بسم الله الرحمن الرحيم

العقل المدبر إلي بيشتغل بـ 0 و 1 "البروسيسور"

قبل ما نبدا هاد الملف ليس خالي من الأخطاء لأن كل الأشخاص معرضين للخطأ.

شو يعني digital system ولشو مفيد تعلم الـ digital systems وليش أغلب العرب ما عندهم قوة بعلم الأنظمة الرقمية.

- شو يعني **ديجيتال:** الديجيتال هو تمثيل المعلومات عن طريق الأرقام الثنائية (0 و 1). كل 0 او 1 اسمه (bitبت). وبنجمعهم مع بعض بصيروا لغة يفهمها الكومبيوتر والأنظمة الحديثة.

- ليش لازم نتعلم **digital systems** ؟
لأن اللب الأساسي أو النواة لفهم كل الأنظمة الرقمية الحديثة، متل ما المكعبات (legos) هي الأساس لبناء أي شكل هندسي.
يعني أغلب الوظائف المطلوبة اليوم وبكثرة أساسها حرفياً هو الـ digital systemsمتل: .(AI , IoT and cybersec)

نحنا العرب يا صديقي ما منحب نتعب يعني بكون تطلعنا فوراً عالنتائج من غير ما نشوف المجهود إلى بذلناه وهاد الأمر غلط لأن حتى رب العالمين سبحانه وتعالى لا يحاسب على النتائج بل يحاسب على السعي والجهد والمحاولة. "لا يكلّف الله نفساً إلّا وُسعها".

فـ العرب بيضعفوا بالمجال هاد بسبب قفزهم فوراً على "النتائج" (متل البرمجة) من غير ما يفهموا الأساس.

بإذن الله رح نشرح كيفية تصميم المعالج نظرياً ...

بسم الله

Number Systems

- الأرقام العشرية (decimal numbers): لأرقام العشرية هي ضمن النطاق التالي [0, 1, 2 ... 9]

هذه للخانة الواحدة لكن من الممكن أن يتم جمعها مع بعض وتصبح أكبر متل:

"63638212"

كل عامود من هي الأرقام له وزنه نحن نقرأ من اليمين إلى اليسار لحساب الأعمدة هي :

10

100

1000

etc .. 10000

وهون الاساس أو الـbase هو الرقم "10" لأن الأرقام من 0 -> 9

فالأساس العشري هو 10 يعني رح نعرف الأرقام هيك

$(63638212)_{10}$

يمثل الرقم العشري المكون من الرقم n (N-digit decimal number)

يمثل أحد الاحتمالات (0, 1, 2 ... 10^n-1) وهاد اسمو نطاق الرقم

مثال: الرقم الممثل من 3 أرقام عشرية متل رقم في احتمال 1000 النطاق الخاص فيه [0، .... ،999].

- الأرقام الثنائية (binary number): هو الرقم المكون من 0 و 1 وبالاخير بيتضمنوا مع بعضن مشان يكونوا رقم متل:
01001101

أساس الرقم الثنائي هو "2" (N-bit binary)

ومشان نحسبوا بطريقة سهلة هو مضاعفات رقم الـ2

1, 2, 4, 8, 16, 32, 64, 128, 256 ...

طيب، كيف منحول الرقم الثنائي لـ عشري ؟

$(1010)_2$ -> $(?)_{10}$

N-bit = N-bit* wight^ column

منبلش من اليمين: $(0*2)^0 + (1*2)^1 + (0*2)^2 + (1*2)^3 = (10)_{10}$

- الأرقام الست عشرية(hexadecimal numbers) اختصارا "hex": الرقم السداسي العشري مكون من أربع بتات --> 2^4 = 16

أرقام الستة عشري تتمثل [0 -> 9 & A -> F]

والأساس الخاص فيه هو 16

DECIMAL	HEX	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

طيب ليش منتعلم صيغ الأرقام ؟ بدون فهم الصيغ بكون صعب فهم كيفية عمل الـ hardware أو البرمجيات منخفضة المستوى الـ low level وهيك ما رح تقدر تطور نظام رقمي أو تقدر تحللو.

Converting:

*** من ستة عشري إلى ثنائي:**

$(4AF)^{16}$ -> $(?)^{2}$

مناخدن رقم رقم

$(4)^{16} = (0100)^{2}$

$(A)^{16} = (1010)^{2}$

$(F)^{16} = (1111)^{2}$

$(4AF)^{16} = (0100\ 1010\ 1111)^{2}$

*** من ستة عشري إلى عشري:**

$(4AF)^{16} = (?)^{10}$

من اليمين يعم الحج: $(15*16)^0 + (10*16)^1 + (4*16)^2 = 15 + 160 + 1024 = (1199)^{10}$

طبعاً هون عوضنا الـ F بـ 15 و A بـ 10

Bits, Nibbles and Bytes - Binary

متل ما حكينا من قبل الـ bit هو عبارة عن 0 أو 1

الـ Byte = 8bits

ويمكن ان يمثل $2^8 = 256$ احتمال

$2^8 - 1 = 1111\ 1111$

الـ Nibble = 4bits يعني نص Byte

ويمكن أن يمثل $2^4 = 16$ احتمال

$2^4 - 1 = 1111$

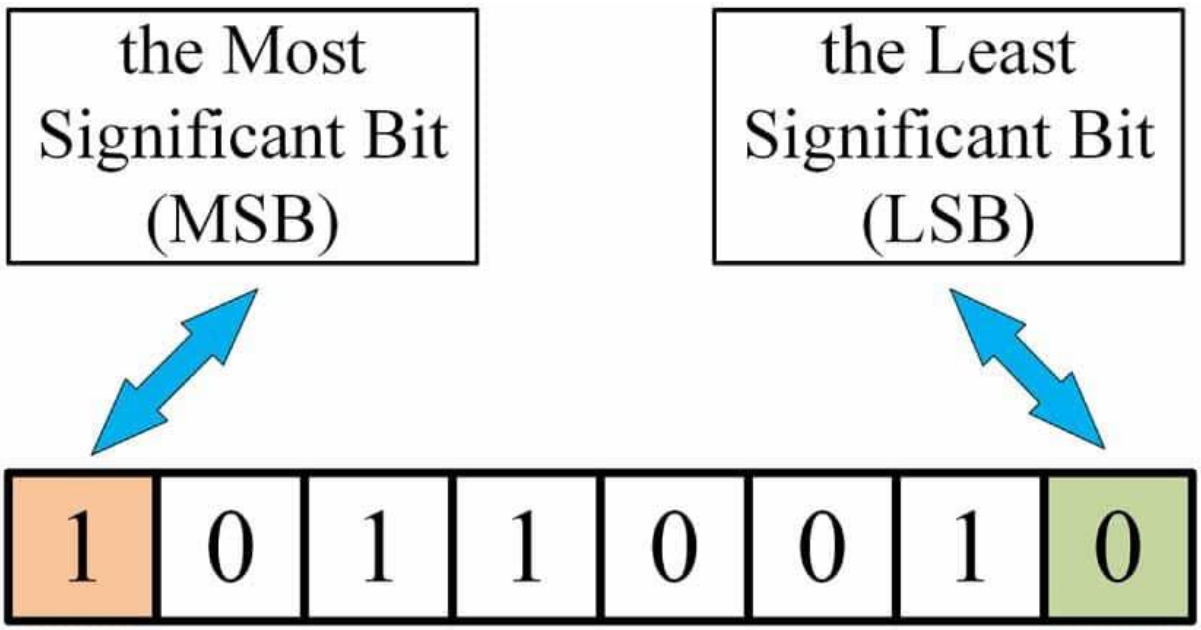
ملاحظة: لتخزين حرف بالـ hex بيحتاج bits4 يعني nibble

الـ **Micro processers** بتتعامل مع البيانات بأجزاء اسمها "word"

حجم **الـword** بيعتمد على بنية (معمارية) المعالج مثلا: إذا كان processer-64 بيكون word = 8bytes = 64bits

LSb & MSb

- البت الموجود بالعامود الأول اسمه Least Significant bit :(LSb)
- البت الموجود بالعامود الأخير اسمه Most Significant bit :(MSb)



وحدات القياس (Unit Perfix)

-كيلو بايت | : kilo

$Kb1 = 1024 = 2^{10}$

-ميغا بايت | :mega

$Mb1 = 1,048,576 = 2^{20}$

-غيغا بايت | :giga

$Gb1 = 1,073,741,824 = 2^{30}$

-تيرا بايت | :tera

$Tb1 = 1,099,511,627,776 = 2^{40}$

Binary Addition

الجمع في الثنائي بيشبه الجمع بالعشري

+1011

=0011

1110

الـ 1+1 = بنحط باليد واحد

0 = 0+0

1 = 0+1

ممكن نتعامل بعدد ثابت ومساوي للمطروح أو المجموع لكن إذا كان غير مساوي أو اكبر هون بصير شي اسمه overflow إذا صار بعد الجمع زيادة بالأرقام.

Signed-Binary Numbers

نحنا كنا بنتعامل مع الأرقام الموجبة من الصفر وما فوق ، طيب بالعمليات الحسابية الطبيعية في سالب كمان؛

في أكتر من طريقة بالباينري الأنظمة المختلفة بتستخدمها مشان تمثل الأعداد السالبة مثل: complement, sign-magnitude

Sign-magnitude Numbers

الـ MSb بيكون هو إلى بيدل عالاشارة والبتات الأخرى هي إلي بتمثل الحجم والرقم.

إذا كان الـ0 = MSb معناها الإشارة -

وإذا كان 1 يعني الإشارة +

مثال:

(0)011 = -3

(1)011 = +3

بس هي الطريقة فيها سلبيات منها عدم القدرة على الجمع والطرح.

Complement Numbers*

كمان الـ MSb هو إلى بيدل عالإشارة

اذا كان 0 : +

واذا كان 1 : -

نتحقق من الموجب والسالب من خلال طرحهما والسلبي يكون '1'

الخطوات:

-أول الشي الصفار كلها بتنزل حتى نشوف رقم 1 مثال:

$(100100)^2$

$(xxxx00)^2$

-بينزلو بس صفرين

$(00)^2$

-وبعدها مننزل الواحد متل ما هو

$(xxx100)^2$

-آخر الشي منعكس الأرقام البقية وهيك بصير الرقم:

$(011100)^2$

طريقة تانية:

اسمها taking the tow's Complement

أول الشي بنعكس الأرقام وبعدين منضيف 1 على الـ LSb

مثال:

$(0010)^2$

هيك بصير:

$(1101)^2$

منضفلو واحد:

$(1101)^2 + 1 = (1110)^2$

هاد أهم شي بيلزمنا من الـ Numbers System