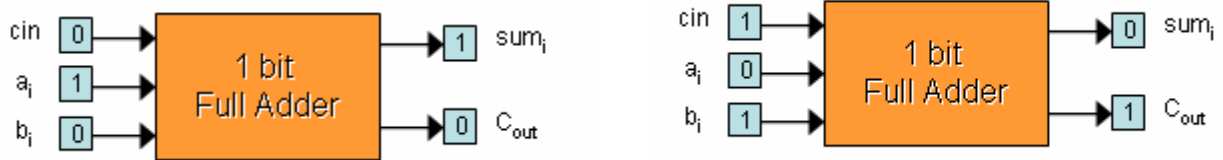
مثال:

نجمع القيمتين التاليتين : **1000 0000** و **1111 1111**. إن الناتج هو : **0111 1111**. وتكون قيمة البت $C_{out} = 1$ (لآخر بتين) و $C_{in} = 0$ (من أجل البتين الأخيرين). وإذا ، تحدث هنا حالة فيض، والنتيجة غير صحيحة. وفي الواقع، تمثل هاتان القيمتان العدد $128-$ ، والعدد $1-$. وناتج الجمع إذن سيكون مساوياً للقيمة $129-$ ، ولا يمكن تمثيلها بكلمة مرمزة على 8 بتات.



التنفيذ العملي

يمكن تنفيذ عملية الجمع للأعداد الصحيحة بواسطة دارة منطقية تركيبية، تسمى دارة الجامع الكامل على n بت (n -bit full adder). تُبنى هذه الدارة من دارة جامع على بت واحد، لها المخطط الصندوقي التالي :



مثال :

اكتب جدول الحقيقة truth table لهذه الدارة، واستنتج العلاقة الثنائية للمخرج بدلالة المدخل.

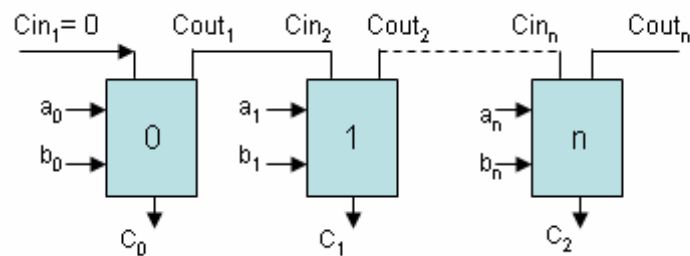
الحل :

بعد كتابة جدول الحقيقة، نجد العلاقتين التاليتين :

$$\text{Sum} = (a.\text{xor}.b) \text{ and } \text{Cin} + \text{Cin} (a.\text{xor}.b)$$

$$\text{Cout} = (a.\text{xor}.b) \text{ Cin} + a.b$$

يمكن استنتاج دارة الجامع على n بت، من دارة جامع على بت واحد، بجمع الدارات الأساسية على التتابع كما يلي :



حالة الطرح

المبدأ : في حالة الأعداد الصحيحة المرمزة بالإتتمام إلى 2 ، يتحقق الطرح بجمع القيمة 1 إلى متمم العدد المطلوب. أي إن عملية الطرح تعود إلى عملية جمع مع المتمم وإضافة القيمة 1 إلى الناتج.

مثال :

احسب حاصل طرح القيمة التالية : 0000 0001

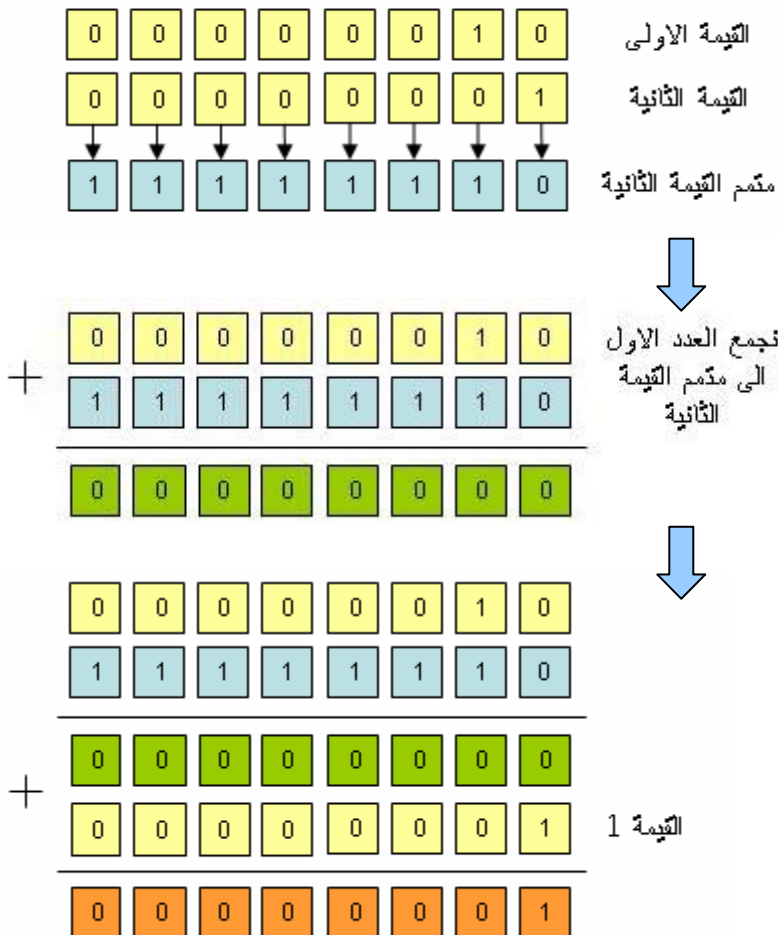
من القيمة 0000 0010.

الحل : إن متمم القيمة الثانية هو : 1110
وبالجمع مع القيمة الأولى نحصل على :

0000 0000. نضيف العدد 1 إلى

المجموع للحصول على الناتج

الأخير، فنجد : 0000 0001.



التنفيذ العملي :

يمكن تحقيق عملية الطرح للأعداد الصحيحة بواسطة دائرة الجامع الكامل، لأن الطرح يعود كما رأينا سابقاً إلى عملية الجمع مع المتمم، والقيمة 1. يبين الشكل التالي طريقة تنفيذ الطرح العتادي.



ملاحظة : يحدث الفيض في حالة طرح الأعداد الصحيحة إذا كان C_{in} يختلف عن C_{out} عند جمع آخر بتين (a_n مع b_n).

ضرب وقسمة الأعداد الصحيحة

حالة الضرب: الطريقة الأولى

يمكن إنجاز الضرب بالجمع المتكرر. تتميز هذه الطريقة بسهولة تنفيذها ولكنها تعاني من بطئها الشديد. يمكن تمثيل مبدأ العمل بالخوارزمية التالية:

```
z=0;
for (i=1;i<=y;i++)
    z=z+x;
```

مثال :

X	Y	Z	
10	2	0	القيم الابتدائية
10	2	10	i=1
10	2	20	i=2

$x=10$, $y=2$, ما هي قيمة z ؟

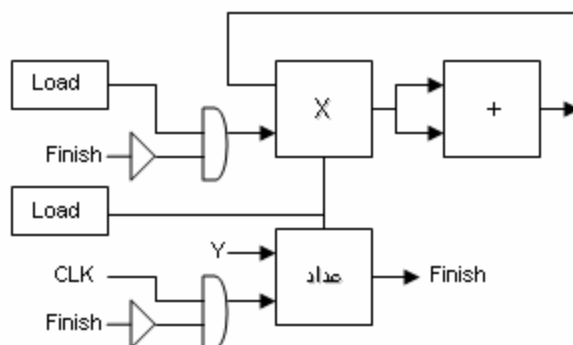
الحل :

بتطبيق الخوارزمية السابقة، يمكن تتبع المراحل التالية :

مثال :

$x=10$, $y=2$, ما هي قيمة z ؟

الحل :



لتنفيذ هذه الطريقة عتادياً، ينبغي الاستعانة بعداد قابل للشحن، ودارة جامع كامل يوصل المدخلان فيها إلى السجل x . تسمح الإشارة Load بتحميل قيمة جديدة في السجل x والعداد. عند انعدام قيمة العداد، تظهر إشارة Finish، التي تعيق إشارة CLK من دخول

العداد. يجري تحميل السجل x بخرج الجامع إلى أن تظهر الإشارة Finish. تمثل القيمة النهائية المخزنة في السجل x حاصل الجداء النهائي.

حالة الضرب: الطريقة الثانية: خوارزمية الضرب والإزاحة (shift and multiply)

لشرح مبدأ هذه الخوارزمية، نبدأ أولاً بعرض المثال التالي.

مثال: لننظر في عملية الجداء التقليدية التي نقوم بها يومياً في النظام العشري :

$$27 \times 253 = ?$$

لإجراء الجداء، نحسب 27×3 ، ثم 27×5 ، ثم 27×2 . نجمع النواتج مع إزاحة القيم خانة واحدة في كل مرة.

تُطبق هذه الخوارزمية على الأعداد ذات التدوين **الموجب**. يمكن وصف مراحل جداء X و Y بالخوارزمية التالية التي تخزن النتيجة في السجلين U و V إضافةً إلى البت C لتخزين الحمل.

```
U=0; V=0;
For (i=1; i<n; i++)
{
    If ( $Y_0=1$ )
        CU=U+X;
    Linear Shift Right CUV;
    Circular Shift Y;
}
```

1. تبدأ الخوارزمية باستبداء السجلين U و V بالقيمة 0.

2. تُفحص بعدئذٍ الخانة الدنيا من Y . إذا كانت معدومة، لا نقوم بأي إجراء، ولكن إذا كانت تساوي الواحد، تُجرى عملية الجمع.

3. بعد ذلك، تُنفذ عملية إزاحة خطية إلى اليمين. ونقصد بالإزاحة الخطية، إزاحة تقليدية لكافة البتات، مع ملء البت الشاغر صفراً.

4. يعذب ذلك **إزاحة دائرية (دوران)** للقيمة Y إلى اليمين لتهيئتها للدورة المقبلة.

الدوران Rotate

إذا كان قيمة x المنطقية المرمزة على 4 بتات، هي : 0111، يؤدي تطبيق عملية الدوران على هذه القيمة إلى نقل كل بت خطوة واحدة إلى اليمين أو اليسار حسب نوع الدوران، وينقل البت المنتمي إلى إحدى النهايتين إلى النهاية المقابلة.

عند الدوران إلى اليمين، نحصل على النتيجة : 1011، وعند الدوران إلى اليسار نحصل على القيمة 1110.

تسمى هذه العملية أحياناً **الإزاحة الدائرية Circular Shift**

الإزاحة المنطقية Logical Shift

إذا كان قيمة x المنطقية المرمزة على 4 بتات، هي : 0111، يؤدي تطبيق عملية الإزاحة المنطقية على هذه القيمة إلى نقل كل بت خطوة واحدة إلى اليمين أو اليسار حسب نوع الإزاحة، ويهمل البت المنتمي إلى إحدى النهايتين . يملأ البت الشاغر صفراً دوماً.

عند الإزاحة المنطقية إلى اليمين، نحصل على النتيجة : 0011، وعند الإزاحة المنطقية إلى اليسار نحصل على القيمة 1110.

الإزاحة الحسابية Arithmetic Shift

تختلف هذه الإزاحة عن سابقتها بأن البت الشاغر لا يملأ صفراً دوماً، بل يملأ البت الكائن في موقع بت الإشارة (أي البت ذو الأعلى) بالقيمة 0 إذا كان العدد موجباً، و 1 إذا كان سالباً.

مثال تطبيقي :

ليكن $x=13$ ، $y=11$.

العملية	i	CUV	Y
U=0; V=0		0 0000 0000	1011
If (y0=1)	1	0 1101 0000	
Lshr (CUV)		0 0110 1000	
Cshr (Y)			1101
If (y0=1)	2	1 0011 1000	
Lshr (CUV)		0 1001 1100	
Cshr (Y)			1110
If (y0=1)	3		
Lshr (CUV)		0 0100 1110	
Cshr (Y)			0111
If (y0=1)	4	1 0001 1110	
Lshr (CUV)		0 1000 1111	
Cshr (Y)			1011

ملاحظة: في حالة الأعداد السالبة، يمكن أخذ الإشارة في الحسبان على النحو الآتي :

```
If (x<0)
    x=-x;
if (y<0)
    y=-y
//multiply
//restore old values
If (x<0) or (y<0)
    result=-result;
```

حالة الضرب: الطريقة الثانية: خوارزمية Booth :

تتسم هذه الخوارزمية بسرعتها وبإمكانية تطبيقها على الأعداد **الموجبة والسالبة** . ويمكن تمثيلها بما يلي :

```
U=0; Y-1=0
For (i=1;i<n; i++)
    If (y0y-1=10)
        U=U-X
    If (y0y-1=01)
        U=U+X
1 Ashr (UV)
2 Cshr (Y)
Y-1=Y0
```

1. يقصد بالبت Y_{-1} القيمة السابقة الناتجة عن الدوران إلى اليمين للسجل Y . وفي البداية، يكون محتوى البت معدوماً.

2. ونلاحظ هنا أيضاً استخدام **الإزاحة الحسابية**³ بدلاً من الإزاحة الخطية (أو المنطقية) ويعني ذلك، أن الإزاحة لا تشمل بت الإشارة. فبعد الإزاحة الحسابية لعدد موجب، نحصل على عدد موجب، وبعد الإزاحة الحسابية لعدد سالب نحصل على عدد سالب أيضاً. لا تكتشف الخوارزمية إلا الحالة 10 أو الحالة 01. أما الحالة 00 أو 11 فتبقى بدون تأثير.

¹ Ashr = Arithmetic Shift Right

² Cshr = Circular Shift Right

³ الإزاحة الحسابية Arithmetic Shift

تختلف الإزاحة الحسابية عن سابقتها (الإزاحة المنطقية) بأن البت الشاغر لا يملأ صفرًا دومًا، بل يملأ البت الكائن في موقع بت الإشارة (أي البت ذو الأعلى) بالقيمة 0 إذا كان العدد موجبًا، و 1 إذا كان سالبًا.

مثال تطبيقي :

ليكن $X=7$ $Y=6$

العملية	i	UV	Y
$U=0, y_{-1}=0$		0 0000 0000	0110
$If(y_0 y_{-1}=10)$	1		
Ashr (CUV)		0 0000 0000	
Cshr (Y)			0011
$If(y_0 y_{-1}=10)$	2	1 1001 0000	
Ashr (CUV)		1 1100 1000	
Cshr (Y)			1001
$If(y_0 y_{-1}=10)$	3		
Ashr (CUV)		1 1110 0100	
Cshr (Y)			1100
$If(y_0 y_{-1}=10)$	4	0 0101 0100	
Ashr (CUV)		0 0010 1010	
Cshr (Y)			0110

حالة الفسمة: الطريقة الأولى: الطرح المتكرر

وهي الطريقة الأسهل تنفيذاً ولكنها الأبطأ أداءً. يمكن وصف عملها بالخوارزمية التالية :

```

z=0;
While (x>=y) do
{
  z=z+1;
  x=x-y
}
```

مثال: $x=5$, $y=2$, ما هي قيمة z ؟

الحل: بتطبيق الخوارزمية السابقة، يمكن تتبع المراحل التالية :

5	2	0
3	2	1
1	2	2

حالة القسمة: الطريقة الثانية : الإزاحة والطرح Shift and Subtract

لحساب حاصل القسمة التالي :

$$y = \frac{U.V}{x}$$

تُستخدم الخوارزمية التالية (وهي مشابهة لحالة الضرب، مع تغيير جهة الإزاحة)

```
If (U>=X)
    Exit
Y=0; C=0;
For (i=1; i<n; i++)
{
1Lshl (CUV);
2Cshl Y;
If (CU>=X)
    Y0=1
    CU=CU-X
}
```

ملاحظة: لا تُستخدم هذه الطريقة إلا للأعداد الموجبة.

¹ Lshl = Linear Shift Left

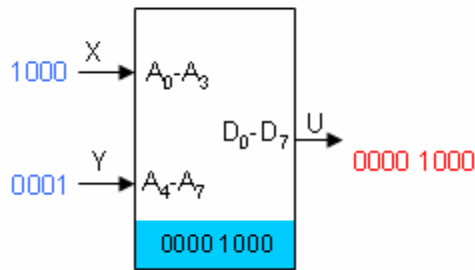
² Cshl = Circular Shift Left

ضرب وقسمة الأعداد الصحيحة

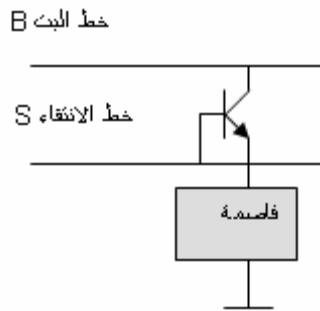
حالة القسمة: الطريقة الثانية : الإزاحة والطرح Shift and Subtract

التنفيذ العملي

يمكن تنفيذ خوارزمية القسمة في الزمن الحقيقي باستخدام جداول التقابل (Look-Up Tables (LUT). تُنفذ هذه الجداول بواسطة ذاكرات من النوع **ROM** (للقراءة فقط)، إذ تُخزن فيها كافة النتائج الممكنة لجداء القيمتين X و Y . تُدخل القيمتان إلى الذاكرة عبر خطوط العنوان، ونحصل على نتيجة الجداء من خطوط المعطيات كما هو مبين في الشكل التالي.



الذاكرة ROM



وهي ذاكرة القراءة فقط، تماثل في واجهتها الخارجية الذاكرة السكونية. يخزن برنامج المستخدم أو معطياته عند صنعها ولا يمكن تغيير محتوياتها بعد البرمجة. تتألف الخلية الأساسية فيها من فاصمة، توصل أو تُفصل عند البرمجة. يبين الشكل التالي بنية هذه الذاكرة.

جمع وطرح الأعداد العشرية المرمزة ثنائياً

مذكرة :

في هذا الترميز، يمثل أي عدد عشري يقع بين 0 و 9، على 4 بتات.

مثال :

العدد	التمثيل	الاحاد	العشرات	المئات	الآلاف
123	عشري	3	2	1	0
	BCD	0011	0010	0001	0000
5349	عشري	9	4	3	5
	BCD	1001	0100	0011	0101

الجمع

عند جمع قيمتين BCD جمعاً ثنائياً تقليدياً باستخدام دارات الجامع الكامل، نحصل على قيمة ثنائية لا تمثل الناتج الحقيقي بالترميز BCD. ولتصحيح الناتج لكي يصبح ممثلاً بالترميز BCD ينبغي إضافة القيمة 6 إلى ناتج الجمع إذا كان المجموع أكبر من 9.

مثال : ما هو ناتج الجمع التالي :

$$(17)_{10} + (04)_{10} = ?$$

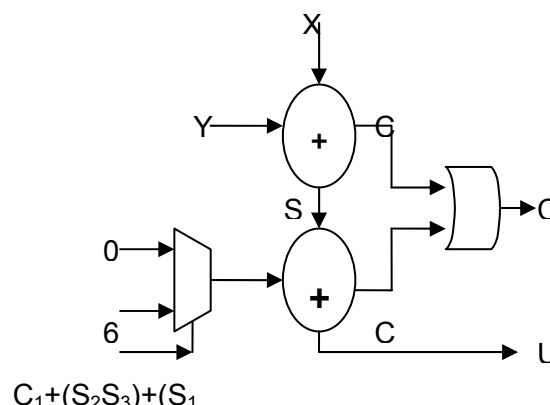
عند الجمع الثنائي النظامي. إن النتيجة لا تمثل ناتج الجمع العشري (وهو 21) ممثلاً بالترميز BCD.

لتصحيح النتيجة، ينبغي إضافة القيمة 6 إلى ناتج جمع أول خانتين (لان المجموع أكبر من 9). فنحصل على ما يلي :

$$(0010\ 0001)_{BCD} = (0000\ 0110)_{BCD} + (0001\ 1011)_{BCD}$$

التنفيذ العملي

يبين الشكل التالي التنفيذ العتادي لإدارة الجمع بالترميز BCD.



لا تُضاف القيمة 6 إلا إذا كان ناتج جمع الخانتين أكبر من 9 ، وهذا ما يعبر عنه المعادلة المنطقية التي تتحكم بإشارة انتقاء الناخب.

الطرح

عند طرح قيمتين BCD طرحاً ثنائياً تقليدياً باستخدام دارات الجامع الكامل، نحصل على قيمة ثنائية لا تمثل الناتج الحقيقي بالترميز BCD. ولتصحيح الناتج لكي يصبح ممثلاً بالترميز BCD ينبغي إعادة عملية الطرح إلى جمع، بالإتمام ثم إضافة القيمة 1. وتُصحح النتيجة بالطريقة ذاتها المذكورة في حالة الجمع أي إضافة القيمة 6 إلى ناتج الجمع إذا كان المجموع أكبر من 9.

جمع وطرح الأعداد بالفاصلة العائمة

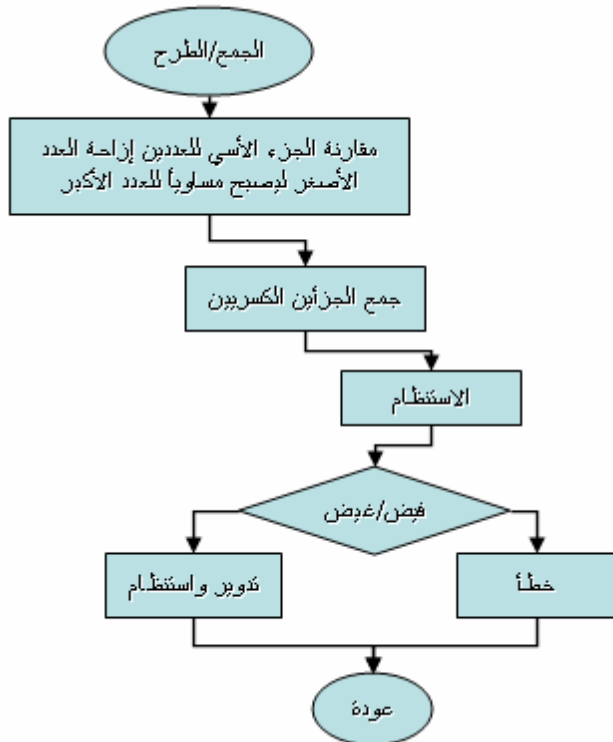
خوارزمية الجمع والطرح

1. لإجراء الجمع أو الطرح، نقوم أولاً بمقارنة الجزء الأسّي من كل عدد. فإذا كانا مختلفين، نزيح العدد الأصغر ليصبح مساوياً للعدد الأكبر، ويرافق ذلك تغيير الجزء الكسري بما يتناسب مع مقدار الإزاحة. بعد تساوي الجزأين الأسيين

2. نجمع الجزأين الكسريين للعددين كأعداد صحيحة مرمزة بطريقة الإتمام.

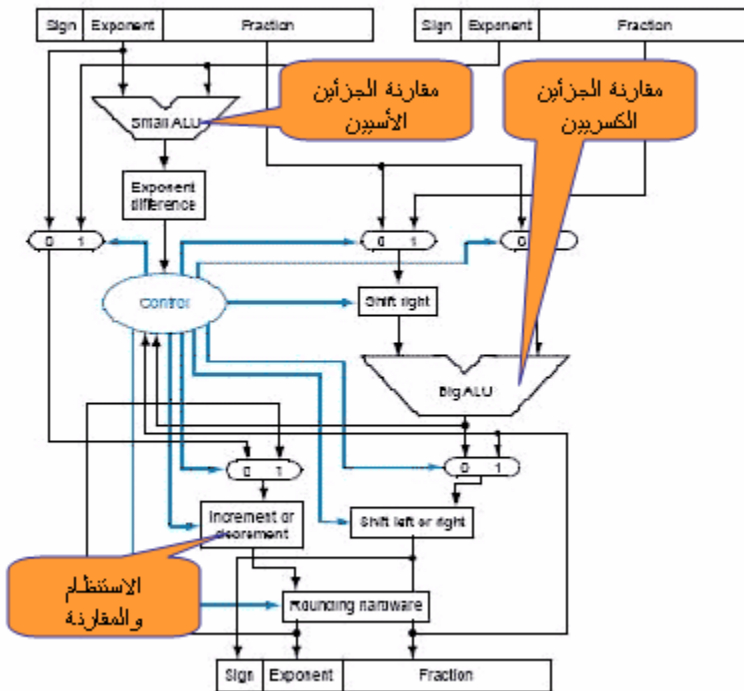
3. يلي ذلك مرحلة استنظام للنتيجة normalization لكتابة النتيجة النهائية بما يتناسب مع المعيار المتبع.

4. نفحص بعدئذٍ حالة راية الفيض والغيض. فإذا أُعلن عن حدوث أي منهما، نعلن عن خطأ في الجمع (مثلاً، نكون قد قمنا بجمع عددين كبيرين بحيث لا يمكن ترميز النتيجة النهائية ترميزاً صحيحاً). وإلا، ندور النتيجة ونطبق الاستنظام ثانياً على القيمة الناتجة، ونهني الخوارزمية.



خوارزمية الجمع: التنفيذ العملي

نبين هنا طريقة لتنفيذ الخوارزمية السابقة عتادياً. تتطلب هذه الطريقة استخدام وحدة حساب ومنطق ¹ALU لإجراء عملية المقارنة والجمع، وسجل إزاحة إلى اليمين وإلى اليسار لاستنظام النتيجة.



¹ وحدة الحساب والمنطق ALU

وهي تزود وحدة المعالجة المركزية بقوتها الحسابية. فهي قادرة على تنفيذ العمليات الحسابية الأساسية (مثل الجمع والطرح والضرب والقسمة)، والعمليات المنطقية الأساسية (مثل عمليات الجمع والضرب والنفي المنطقي). وقد تحوي هذه الوحدة كتلة خاصة بالعمليات الحسابية للأعداد ذات الفاصلة العائمة floating-point.

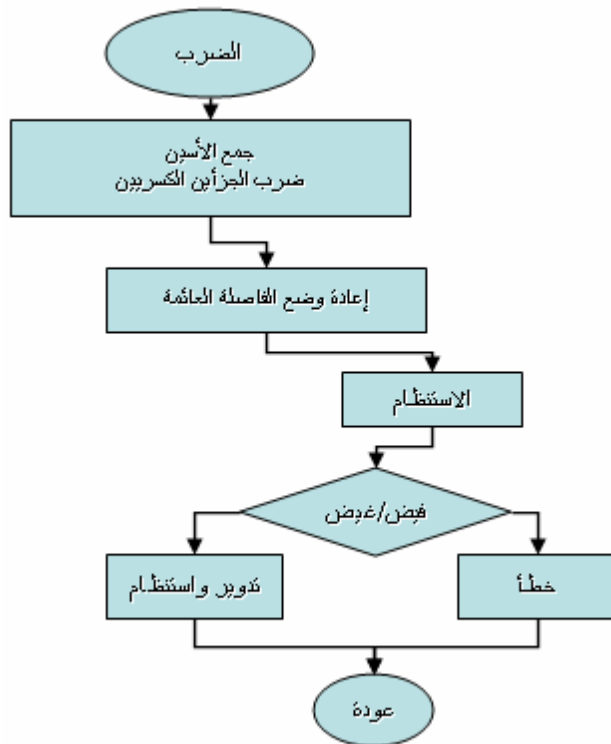
نبين هنا العمليات اللازمة لجمع عددين وفق الخوارزمية المقترحة.

$$9.999 \times 10^1 + 1.61 \times 10^{-1} = ?$$

- 1- 0.0161×10^1 Same Exponent
- 2- $9.999 \times 10^1 + 0.161 \times 10^1 = 10.015 \times 10^1$ Addition
- 3- 1.0015×10^2 Normalization
- 4- 1.001×10^2 Rounding

ضرب الأعداد بالفاصلة العائمة

خوارزمية الضرب



عند ضرب عددين بالفاصلة العائمة، يجمع أولاً الجزأين الأسيان كجمع الأعداد الصحيحة، ويحسب جداء الجزأين الكسريين كعددين صحيحين مرمزين بطريقة الإشارة والمطال. وتُستَظَم النتيجة النهائية لوضعها وفق المعيار المعتمد للأعداد بالفاصلة العائمة.

لتوضيح مبدأ ضرب الأعداد ذات الفاصلة العائمة، نأخذ المثال التالي

$$9.200 \times 10^{-5} \times 1.110 \times 10^{10} = ?$$

- 1- $-5 + 10 = 5$ Add Exponents
- 2- $9200 \times 1110 = 10212000$ Multiplication
- 3- 10.212×10^5 Floating-point location
- 4- 1.0212×10^6 Normalization
- 5- 1.021×10^6 Rounding