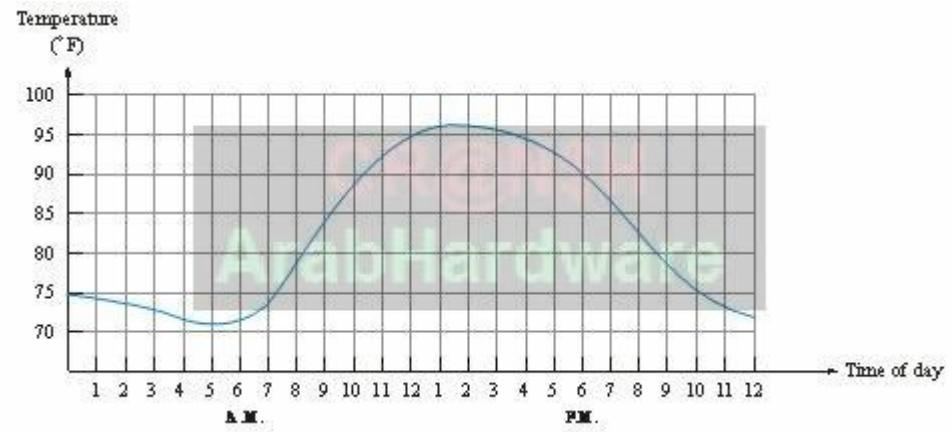
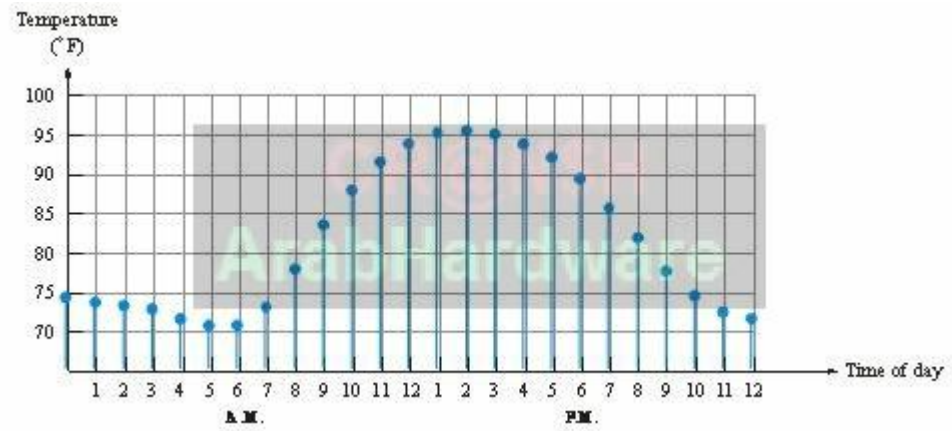
كيف تعمل وحدة المعالجة المركزية (CPU) – الجزء الثاني

هذا الموضوع هو الجزء الثاني من سلسلة كيف يعمل الحاسب لذلك يفضل مراجعة الجزء الأول قبل البدء به،كل المناهج العالمية التي تدرس بنية الحاسب و كيفية عمل المكونات تتطلب بمعرفة جيدة لمنهاج يدعى النظم المنطقية

وهو ما سأقوم بشرحه حاليا ولو بشكل مبسط،الإشارة المستمرة و الإشارة المتقطعة ،في العالم من حولنا يوجد العديد من المتغيرات و تمتاز هذه المتغيرات باستمراريتها، أما معنى الاستمرارية فهو بساطة استمرارها مع الزمن، فالحرارة على سبيل المثال ممكن أن تتغير أو تبقى ثابتة مع الزمن و لكنها مستمرة معه كما يوضح الشكل التالي و الذي يعبر عن تغير درجات الحرارة خلال ال٢٤ ساعة:



هذه الإشارة تسمى بإشارة تشابهية مستمرة،الأن حتى نستطيع أن نعالج متغير كالحرارة بشكل رقمي يجب أن نقوم بتقطيع الإشارة أي نأخذ منها عينات بفترات زمنية متساوية فتصبح الإشارة المتقطعة بالشكل التالي:

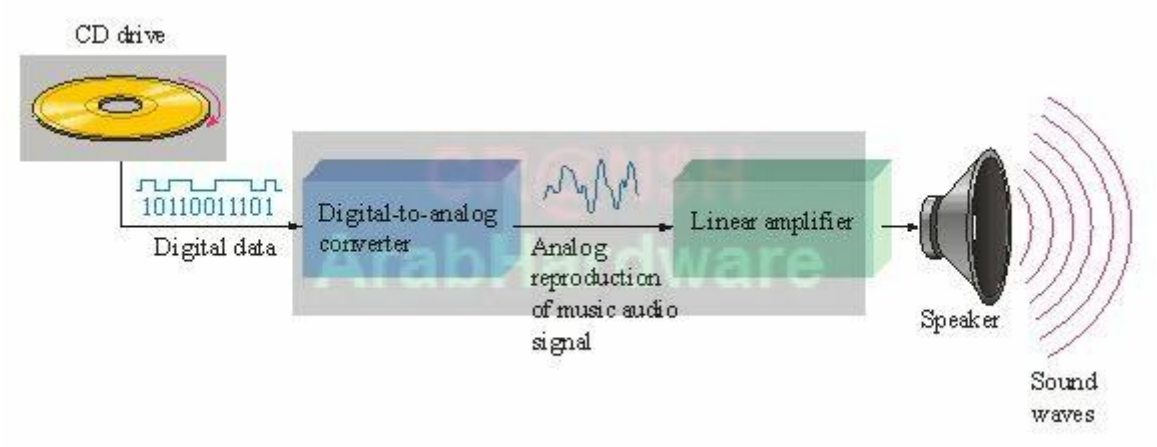


هذه الإشارة تسمى بإشارة تشابهية متقطعه،هناك العديد من الشروط لتكون هذه العملية صحيحة ولكن لن نتطرق لها فهذا ليس محور حديثنا،طبعا هناك مصطلحان قد مرا و لكن لم أقم بشرحهما وهم "نعالج" و "بشكل رقمي" :

المعالجة بلغة الحاسب هي الحصول على معلومات مفيدة من بيانات محددة. كمثال درجات الحرارة خلال اليوم بحد ذاتها قد لا تكون مفيدة ولكن قد نحتاج إلى متوسط الحرارة خلال ساعات النهار أو خلال ساعات الليل. عملية الحصول على هذه القيم من درجات الحرارة تسمى "معالجة".

أما كلمة بشكل رقمي فيجب التفريق بين مصطلحي التشابهي و الرقمي. الإشارة التشابهية هي أي متحول فيزيائي موجود من حولنا "الحرارة، الرطوبة، الطول، التيار الكهربائي، السرعة ..." حيث يكون له قيمة معينة خلال لحظة زمنية معينة
الإشارة الرقمية هي تحويل لإشارة تشابهية ما بحيث يمكن إدخالها إلى معالج رقمي لتتم معالجتها بعد أن يتم تقطيعها و تحويل قيمها إلى الشكل الرقمي

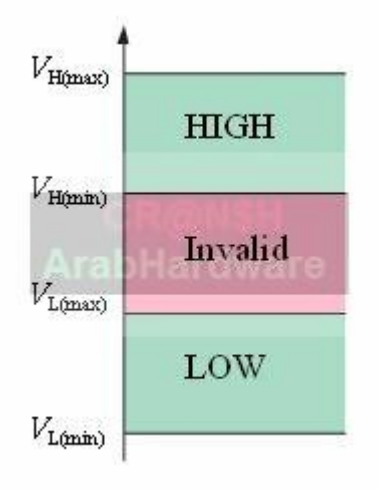
مثال بسيط يوضح فكرة الإشارة الرقمية و التشابهية



هذا الصورة تمثل و بشكل بسيط كيفية تحويل الإشارة الرقمية المخزنة في القرص المضغوط إلى إشارة تشابهية عن طريق محول تشابهي رقمي converter Digital to analog ثم تتم تهيئة الإشارة التشابهية عن طريق مضخم Amplifier و ثم تحول إلى إشارة صوتية عن طريق المجهر Speaker.

توصيف الأعداد الثنائية فيزيائياً

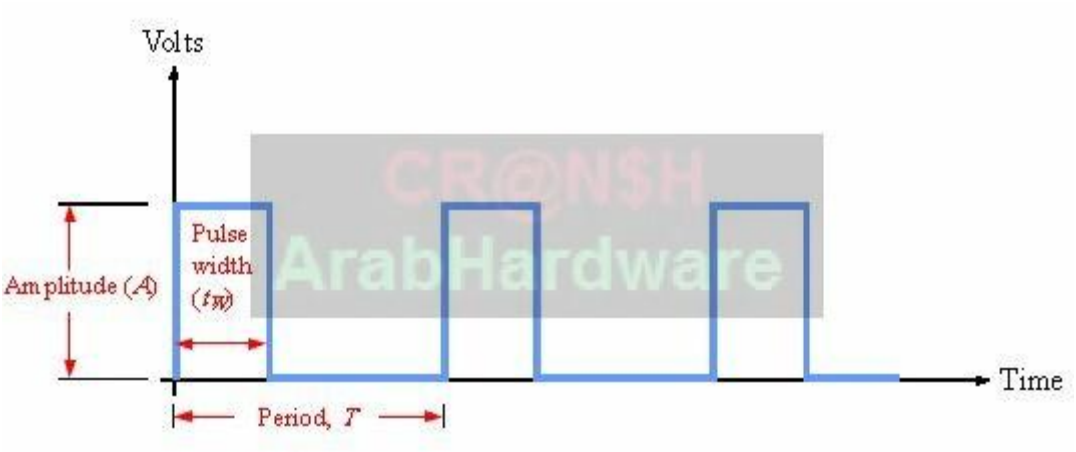
لقد عرفنا أننا نقوم باستعمال الأعداد الثنائية في الحاسب و عرفنا أيضا أن هذه الأعداد الثنائية تعبر عن حالة مرور التيار الكهربائي في دارة معينة وهذا الكلام بشكل نظري أما عمليا يتم التعبير عن ٠ و ١ من خلال مجال معين "وليس قيمة واحدة" و غالبا ما يتوسط هذا المجال منطقة غير معرفة .



في الحاسب الشخصي غالبا ما يمثل ١ بجهد كهربائي بين (٥ و ٣.٣)فولت و ال٠ بجهد كهربائي بين (٢ و ٠) فولت والمنطقة بين (٣.٣ و ٢)فولت تعتبرمنطقة غير معرفة "لا يمكن معرفة القيمة التي سيفهمها الحاسب هل هي ١ أم ٠".

نبضات الساعة Clock pluses:

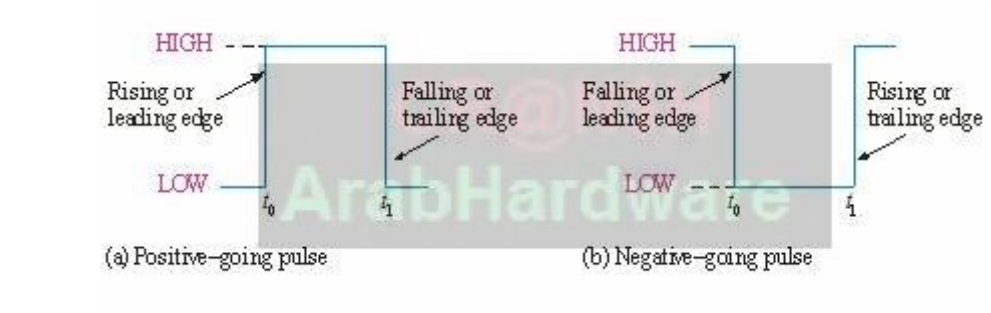
في جميع الأنظمة الإلكترونية نحتاج إلى من يضبط إيقاع عملها وهذه هي مهمة الClock "هل تفكر بشيء يدعى OverClock!.



مهمة هذه النبضات أن تؤمّن التزامن بين أجزاء الحاسب كافة و غالبا ما يكون هناك تردد أساسي في الحاسب لنبضات الساعة و يتم الحصول على ترددات عمل أجزاء الحاسب الأخرى من خلال مضاعفته أو تجزيئه فعلى سبيل المثال الحاسب الذي أملكه فتردد مولد نبضات الساعة لدي بشكل إفتراضي هو ٢٠٠Mhz
يتم الحصول على تردد المعالج منه عن طريق مضاعفته ١٤ مرة أي ١٤ * ٢٠٠MHz = 2800MHz
يتم الحصول على تردد الذواكر منه عن طريق مضاعفته مرتان فقط أي ٢ * ٤٠٠MT/s = 800MHz
طبعا من المعروف أن تردد ذواكر ال DDR الحقيقي يكون نصف ما يذكر عليها "وهو التردد الفعلي"

جبهات نبضات الساعة :

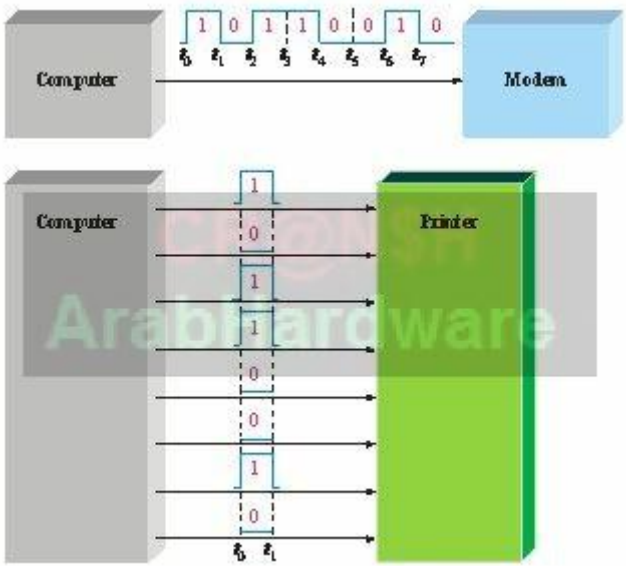
تتألف كل نبضة ساعة من جبهتين الأولى صاعدة من ٠ إلى ١ و الثانية هابطة من ١ إلى ٠ أي أن كل نبضة تحوي على جبهتين.



في ذواكر الDDR يتم تنفيذ عملية (قراءة أو كتابة) عند كل جبهة و ليس عند النبضة مما يعطي تردد عمل فعلي لها ضعف الحقيقي.

طرق نقل القيمة الرقمية

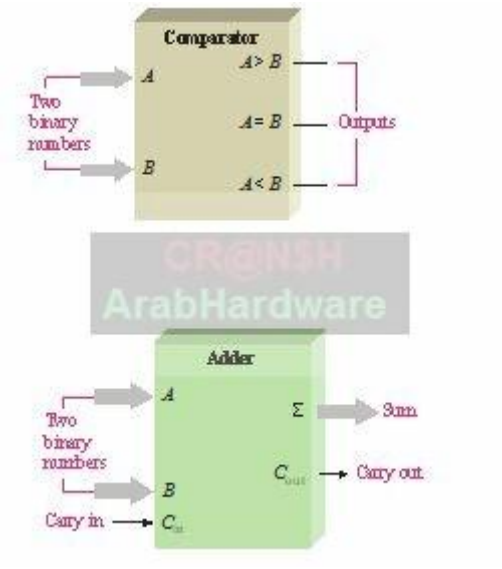
،عرفنا أن الحاسب يقيس حجم البيانات بوحدة البايت و أن كل بايت يحتوي على ٨ بت، هناك طريقتين لنقل البايت الواحد من أي جزء للحاسب إلى جزء أخر



الأولى بشكل تسلسلي "الشكل الأول" وهو أن يتم نقل البايت الواحد على شكل قطار من النبضات (١ أو ٠ حسب البايت المنقول) من طرف إلى أخر من خلال سلك نقل وحيد "على الأقل" و كمثال عليه منفذ الCOM في الحاسب أو منفذ الUSB ، و تتميز هذه الطريقة بقلة عدد خطوط نقل الإشارة و ممانعتها الكبيرة للتشويش الخارجي لذلك تستخدم بكثرة في إتصال الحاسب مع العالم الخارجي و يعيبها أنها أبطأ من الطريقة الثانية،الثانية بشكل تفرعي حيث يتم استخدام عدد كبير من خطوط نقل البيانات و تساوي أقل عدد بتات يجب أن تنقل دفعة واحدة فمثلا ٨ للبايت و ١٦ للكلمة الرقمية وهكذا و يتم نقل البايت "مثلا" دفعة واحدة و تتميز بسرعتها و لكنها تكون معرضة بشكل كبير للتشويش الداخلي "بين خطوط النقل بعضها ببعض" و التشويش الخارجي لذلك غالبا ما تستخدم لنقل البيانات بين أجزاء الحاسب الداخلية "بين المعالج و الذاكرة مثلا"

الدارات الرقمية الأساسية

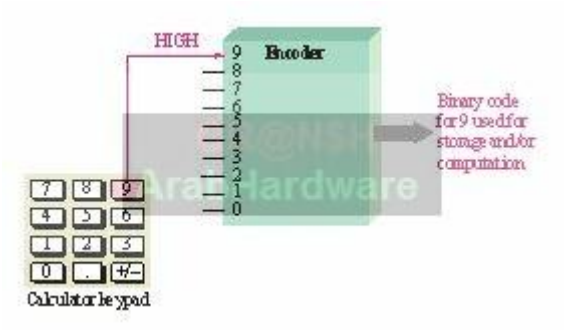
،سأقوم بذكر أهم الدارات الرقمية المتواجدة حاليا و التي تعد حجر الأساس للمعالج و سأتحدث عن وظيفتها فقط حاليا أما لاحقا سأقوم بشرح تكوينها الداخلي في مواضيع لاحقة إن إقتضى الأمر.



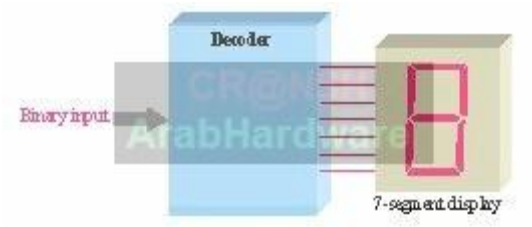
المقارن Comparator،يتضح من اسم هذه الدارة أنا تستقبل عددان ثنائيان بنفس الطول B&A "متساويان بعدد البتات" و تضع قيمة ١ منطقي على أحد مخارجها
التي هي A<A B

الجامع Adder،وهي دارة تقوم بجمع عددين ثنائيين بنفس الطول B&A و تضع قيمة الجمع على خرجها و تقبل أن يكون هناك حمل من عملية سابقة و من الممكن تعطي حمل لعملية لاحقة،أما عملية الطرح فهي تحول إلى عملية جمع وتنفذ على هذه الدارة أيضا كما سيتضح معنا لاحقا.

المشفر Encoder و فاك الشيفرة Decoder،مهمة المشفر و فاك السفرة بعكس بعضهما البعض "طبعا واشح من الاسم" ،مهمة المشفر هي تحويل قيمة معينة من دخله إلى قيمة ثنائية تماثل هذه القيمة.



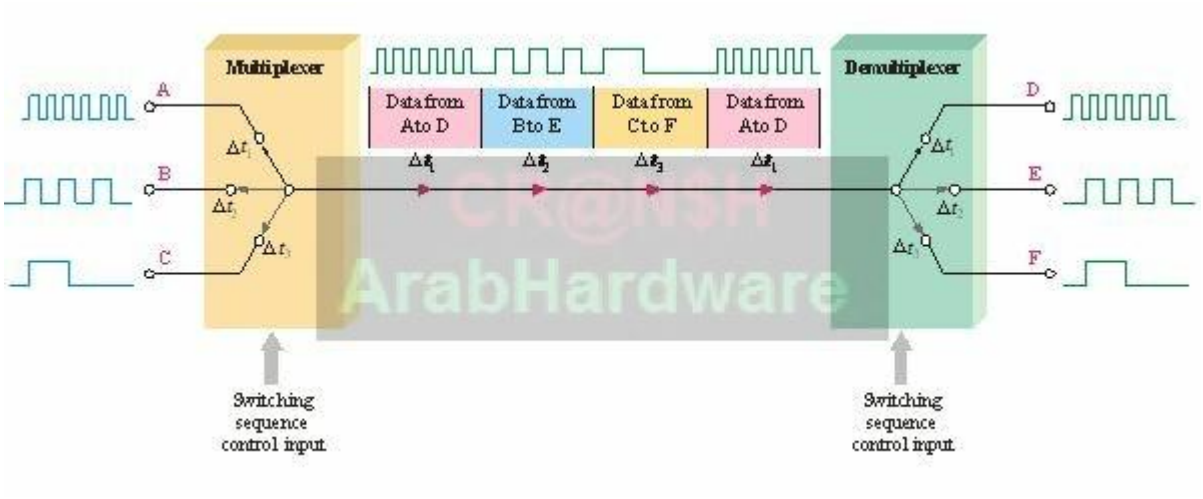
كما في الصورة يستخدم المشفر لإعطاء العدد الثنائي المقابل للمفتاح الذي تم ضغطه من لوحة المفاتيح. مهمة فاك الشيفرة هي تحويل عدد ثنائي إلى شكل أخر "قيمة عشرية مثلا أو حرف " حسب تسمية مخارجه و نوعها



توضح هذه الصورة فاك تشفير يسمى Binary to 7-Segment Decoder و مهمته تحويل رقم ثنائي بين ٠ و ٩ إلى صيغة معينة تسمح بإظهار الرقم على الشاشة سباعية القطع "غالبا ما نراها في لوحات المصاعد".

<u>الناخب Multiplexer و الموزع De-multiplexer</u>

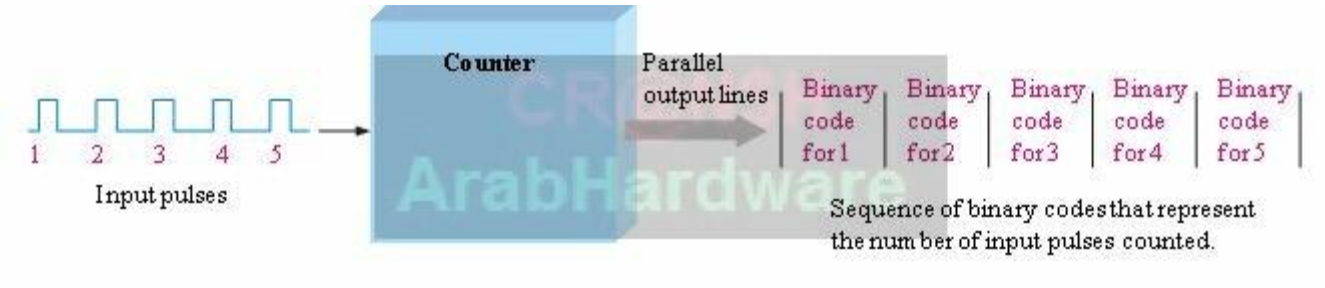
مهمة الناخب هي يصل أحد مداخله المتعددة إلى خرجه الوحيد اعتمادا على القيمة الموجودة على مداخل التحكم.
مهمة الموزع هي مدخله الوحيد إلى أحد مخارجه المتعددة اعتمادا على القيمة الموجودة على مداخل التحكم.



يوضح هذا المثال كيفية استخدام الناخب و الموزع لنقل عدة بيانات على خط و حيد و هنا تبرز أهمية الساعة التي تتحكم بعمل كل منهما "ولو بشكل غير مباشر" فلو كان كل جزء يعمل بسرعة مختلفة عن الأخر لأصبح من المستحيل الفصل بين الإشارات المستقبلة،و تستخدم هذه الدارات بشكل كبير في أجزاء الربط بين المعالج و الذاكرة على سبيل المثال

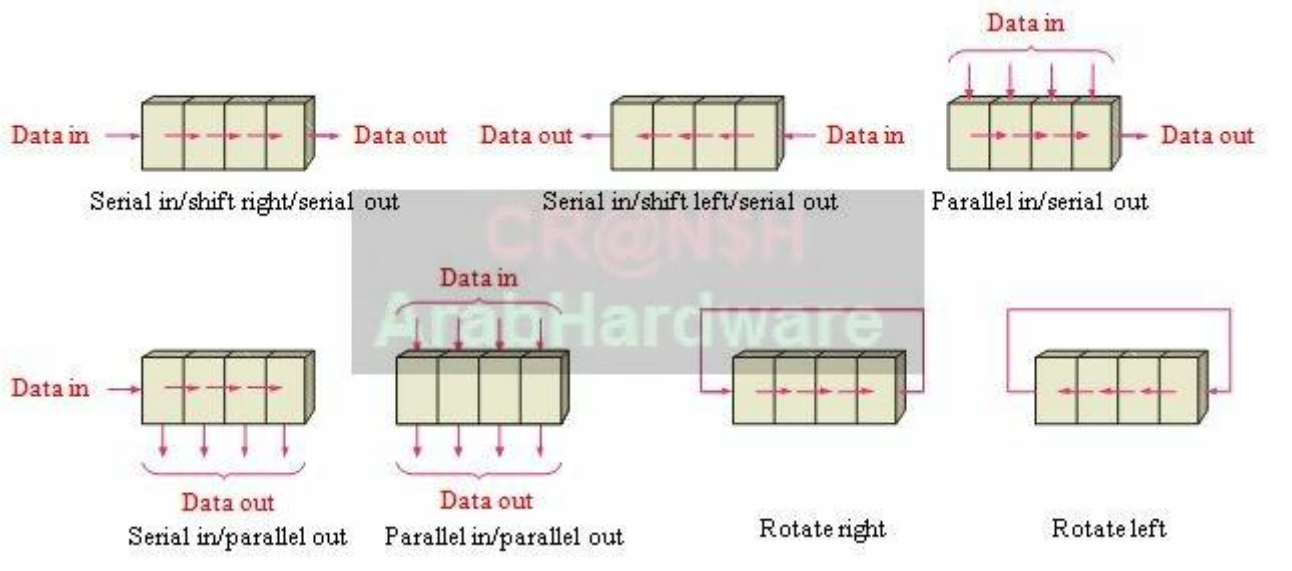
العدادات Counters

تستخدم هذه الدارات لتشكيل قيم على خرجها "غالبا متتالية" مقابل كل نبضة ساعة تصلها



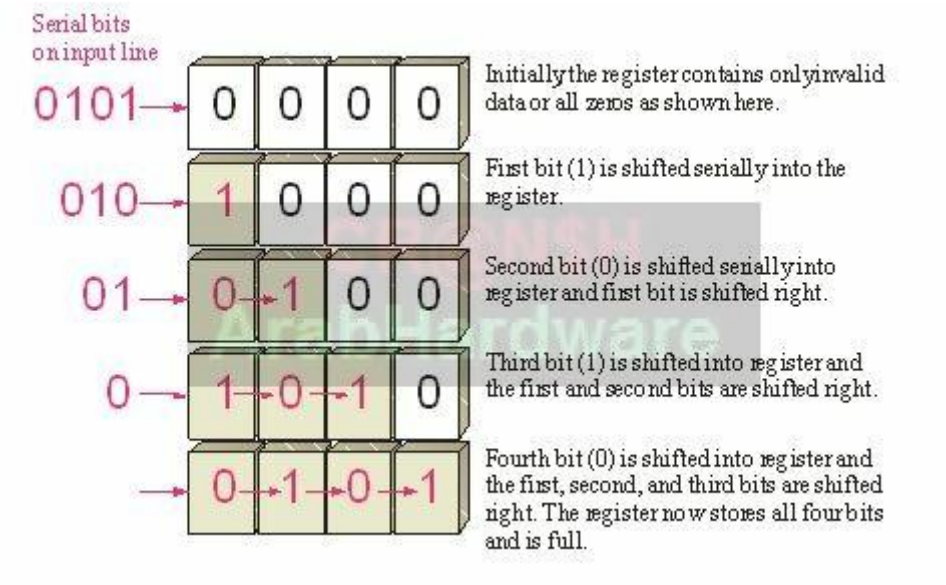
المسجلات Register

المسجلات تقسم إلى العديد من الأنواع حسب طريقة دخول البيانات إليها و خروجها و هي عبارة عن حجرات ذاكرية تحتفظ بقيمتها ما دامت مغذاه "غالبا" وهي



١-مسجلات تسلسلية تسلسلية مع إزاحة إلى اليمين أو اليسار
٢-مسجلات تفرعية تسلسلية
٣-مسجلات تسلسلية تفرعية
٤-مسجلات تفرعية تفرعية
٥-مسجلات الإلتفات إلى اليمين أو اليسار

حيث يعبر نوع نقل البيانات الأول على طريقة إدخال القيمة للمسجل و تعبر نوع نقل البيانات الثاني عن طريقة إخراج الملفات من المسجل،و سنتعرف على استخدامات العديد منها لاحقا يكفي أن تعرف أن مسجلات الإلتفات تفيد كثيرا في عمليات الضرب و القسمة



هذه الصورة تمثل مسجل من النوع التسلسلي وكيفية إدخال المعلومات فيه

<u>نظرة أعمق إلى نظام العد الثنائي</u>

Decimal Number	Binary Number
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

سأقوم بإضافة بعض الأمثلة والمعلومات الإضافية عن النظام الثنائي

التحويل من الثنائي إلى العشري :

كما تحدثنا من قبل أن كل خانة من الأعداد الثنائية لها وزن يعبر عنه بالأساس ٢ مرفوع إلى موقع الخانة،عند التحويل نقوم بضرب قيمة الخانة الثنائية (١ أو ٠) بوزن الخانة كما يوضح المثال التالي:

Example Convert the binary number 100101.01 to decimal.

Solution Start by writing the column weights; then add the weights that correspond to each 1 in the number.

2^5	2^4	2^3	2^2	2^1	2^0.	2^{-1}	2^{-2}	
32	16	8	4	2	1 .	½	¼	
1	0	0	1	0	1.	0	1	
32			+4		+1		+¼ =	37¼

نقوم بكتابة أوزان الخانات "السطر الأحمر"،نقوم بكتابة قيمة الأوزان بشكل رقم صحيح عوضا عن أس و أساس "ليست شرطا"،نتكب قيمة الخانة الثنائية حسب موقعها ثم نقوم بعملية الضرب،نجمع نواتج الضرب و نحصل بذلك على الرقم العشري المقابل للرقم الثنائي

التحويل من العشري إلى الثنائي:

هناك طريقتين واحدة طويلة و لكنها عامة "تصلح لأي نظام عد كان" و لن أشرحها هنا أما الثانية فهي البسيطة و القصيرة وهي المفضلة غالبا،تعتمد الطريقة على حفظ الأوزان الثنائية (١ ٢ ٤ ٨ ١٦ ٣٢ ٦٤ ١٢٨ ٢٥٦ ٥١٢ ١٠٢٤ ٢٠٤٨) حيث أن كل خانة هل ضعف سابقتها.

Example Convert the decimal number 49 to binary.

Solution The column weights double in each position to the right. Write down column weights until the last number is larger than the one you want to convert.

2^6	2^5	2^4	2^3	2^2	2^1	2^0.
64	32	16	8	4	2	1.
0	1	1	0	0	0	1.

الأن لنأخذ مثلا الرقم ٤٩:

نكتب الخانات الثنائية بحيث أن تكون أخر خانة أكبر من العدد المطلوب تحويله ١ ٢ ٤ ٨ ١٦ ٣٢ ٦٤

نبحث عن أكبر خانة ثنائية بحيث تكون أصغر أو تساوي ٤٩ فنجد أنها ٣٢ ،نقوم بعملية الطرح ٤٩-٣٢ فيكون الناتج ١٧ نبحث عن أكبر خانة ثنائية بحيث تكون أصغر أو تساوي ١٧ فنجد أنها ١٦،نقوم بعملية الطرح ١٧-١٦ فيكون الناتج١ نبحث عن أكبر خانة ثنائية بحيث تكون أصغر أو تساوي ١ فنجد أنها ١،نقوم بعملية الطرح ١-١ فيكون الناتج ٠ و عندها نتوقف،الأن نضع الرقم ١ أسفل كل خانة ثنائية تم استخدامها أي أسفل ال٣٢ و ١٦ و ١ و أسفل بقية الخانات ٠

١ ٢ ٤ ٨ ١٦ ٣٢ ٦٤، ١ ٠ ٠ ٠ ١ ١ ٠

و بهذا يكون تحويل ٤٩ إلى ثنائي هو ١١٠٠٠١

العمليات الحسابية على الأعداد الثنائية

الجمع

نجمع كل خانة إلى الخانة المقابلة وفق الجدول التالي

0 + 0 = 0	Sum = 0, carry = 0
0 + 1 = 0	Sum = 1, carry = 0
1 + 0 = 0	Sum = 1, carry = 0
1 + 1 = 10	Sum = 0, carry = 1

حيث Sum هي ناتج الجمع و Carry هي الحمل للقيمة التالية "كما يحصل في عملية الجمع العشرية عندما يكون الناتج أكبر من ٩"،مثال:

```
 0111
 00111     7
 10101    21
 -----    --
 11100  = 28
```

الطرح،تتم عملية الطرح كما الجدول التالي:

0 – 0 = 0	
1 – 1 = 0	
1 – 0 = 1	
10 – 1 = 1	with a borrow of 1

و من الممكن أن نحتاج إلى إستعارة b١٠ من الخانة الأكبر كما يوضح المثال التالي:

```
  111
 10101    21
 00111     7
 -----    --
 01110  = 14
```

الضرب،القاعدة الأساسية في الضرب هي :

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

و عند ضرب عدد من الخانات نقوم بالعملية كما لو كنا نتعامل مع أعداد عشرية،مثال:

Perform the following binary multiplications:

(a) 11×11 (b) 101×111

Solution (a)

```
          11      3
        × 11    × 3
Partial   11      9
products +11
         1001
```

(b)

```
          111      7
        × 101    × 5
Partial   111     35
products  000
         +111
        100011
```

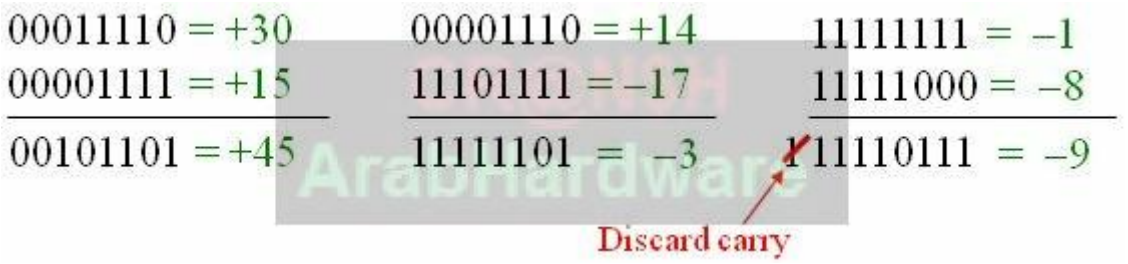
المتمم الأحادي و المتمم الثنائي،فكرة المتمم موجودة في جميع نظم العد وهي المتمم التي إذا جمعناها مع عدد الذي تم حسابه منه تعطنا قيمة معينة و هي إما عدد مؤلف من أخر رمز في نظام العد "المتمم الأحادي" أو المتمم الأحادي +١ و هو ما يسمى المتمم الثنائي.
ففي الأعداد العشرية يعطى المتمم الأحادي للعد ٥٩٦ بالشكل التالي: ٩٩٩-٥٩٦=٤٠٣ أما المتمم الثنائي يكون ٤٠٤
أما في الأعداد الثنائية فالعملية أسهل "رغم أنها تتبع إلى نفس القاعدة" ،فالعدد ١١٠١٠١٠ متممه الأحادي هو ٠٠١٠١٠١ و متممه الثنائي هو ٠٠١٠١١٠
إذ أننا نبدل كل ١ ب ٠ و كل ٠ ب ١ و ذلك للحصول هعلى المتمم الأحادي و عند جمع الرقم ١ نحصل على المتمم الثنائي من المتمم الأحادي،فائدة المتتمات تظهر عند عملية الطرح ففي الحاسب يتم التعامل مع جميع الأرقام على أنها متممات ثنائية.
على فرض أن لدينا بايت . سيستطيع هذا البايت أن يتسع الأرقام من ٠٠٠٠٠٠٠٠ حتى ١١١١١١١١ و قد تم الإتفاق على أن جميع الأعداد من من ٠٠٠٠٠٠٠٠ إلى ٠١١١١١١١ هي أعداد موجبة و أن جميع الأعداد من ١٠٠٠٠٠٠٠ إلى ١١١١١١١١ على أنها أعداد سالبة هذا يجعل البايت الواحد يتسع إلى أعداد من -١٢٨ إلى ١٢٧ ،و سبب ذلك أنه تم الإتفاق على جعل أخر بت في كل بايت يحتوي على قيمة سالبة "وهو خانة ١٢٨"وهذا المثال يوضح كيفية تحويل من متمم ثنائي إلى عدد عشري مؤشر

Example Assuming that the sign bit = −128, show that 11000110 = −58 as a 2's complement signed number:

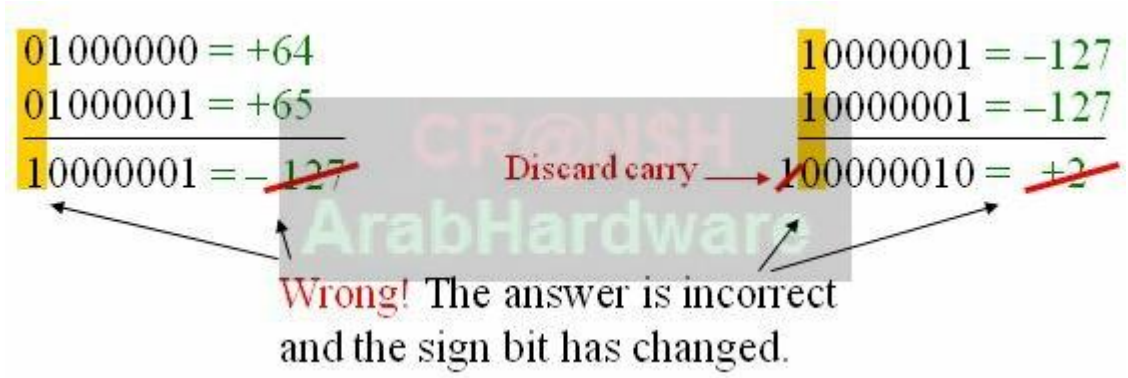
Solution Column weights: −128 64 32 16 8 4 2 1.

```
                1  1  0  0  0  1  1  0
−128 +64                   +4 +2      = −58
```

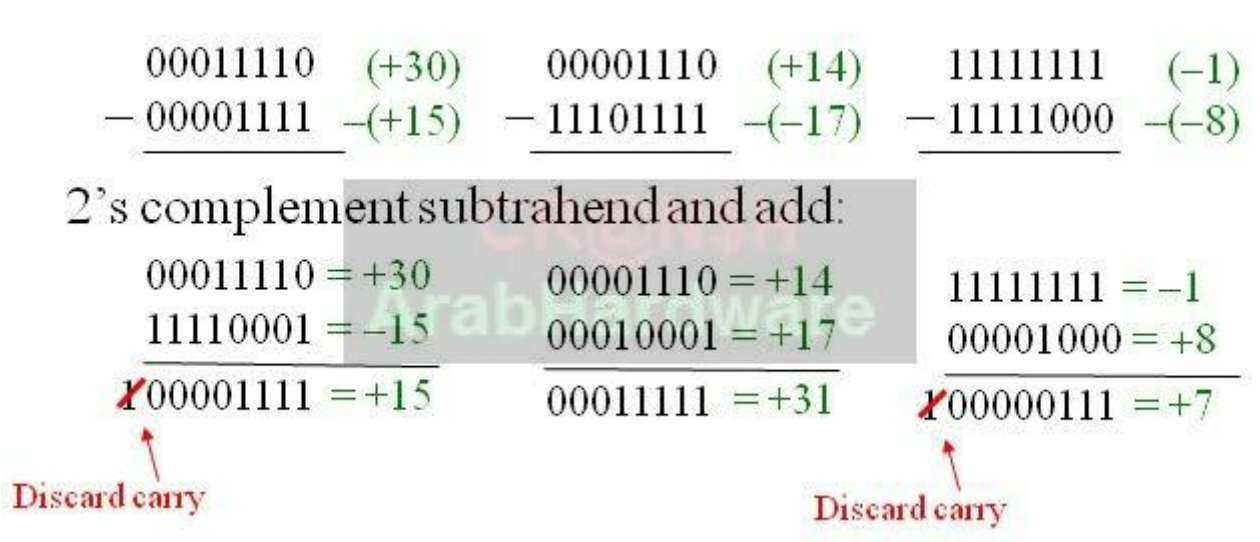
<u>العمليات على المتمم الثنائي</u> "ما يحدث فعلا داخل المعالج"،على إعتبار أن كل ما يتعرف عليه الحاسب من أرقام تتم الحسابات عليها على أنها متمم ثنائي فيجب أن نعرف أنه يتعامل مع الجمع و الطرح على أنهما عملية واحدة و الأمثلة التالية توضح ذلك



(الحالة ١) نلاحظ في حالة جمع عددين موجبين لا يوجد أي مشكلة "جمع تقليدي"
(الحالة ٢) نلاحظ في الحالة الثانية حولت عملية الطرح إلى جمع بعد أن تمت كتابة الرقمين على أنهما متمم ثنائي و كان الناتج أيضا متمم ثنائي
(الحالة ٣) نلاحظ في الحالة الثالثة كيف أصبحت عملية الجمع بين عددين سالبين عملية جمع تقليدية و كان الناتج بشكل متمم ثنائي و لكن هنا يجدر ذكر أنه في حال كان هناك طفحان يتم إهماله و الطفحان هو تخطي الناتج عدد الخانات المخصصة للعملية،في بعض الأحيان ممكن أن يحصل ما يسمى بتغير خانة الإشارة و هذا يحدث عندما نجمع عددين موجبين و يكون الناتج سالب أو جمع عديين سالبين و يكون الناتج موجب كما في الصورة



<u>أمثلة إضافية توضح كيف تحول عملية الجمع و الطرح إلى جمع فقط عند التعامل مع المتمم الثنائي</u>



الأن أي شخص درس أي لغة برمجة سيعترض و بقوة و سيقول لي "متحولات الInteger و التي تحوي على قيم صحيحة بطول Byte١ يمكن أن تتسع إلى أرقام من ٠إلى ٢٥٥ و على حسب ما ذكرت سأحتاج إلى Byte٢ كي يتسع إلى ٢٥٥" و سيكون الجواب هنا يأتي دور لغة البرمجة إذ أن المعالج بالنسبة له كل ما يدور حوله يعتبر أرقام فقط و لن يفرق بين عدد موجب أو عدد سالب و لكنه زود ببعض الادوات التي تساعده على التعامل مع أي شيء فالمعالج يحتوي على ما يسمى مسجل الحالة إذ أن هذا المسجل يحتوي يعلم البرنامج المنفذ عن

حدوث حالة معينة في المعالج قد حدثت حتى يستطيع البرنامج فهم سبب الجواب الغريب و يخرجه للمستخدم بشكل صحيح ،عادة ما يكون طول هذا المسجل بايت و لكنه الأن ليس حديثنا ما يهمنا أن نعرفه أنه يحوي على ما يسمى Negative flag و Flag Overflow و الكل بت من هذا المسجل يسمى Flag
سأخذ الأن كما يسمى سيناريو و اسقطه على المعالج و نرى كيف يقوم البرنامج بفهم ما يحدث من خلال مسجل الحالة،لنفترض أني قمت بالعملية التالية٠١٠٠٠٠٠٠ (+٦٤)،+ ٠١٠٠٠٠٠١ (+٦٥)

١٠٠٠٠٠٠١ (-١٢٧)

الأن ما سيحدث هو في حال كان المتحول أثناء كتابة البرنامج معرف على أنه Integer فلن ينظر إلى الNegative flag و الذي يحدد إن كان الناتج سالب "تصبح قيمته ١ عندما يكون ناتج العملية سالب" و يتم التعامل مع الناتج على أنه عدد ثنائي عادي و ليس متمم ثنائي و هذه العمليات هي من إختصاص لغة البرمجة

<u>نظام العد الست عشري</u> ،هذا النظام يتكون من ١٦ عشر رمز "من ٠ إلى F" و تتوزع وفق الجدول التالي

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

CRONSH
ArabHardware

و هو مهم جدا عند التعامل مع الذاكرة و ذلك لأنه يستطيع وصف عدد كبير من خلال رموز قليلة فمثلا،العدد FFFF يمثل في الأعداد العشرية ٦٥٥٣٥ وفي الأعداد الثنائية ١١١١١١١١١١١١١١١١ و يتميز هذا النظام أن أساسه يقبل القسمة على اساس النظام الثنائي لذلك يمكن أن توصف كل خانة من نظام العد الست عشري ب بأربع خانات من الأعداد الثنائية لأن ٢ مرفوعة للأساس ٤ تعطي الرقم ١٦

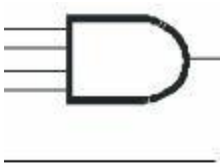
<u>نظام العد الثماني</u>،و يتكون من ثمانية رموز "٠ إلى ٧"

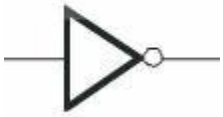
Decimal	Octal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111

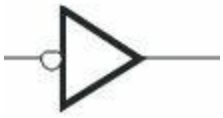
ARAB FRONSH
ArabHardware

وهذا النظام يملك نفس ميزة النظام الست عشري و لكن هنا كل ٣ خانات ثنائية تمثل خانة ثمانية

البوابات المنطقية الأساسية،البوابات المنطقية هي عبارة عن دارات إلكترونية تعد حجر الاساس لأي نظام الكتروني كان فالدارات الرقمية السابقة ماهي إلا مجموعة من البوابات متصلة مع بعضها البعض،قد سميت بوابات لأنها تتحكم بمرور التيار الكهربائي فحسب دخلها تعطي قيمة للخرج و هي







AND
OR
NOT

و تتميز كل بوابة بما يسمى جدول حقيقة وهو جدول يوضح جميع القيم الممكنة للدخل و ما يقابلها في الخرج

بوابة AND&،وهي بوابة تعطي على خرجها ١ عندما تكون قيم مداخلها جميعا ١ وهذا جدول الحقيقة لبوابة AND بمدخلين

Inputs		Output
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

بوابة OR |،وهي بوابة تعطي على خرجها ١ عندما تكون إحدى أو جميع مداخلها ١ وهذا جدول الحقيقة لبوابة OR بمدخلين

Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

بوابة NOT ~و تسمى العاكس أيضا و تعطي قيمة معاكسة للقيمة الداخلة إليها

Input	Output
A	X
LOW (0)	HIGH (1)
HIGH (1)	LOW(0)

من خلال هذه البوابات الأساسية نستطيع الحصول على بعض البوابات المساعدة الأخرى و هي NAND ،هذه البوابة عبارة عن بوابة AND يوجد على خرجها بوابة NOT

ويعطى جدول الحقيقة الخاص بها بالشكل

Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

NOR،هذه البوابة عبارة عن بوابة OR يوجد على خرجها بوابة NOT

ويعطى جدول الحقيقة الخاص بها بالشكل

Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

XOR

وهي بوابة تسمى ببوابة عدم التماثل و تعطي ١ على خرجها عندما تختلف مداخلها كما في الجدول

Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

سأكتفي بهذا القدر حاليا،الموضوع التالي سأشرح كيفه كيفية عمل الترانزستور و كيفية تشكيل البوابات الرئيسية من الترانستورات و شرح لبعض الأفكار الرئيسية في تقنيات تصنيع المعالج وبعض العلاقات الرياضية الجميلة

المصدر : عرب هاردوير

http://arabhardware.net/articles/hardware/cpu-and-motherboard/480-how-cpu-works-part-2.html

تم التحميل من شبكة المنهل التعليمية

http://111000.net