Pannon Egyetem
Képfeldolgozás és Neuroszámítógépek Tanszék

Digitális Rendszerek és Számítógép Architektúrák

1. előadás: Számrendszerek,
Nem-numerikus információ ábrázolása

Előadó: Vörösházi Zsolt
Szolgay Péter

Jegyzetek, segédanyagok:

- Könyvfejezetek:
  - http://www.knt.vein.hu
    → Oktatás → Tantárgyak → Digitális Rendszerek és Számítógép Architektúrák (nappali)
    (chapter02.pdf)
- Fóliák, óravázlatok .ppt (.pdf)
- Feltöltésük folyamatosan

Információ ábrázolás:

- A) Számrendszerek:
  - I.) Egész típusú:
    - előjel nélküli,
    - 1-es komplemens,
    - előjeles 2-es komplemens számrendszerek
  - II.) Fixpontos,
  - III.) Lebegőpontos (IBM-32, DEC-32, IEEE-32),
  - Excess kód (exponens kódolására)
- B) Nem-numerikus információ kódolása
  - Hibajavítás és detektálás (Hamming kód)

A) Számrendszerek

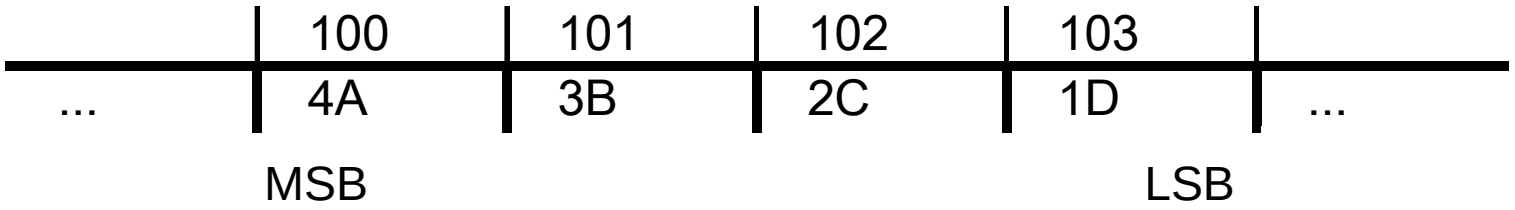
Endianitás (endianness)

- A számítástechnikában, az **endianitás** (byte-sorrend a jó fordítás) az a tulajdonság, ami bizonyos adatok - többnyire kisebb egységek egymást követő sorozata - tárolási és/vagy továbbítási sorrendjéről ad leírást (pl. két protokoll, vagy busz kommunikációja). Ez a tulajdonság döntő fontosságú az integer értékeknek a számítógép memóriájában byte-onként való tárolása (egy memória címhez relatívan), továbbítása esetében
- Byte sorrend megkötés:
  - Big-Endian formátum
  - Little-Endian formátum

Háttér: Az eredeti angol kifejezés az *endianness* egy utalás arra a háborúra, amely a két szembenálló csoport között zajlik, akik közül az egyik szerint a lágytojás nagyobb végét (big-endian), míg a másik csoport szerint a lágytojás kisebb végét (little-endian) kell feltörni. Erről Swift ír a *Gulliver Utazásai* című könyvében.

„Nagy a végén” - Big-endian

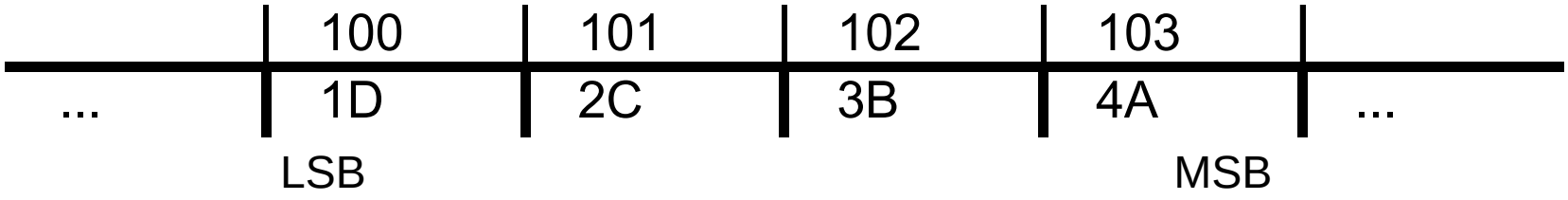
- A 32 bites egész értéket, (ami legyen „4A 3B 2C 1D”, a 100 címtől kezdve) tároljuk a memóriában, **1 byte-os** elemi tárolókból, 1 byte-onként növekvő címekkel rendelkezik, ekkor a tárolást a következők szerint végzi:



- Ebben az esetben, a „legjellemzőbb” byte - erre általában az ismert angolkifejezést "most significant byte" használják a számítástechnikában (rövidítve *MSB*, ami itt a „4A”) - a memóriában az *legalacsonyabb címen van tárolva*, míg a következő "jellemző byte" (3B) a következő, egyel nagyobb címen van tárolva, és így tovább.
- Bit-reversed format!
- Pl: beágyazott processzorok FPGA-n (MicroBlaze, PowerPC)

„Kicsi a végén” - Little-endian

- Ekkor a 32-bites „4A 3B 2C 1D” értéket a következő módon tárolják
- 100 101 102 103...”1D 2C 3B 4A”...
  Így, a kevésbé jellemző ("legkisebb") byte (az angol least significant byte rövidítéséből *LSB* néven ismert) az első, és ez az 1D, tehát a *kis vég kerül előre:*



- Általánosan használt formátum

I.) Egész típusú számrendszer:

- Bináris számrendszer: 1 / 0 (I / H, T / F)
- N biten 2^N lehetséges érték reprezentálható
- Összehasonlító táblázat:

Table 2.1. Number of Representable Values.

Number of Bits	*Number of Representable Values*	*Machines. Uses*
4	16	4004, control
8	256	8080, 6800 control, communication
16	65.536	PDP11, 8086, 32020
32	4.29×10^9	IBM 370, 68020. VAX11/780
48	1.41×10^{14}	Unisys
64	1.84×10^{19}	Cray, IEEE (dp)

a.) előjel nélküli egész:

- Unsigned integer: $$V_{\text{UNSIGNEDINTEGER}} = \sum_{i=0}^{N-1} b_i \times 2^i$$
- ahol b_i az *i*-edik pozícióban lévő ‘0’ vagy ‘1’
- Reprezentálható értékek határa: 0-tól 2^N-1 -ig
- Helyiértékes rendszer
- Negatív számok ábrázolása nem lehetséges!

- **Pl:**

101101 $\Rightarrow$1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 45

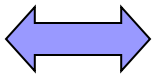
b.) előjeles (kettes komplemens) rendszer:

- 2's comp:

$$V_{\text{2'S COMPLEMENT}} = -b_{N-1} \times 2^{N-1} + \sum_{i=0}^{N-2} b_i \times 2^i$$

- reprezentálható értékek határa: –(2^(N-1)) től 2^(N-1)–1 ig
- Ha MSB='1', negatív szám
- Fólia: 2.2 táblázat
- Pl. 101101 ⇒ –1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = –19
- Pl.

$$\begin{array}{r} 100000000 \\ -\ 01001100 \\ \hline 10110100 \end{array} \qquad \begin{array}{r} 256 \\ -\,76 \\ \hline 180 \end{array}$$



$$\begin{array}{rl} 01001100 & \\ 10110011 & \text{1's kompl.} \\ +\qquad 1 & +1 \\ \hline 10110100 & \qquad -76 \end{array}$$

2.2 táblázat:

Table 2.2. 8-Bit Two's Complement Representations.

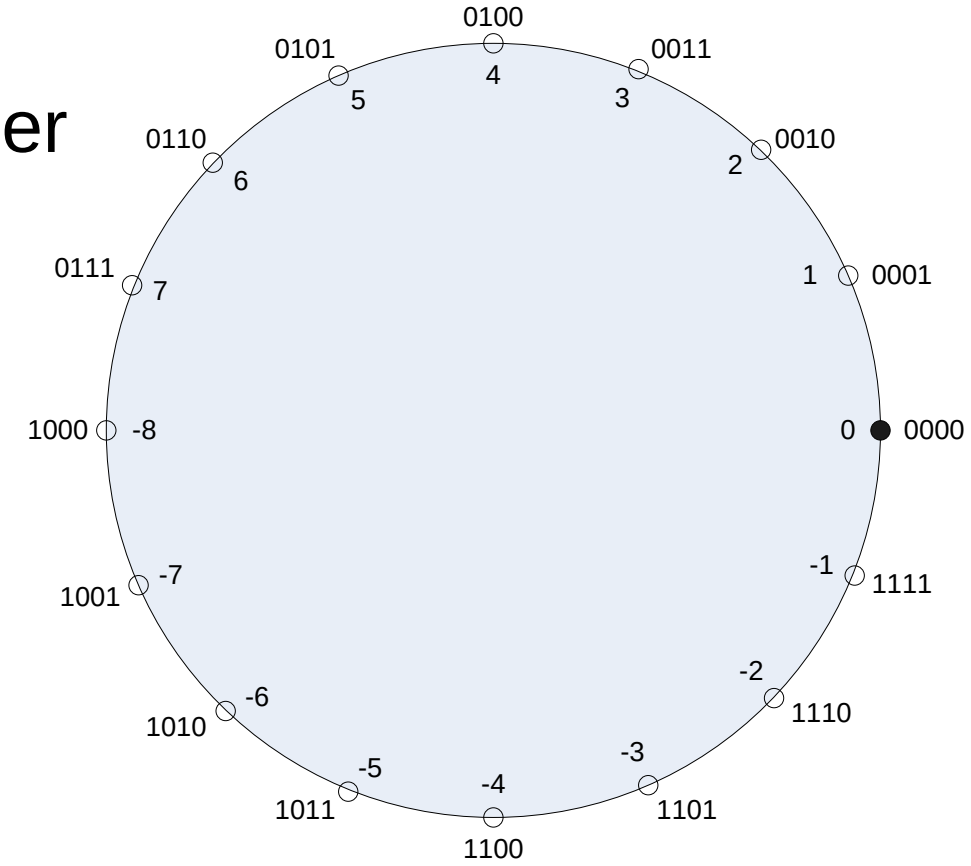
Bit Pattern	*Value*	*Note*
01111111	127	Largest representable value.
01111110	126	
01111101	125	
...	...	
...	...	
00000010	2	Note that leading zero indicates
00000001	1	positive number.
00000000	0	Unique representation of **zero.**
11111111	–1	Minus one is always all ones.
11111110	–2	Note that leading one indicates
11111101	–3	negative number.
...	...	
...	...	
10000010	–126	
10000001	–127	
10000000	–128	Smallest (most negative) representable value.

Pl: Körkörös számláló (circular nature):

- 4 bites 2’s komplemens rendszer
- Overflow:

  0111 -> 1000
- Underflow:

  1000 -> 0111



c.) 1-es komplemens rendszer:

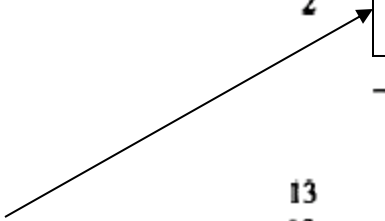
- V értékű, N bites rendszer: $\mathbf{2^N–1–V}$.
- „0” lesz ott ahol „1”-es volt, „1”-es lesz ott ahol „0” volt (mivel egy szám negatív alakját, bitjeinek kiegészítésével kapjuk meg).
- csupán minden bitjét negálni, (gyors műveletet)
- Értékhatár: $2^{(N-1)}–1$ től $–(2^{(N-1)}–1)$ ig terjed,
- Nem helyiértékes rendszer,
- kétféleképpen is lehet ábrázolni a zérust!! (ellenőrzés szükséges)
- end-around carry: amelyben a részeredményhez kell hozzáadni a végrehajtás eredményét

Példa:

Example 2.3: ***One's complement*** arithmetic: Consider a 6-bit one's complement system. Represent 15. –15. 13. and –13 in this system. Then perform the following additions: 15 + 13, 15 + **(–13),** 13 + **(–13),** and 13 + (–15).

The numbers are derived in a simple fashion:

Decimal Value	*One's Complement*	*Comment*
15	001111	Positive numbers same as two's complement.
–15	110000	Complement bits to negate.
13	001101	Positive numbers same as two's complement.
–13	110010	Complement bits to negate.
Now for the additions:		
15	001111	This proceeds just like the two's
+ 13	+ 001101	complement version.
28	0 011100	No cany out: number is correct, and the result is as we expect.
15	001111	The addition is done in the normal
+ –13	+ 110010	fashion, but
2	1 000001	the result of one is incorrect;
		however, the presence of a carry says we
	+ 000001	should add that as a 1 in the LSB
	000010	which gives the expected result.
13	001101	This time we will add a positive number
+ –13	+ 110010	to its negative (which is just complement)
0	111111	and end up with all ones — a valid zero.
13	001101	Here the positive number is smaller
+ –15	+ 110000	than the negative number, so result
–2	0 111101	is negative; no cany — the value is correct.



end-around: cirkuláris carry, körkörös átvitel

II.) Fixpontos számrendszer

- Műveletek:
  - +, - : ugyanaz, mint az egész szám rdsz. esetén
  - *, / : meg kell bizonyosodni arról, hogy a tizedespont helyén maradt-e

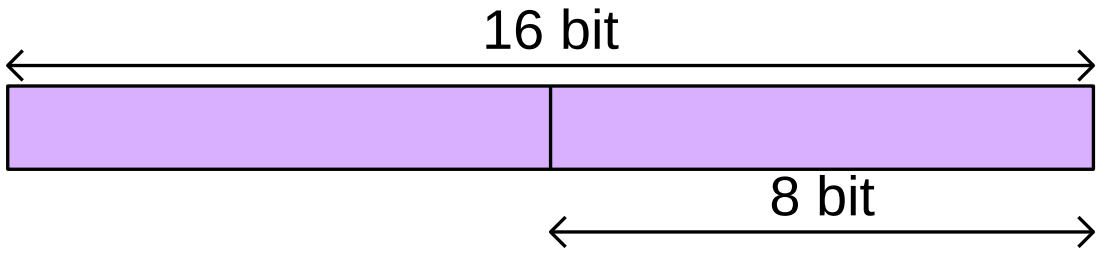
$$V_{\text{FIXEDPOINT}} = -b_{N-1} \times 2^{N-p-1} + \sum_{i=0}^{N-2} b_i \times 2^{i-p}$$

- p: radix (tizedes) pont helye, tizedes jegyek száma
- *differencia*, $\Delta r = 2^{-p}$ (számrendszer finomsága)
  - *Ha p=0 → Δr=1, egész rendszer, különben fixpontos*
- Alkalmazás: pl. jelfeldolgozás (DSP – Texas Instruments)

Példa: fixpontos rendszer

- Kérdés: Legyen egy 16 bites 2's comp. fixpontos rdsz. ahol p=8. V(smallest)=?, V(largest)=?, Δr = *? (decimális értékben megadva)*

- Megoldás:



- V(smallest absolute)=00000000.00000001=2^{-8}=$\underline{0{,}390625*10^{-2}}$
- V(largest absolute)=01111111.11111111=$\underline{\sim 128}$
- V(largest negative)=10000000.00000000=-2^{7}=$\underline{-128}$
- Differencia $\Delta r = 2^{-8}$=$\underline{0{,}390625*10^{-2}}$
- !!DE V(zero)=00000000.00000000 vagy 11111111.11111111

Excess kód rendszer

- Lebegőpontos számok *kitevőit* (*exponens*-eit) tárolják / kódolják ezzel a módszerrel
- S: a reprezentálni kívánt érték (eredmény), amit tárolnunk
- V: a szám valódi értéke, E: az excess
- S = V + E.
- Két számot összeadunk, akkor a következő történik:

$$S1 + S2 = (V1 + E) + (V2 + E) = (V1 + V2) + 2 \times E$$

- pontos eredmény: [(V1+V2) + E] (ki kell vonnunk E-t!)
- Fólia: 2.4, 2.5, 2.6-os példák

Példa 2.4:

Example 2.4: Number representation in excess codes: What is the representation of $+37_{10}$ in an 8-bit excess *128* code? What is the representation of -23_{10} in an 8-bit excess 128 code? What is the sum of the two numbers, in the 8-bit excess *128* code?

An 8-bit unsigned number can represent values between **0** and ***255.*** The excess representation can then represent values from **–**128 to **+127.**

+ 128	10000000	This is the excess.
+ 37	00100101	The value to be represented.
165	10100101	The representation of 37_{10} in excess 128 code.
+ 128	10000000	This is the excess.
– 23	00010111	The value to be represented.
105	01101001	The representation of -23_{10} in excess **128** code.
165	10100101	This is +37 in excess **128.**
+ 105	+ 01101001	This is –**23** in excess 128.
270	1 00001110	Note the carry out in this operation. **270** is too big to represent in **8** bits; to correct for the 2 **x** E that is in this sum, subtract **128.**
– 128	– 10000000	In binary, is this add or subtract?
142	10001110	This is the representation of **14,** the correct result. in excess **128.**

Táblázat 2.3: BCD

Table 2.3. Binary Coded Decimal (RCD) Representations.

Bit Pattern	*Value*
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	Not valid
1011	Not valid
1100	Not valid
1101	Not valid
1110	Not valid
1111	Not valid

Példa 2.5:

Example 2.5: BCD excess 3 system: Consider *a* system that works with 3-digit decimal numbers, and it stores the digits in excess 3 format. What is the representation of **573?** What is the representation of **142?** Add the two numbers, and give the correct result in excess 3 format.

The numbers are handled on a digit-by-digit basis, with the excess being included with each digit:

Decimal	Binary	
+3 3 3	0011 0011 0011	This is the excess.
5 7 3	0101 0111 0011	And the number to be represented.
8 10 6	1000 1010 0110	The excess 3 representation.
+3 3 3	0011 0011 0011	This is the excess.
1 4 2	0001 0100 0010	This is the number to be represented.
4 7 5	0100 0111 0101	The excess 3 representation.
573	1000 1010 0110	Do the addition in decimal and in
+ 142	0100 0111 0101	binary. Correct as needed to
		make output correct.
715	1100 1 0001 1011	

Carry out of second set of 4 bits indicates that the most significant digit should be incremented by one. Also indicates that this value is **correct** as it stands (since 2 × E = 6 and the **carry** out indicates that the number overflowed into the next digit) so we need to add 3. Therefore, the MSD needs to **be** incremented by one and decremented by 3; the middle digit needs to have 3 added; and the **LSD** needs to be decremented by 3.

(-3+1)	(+3)	(-3)
−0010	+0011	−0011
1010	0100	1000
10	4	8

Which is the correct excess 3 representation for 715_{10}.

MSD: Most significant digit

LSD: Least significant digit

III.) Lebegőpontos rendszer:

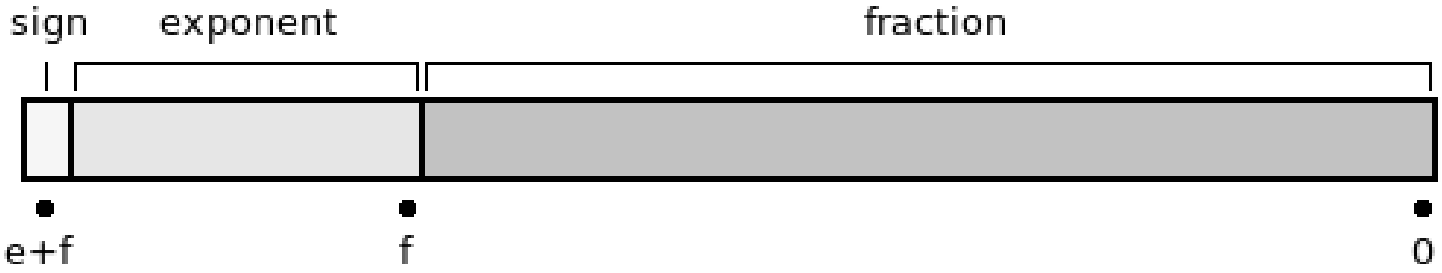
- 7 különböző tényező: *a számrendszer alapja, előjele és nagysága, a mantissza alapja, előjele és hosszúsága, ill. a kitevő alapja*.
- matematikai jelölés:

$$(\text{előjel})\,\text{Mantissza} \times \text{Alap}^{\text{Kitevő}}$$

- Fixpontosnál nagyságrendekkel kisebb vagy nagyobb számok ábrázolására is mód van:
  - Pl: Avogadro-szám: 6.022*10^23
  - Pl: proton tömege 1.673*10^-24 g

IEEE 754-1985

- Szabvány bináris lebegőpontos számok tárolásánál, amely tartalmazza még a:
  - negatív zérust: –0 = 111…1
  - Normalizálatlan számok
  - NaN: nem szám
  - – /+ végtelen



Sign-magnitude format („előjel-hossz” formátum): előjel külön kerül tárolásra (MSB), exponens eltolt (Excess-el kódolt), törtrész utána következik.

Lebegőpontos rendszer jellemzői

- Számrendszer / kitevő alapja: r_b r_e
- Mantissza értéke: $V_M = \sum_{i=0}^{N-1} d_i \times r_b^{i-p}$
  - Maximális: $V_{M_{max}} = 0.d_m d_m d_m ... = (1 - r_b^{-N})$
  - Minimális: $V_{M_{min}} = 0.100 .. = 1/r_b$
  - Radix pont helye: p
  - Mantissza bitjeinek száma: m
- Exponens értéke (max / min): V_E $V_{E_{max}}$ $V_{E_{min}}$
- Lebegőpontos szám értéke: $V_{FPN} = (-1)^{SIGN} V_M \times r_b^{V_E}$

Normalizált lebegőpontos rendszer jellemzői

- Ábrázolható maximális érték: $V_{FPN(MAX)} = V_{M(MAX)} \times r_b^{V_{E(MAX)}}$
- Ábrázolható minimális érték: $V_{FPN(MIN)} = V_{M(MIN)} \times r_b^{V_{E(MIN)}}$
- Legális mantisszák száma: $NLM_{FPN} = (r_b - 1) \times r_b^{m-1}$
- Legális exponensek száma: $NLE_{FPN} = V_{E(MAX)} + \left|V_{E(MIN)}\right| + 1_{ZERO}$
- Ábrázolható értékek száma: $NRV_{FPN} = NLM_{FPN} \times NLE_{FPN}$

- Normalizálás: mantissza értékét általában [0…~1] közé
- Pl: $32\ 768_{10}$ = $0.32768*10^5$ = $3.2768*10^4$ = $32.768*10^3$ = $327.68*10^2$ = $3267.8*10^1$

Példa: normalizált lebegőpontos rendszer

- Adott: Legyen $r_b = 10$, $r_e = 10$, $m = 3$, $e = 2$
- Kérdés: jellemző paraméterek?
- Megoldás:

$$V_{M(MAX)} = 0.999 = 1.000 - 10^{-3}$$
$$V_{M(MIN)} = 0.100$$
$$V_{E(MAX)} = (r_e \wedge e - 1) = 99$$
$$V_{E(MIN)} = -(r_e \wedge e - 1) = -99$$
$$V_{FPN(MAX)} = 0.999 \times 10^{99}$$
$$V_{FPN(MIN)} = 0.100 \times 10^{-99}$$

$$NLM_{FPN} = 9 \times 10 \times 10 = 900 = 9 \times 10^2$$
$$NLE_{FPN} = 99 + |-99| + 1_{ZERO} = 199$$
$$NRV_{FPN} = 2 \times 900 \times 199 = 358{,}200$$

Normalizált lebegőpontos rdsz.

Table 2.4. 6-Bit Normalized Floating Point System, Base 2.

$r_b = 2,\ r_e = 2,\ m = 4,\ e = 2$

				$V_E \rightarrow$	00	01	10	11
				$2^{V_E} \rightarrow$	1	2	4	8
V_M base 2				V_M	$V_M \times 2^{V_E}$			
1	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	1	2	4
1	0	0	1	$\frac{9}{16}$	$\frac{9}{16}$	$1\frac{1}{8}$	$2\frac{1}{4}$	$4\frac{1}{2}$
1	0	1	0	$\frac{5}{8}$	$\frac{5}{8}$	$1\frac{1}{4}$	$2\frac{1}{2}$	5
1	0	1	1	$\frac{11}{16}$	$\frac{11}{16}$	$1\frac{3}{8}$	$2\frac{3}{4}$	$5\frac{1}{2}$
1	1	0	0	$\frac{3}{4}$	$\frac{3}{4}$	$1\frac{1}{2}$	3	6
1	1	0	1	$\frac{13}{16}$	$\frac{13}{16}$	$1\frac{5}{8}$	$3\frac{1}{4}$	$6\frac{1}{2}$
1	1	1	0	$\frac{7}{8}$	$\frac{7}{8}$	$1\frac{3}{4}$	$3\frac{1}{2}$	7
1	1	1	1	$\frac{15}{16}$	$\frac{15}{16}$	$1\frac{7}{8}$	$3\frac{3}{4}$	$7\frac{1}{2}$

Smallest fraction = 0.1000_2 $= \frac{1}{2}$

Largest fraction = 0.1111_2 $= \frac{15}{16}$

Smallest number = $0.1000_2 \times 2^0$ $= \frac{1}{2}$

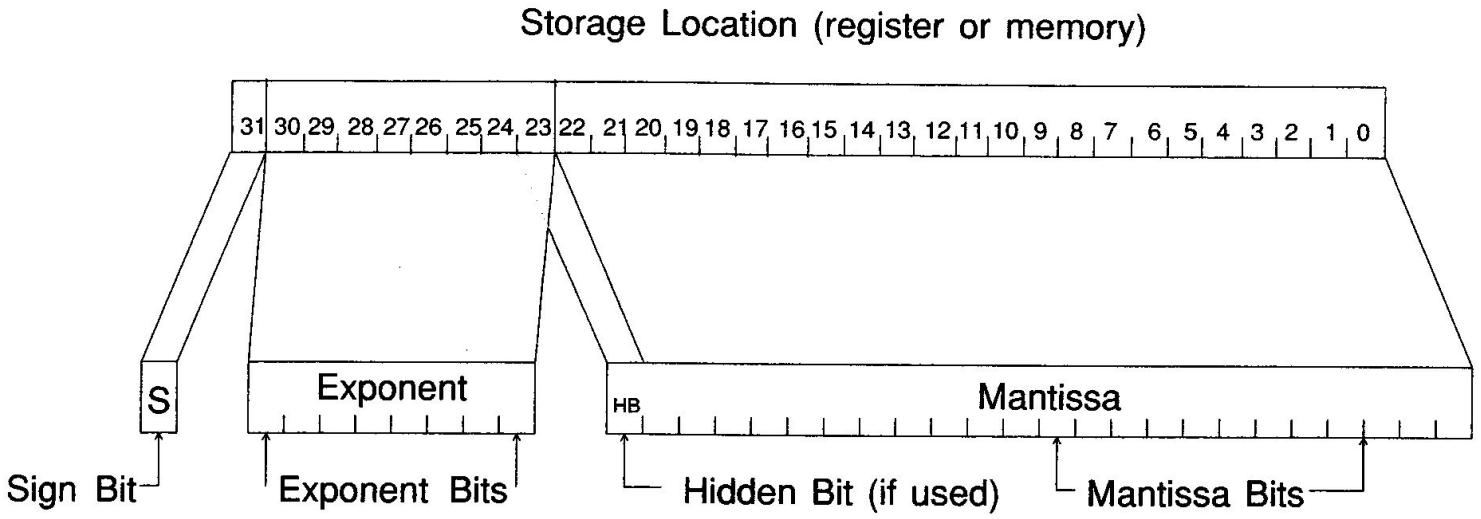
Largest number = $0.1111_2 \times 2^3$ $= 7\frac{1}{2}$

Number of fractions = $1 \times 2 \times 2 \times 2 = 8$

Number of values = 8×4 $= 32$

Lebegőpontos ábrázolás: Rejtett (hidden bit) technika

- „Vezető” egyesek számát a legtöbb rendszer tervezésekor konstans értékként definiálják (1)
  - Ilyen a *mantissza legmagasabb helyiértékű bitje*, amelyet rejtett (*hidden: HB*) bitnek hívunk, amely közvetlenül az exponensbitek mögött helyezkedik el.
  - Ezt beállítva, duplájára nő a legális mantisszák, így az ábrázolható értékek száma is. Másrészt a nullát nem könnyen tudjuk reprezentálni, mivel a legkisebb ábrázolható érték a 00 0000, ami a HB ‘1’-es konstansként való definiálása miatt $0.1000_2 \times 2^0$ = ½ -nek felel meg (m=4, e=2, r_b=r_e=2 esetén)!



Példa: 2-es alapú **DEC 32**-bites, normalizált lebegőpontos rendszer

- Adott: r_b=2, r_e=2, m=23+(1)=24 együtt, p=24 (m=p!), e=8, az exponenst tároljuk Excess-128 kódolással, és a számokat tároljuk "előjel-hossz" formátumban (~tekintsük a mantisszát pozitívnak).

$$V_{M(MIN)} = 0.\underbrace{1000\ldots_2}_{24db} = 1/2$$

$$V_{M(MAX)} = 0.1111\ldots_2 = 0.999999940395 = 1.0 - 2^{-24}$$

$$V_{E(MIN)} = -(r_e{}^{e-1} - 1) = -(2^{8-1} - 1) = -127$$

$$V_{E(MAX)} = r_e{}^{e-1} - 1 = 2^{8-1} - 1 = 127$$

$$V_{FPN(MIN)} = 0.1000\ldots_2 \times 2^{-127} = 2.9387 \times 10^{-39}$$

$$V_{FPN(MAX)} = 0.1111\ldots_2 \times 2^{+127} = 1.7014 \times 10^{38}$$

$$NLM_{FPN} = 2^{23} = 8,388,608$$

$$NLE_{FPN} = 127 + \left|-127\right| + 1_{ZERO} = 255 = (r_e{}^{e} - 1) = 2^8 - 1$$

$$NRV_{FPN} = 2^{23} \times (2^8 - 1) = 2.139 \times 10^9$$

Példa: 16-os alapú **IBM-32** bites normalizált lebegőpontos rendszer

- Adott: r_b=16, r_e=2, m=6, p=6, e=7, az exponenst tároljuk Excess-64 kódolással, és a számokat tároljuk "előjel-hossz" formátumban (tekintsük a mantisszát pozitívnak).

$$V_{M(MIN)} = 0.100000_{16} = 1/16$$

$$V_{M(MAX)} = 0.\text{FFFFFF}_{16} = 0.999999940395 = 1.0 - 16^{-6}$$

$$V_{E(MIN)} = -(r_e^{\,e-1} - 1) = -(2^{7-1} - 1) = -63$$

$$V_{E(MAX)} = r_e^{\,e-1} - 1 = 2^{7-1} - 1 = 63$$

$$V_{FPN(MIN)} = 0.100000 \times 16^{-63} = 8.636 \times 10^{-78}$$

$$V_{FPN(MAX)} = 0.\text{FFFFFF}_{16} \times 16^{+63} = 7.237 \times 10^{75}$$

$$NLM_{FPN} = 15 \times 16^5 = 15,728,640$$

$$NLE_{FPN} = 63 + |-63| + 1_{ZERO} = 127 = 2^7 - 1$$

$$NRV_{FPN} = 15 \times 16^5 \times (2^7 - 1) = 1.9975 \times 10^9$$

Bővebb tartomány mint a DEC!

7%-al kevesebb mint a DEC!!

Példa: **IEEE-32** bites normalizált lebegőpontos rendszer

- Adott: r_b=2, r_e=2, m=24, de p=23! (+HB!), , e=8, az exponenst tároljuk Excess-127 kódolással, és a számokat tároljuk "előjel-hossz" formátumban (~tekintsük a mantisszát pozitívnak).

$$V_{M(MIN)} = 1.\underbrace{000\ldots_2}_{23db} = 1$$

$$V_{M(MAX)} = 1.111\ldots_2 = 1.99999988 = 2.0 - 2^{-23}$$

$$V_{E(MIN)} = -126$$ // Zérus pontosabb ábrázolása miatt

$$V_{E(MAX)} = 127$$

$$V_{FPN(MIN)} = 1.000\ldots_2 \times 2^{-126} = 1.1755 \times 10^{-38}$$ // 4 x DEC!

$$V_{FPN(MAX)} = 1.111\ldots_2 \times 2^{+127} = 3.4028 \times 10^{38}$$ // 2 x DEC!

$$NLM_{FPN} = 2^{23} = 8,388,608$$

$$NLE_{FPN} = 127 + |-126| + 1_{ZERO} = 254$$

$$NRV_{FPN} = 2^{23} \times (2^8 - 1) = 2.131 \times 10^9$$

- *DEC*: zérushoz közelítve szakadás (az első érték aránytalanul messze van)
- Ezt küszöböli ki az *IEEE* rendszer (Normalizálatlan tartományban lineárisan tart a zérushoz): ezért használja a V_E= 126

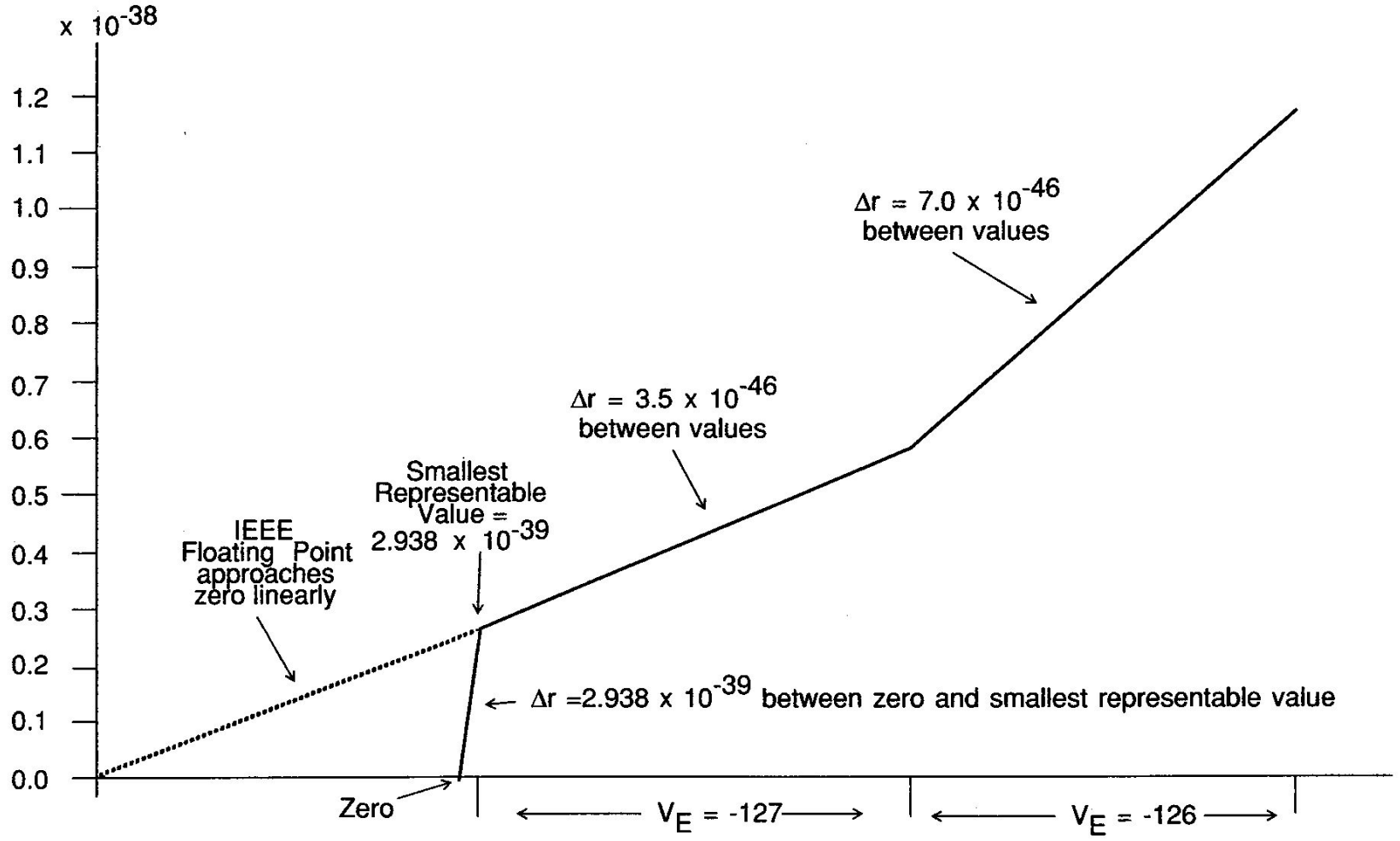


Figure 2.3. Values of the DEC Normalized Floating Point System Near Zero.

Példa: **IEEE-32** bites normalizált lebegőpontos rendszer (folyt.)

- V_E = [-127, 126] = [1, 254] eltolt Excess-127 tartomány
- Speciális jelentőség:
  - V_E = 0 értékénél (zérus ábrázolása)
  - V_E = 255 értékénél lehetőség van bizonyos információk tárolására:

V_E	Előjel	Ábrázolás jelentése
≠ 0	1,0	Nem egy szám (NaN)
0	0	+∞
0	1	-∞

B) Nem-numerikus információ kódolása

Nem-numerikus információk

- Szöveges,
- Logikai (Boolean) információt,
- Grafikus szimbólumokat,
- és a címeket, vezérlési karaktereket értjük alattuk

Szöveges információ

- Minimális: 14 karakterből álló halmazban: számjegy (0-9), tizedes pont, pozitív ill. negatív jel, és üres karakter.
- + ábécé (A-Z), a központozás, címkék és a formátumvezérlő karakterek (mint pl. vessző, tabulátor, (CR: Carriage Return) kocsi-vissza, soremelés (LF:Line Feed) , lapemelés (FF: From Feed), zárójel)
- Így elemek száma 46: 6 biten ábrázolható

$$\lceil \log_2 46 \rceil = 6\,\text{bit}$$

- De 7 biten tárolva már kisbetűs, mind pedig a nagybetűs karaktereket is magába foglalja

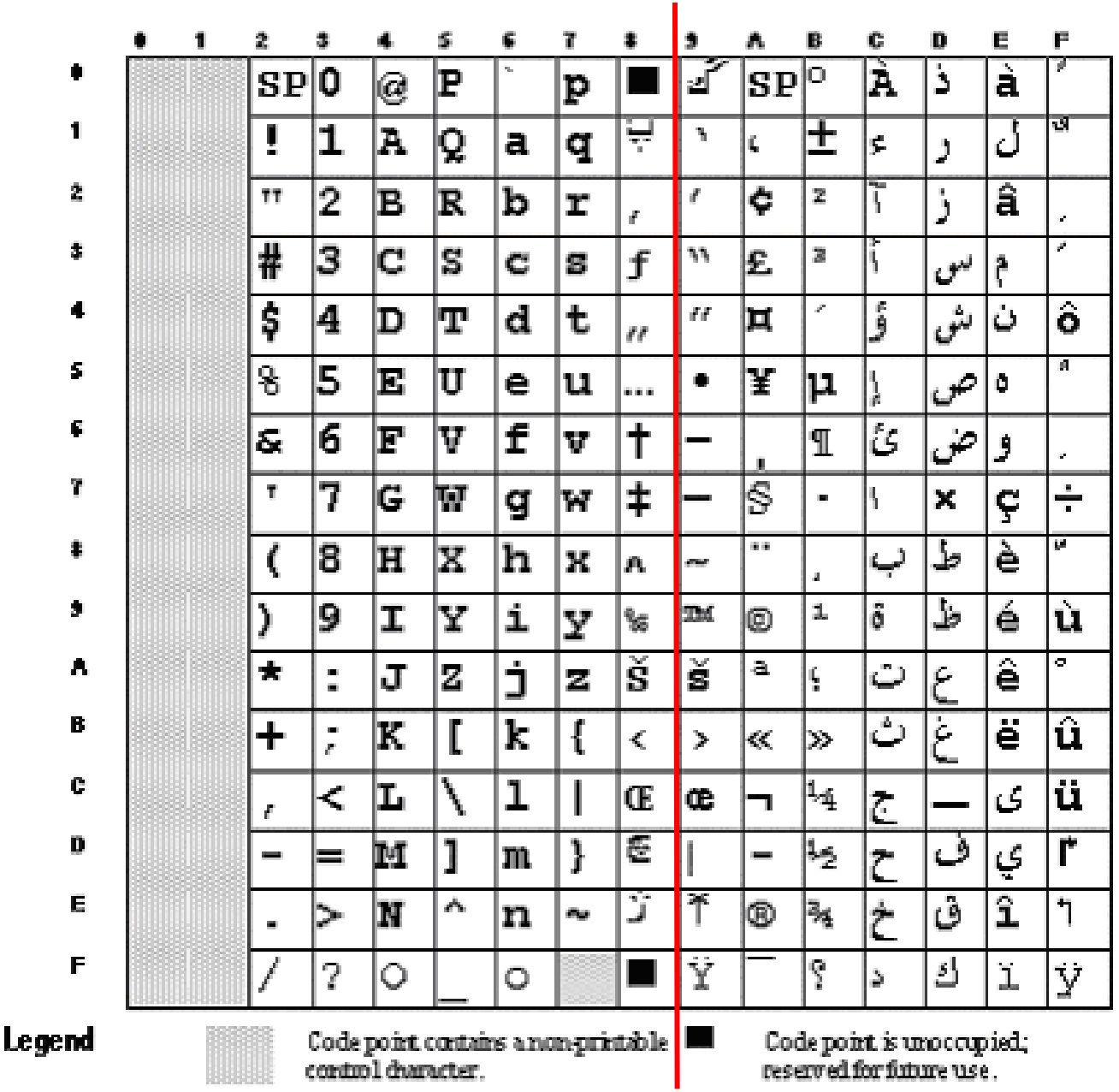
Szöveges információ kódolás

- BCD (Binary Coded Decimal): 6-biten
  - nagybetűk, számok, és speciális karakterek
- EBCDIC (Extended Binary Coded Decimal Interchange Code): 8-biten (A. Függelék)
  - + kisbetűs karaktereket és kiegészítő-információkat
  - 256 értékből nincs mindegyik kihasználva
  - Továbbá I és R betűknél szakadás van!
- ASCII (American Standard Code for Information Interchange): (A függelék) – alap 7-biten / extended 8-biten
- UTF-n (Universal Transformation Format): változó hosszúságú karakterkészlet (többnyelvűség támogatása)

EBCDIC

HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp) SP010000	& SM030000	- SP100000	ø LO810000	Ø LO820000	° SM190000	µ SM170000	¢ SC040000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA810000	j LJ010000	~	£	A 000	J LJ020000	÷ SA080000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000			000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000			000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000			' 000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	' LV010000	x SM240000	_ LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	Œ LO520000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	œ LO510000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	Ÿ LY180000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	Ý LY120000	! SP020000	Š LS220000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	1 ND011000	2 ND021000	3 ND031000
-B	. SP110000	Š SC030000	, SP060000	# SM010000	» SP180000	º SM200000	¿ SP160000	š LS210000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA200000	* SM040000	% SM020000	@ SM050000	ð LD830000	æ LA510000	Ð LD640000	¨ SD310000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP080000	) SP070000	_ SP090000	' SP050000	ý LY110000	ž LZ210000	[SM080000	] SM090000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA210000	; SP140000	> SA050000	= SA040000	þ LT830000	Æ LA520000	Þ LT840000	Ž LZ220000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	^ SD150000	? SP150000	" SP040000	± SA000000	€ SC300000	® SM530000	× SA570000	õ LO190000	ÿ LY170000	Õ LO200000	EO

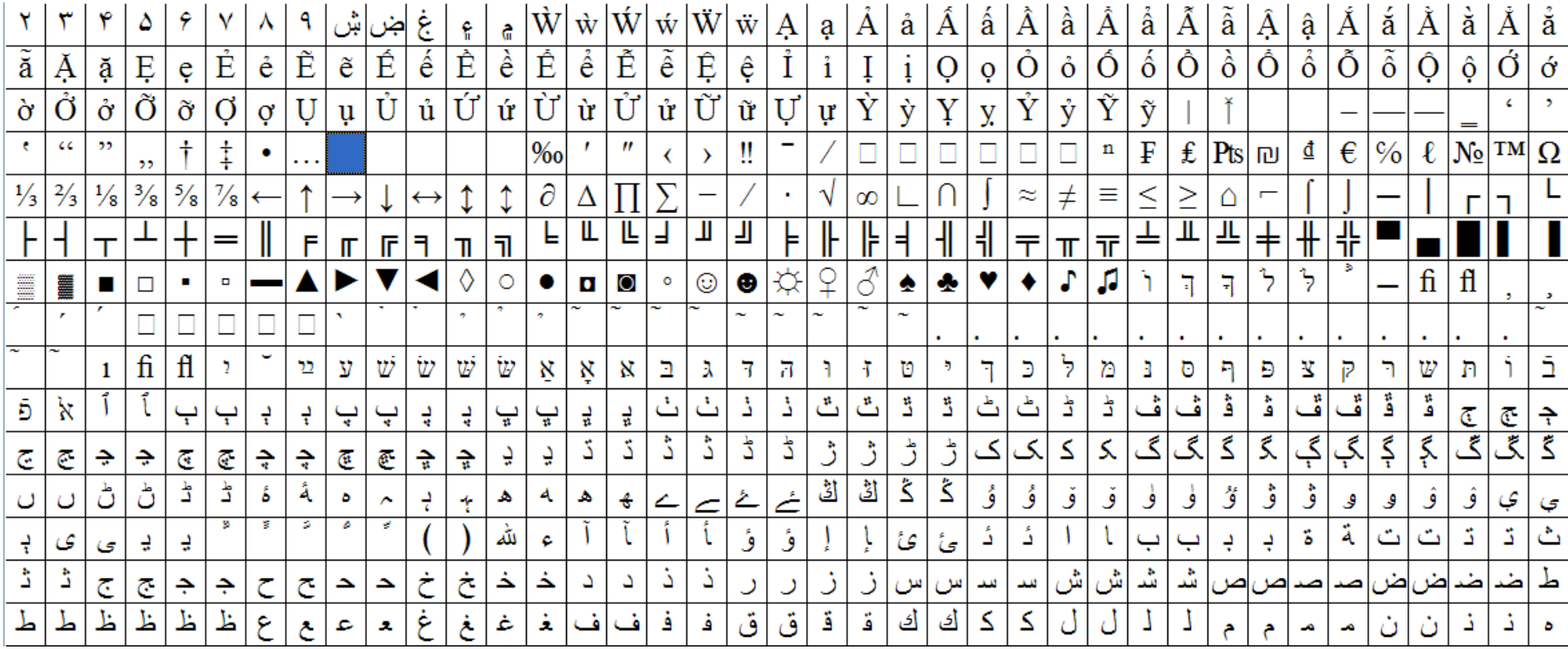
Extended ASCII (1 byte)



Unicode

Nyelvkészletek: Alap, - Latin 1/2, görög, cirill, héber, arab stb.

Általános írásjelek, matematikai, pénzügyi, mértani szimbólumok stb.



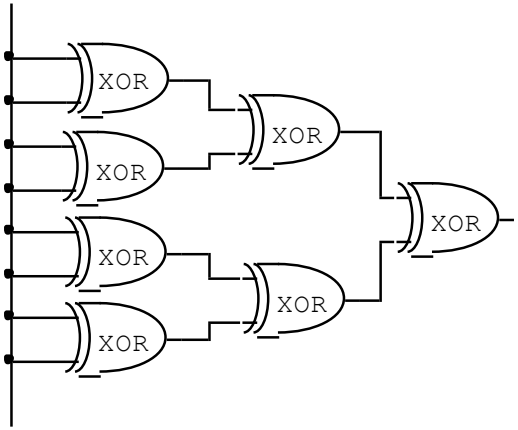
*részlet

Hibakódolás - Hibadetektálás és Javítás

- N bit segítségével 2^N különböző érték, cím, vagy utasítás ábrázolható
- 1 bittel növelve (N+1) esetén: $2^{\wedge N}$ -ről $2^{\wedge N+1}$ –re, megduplázódik
- Redundancia, detektálás, hibajavítás

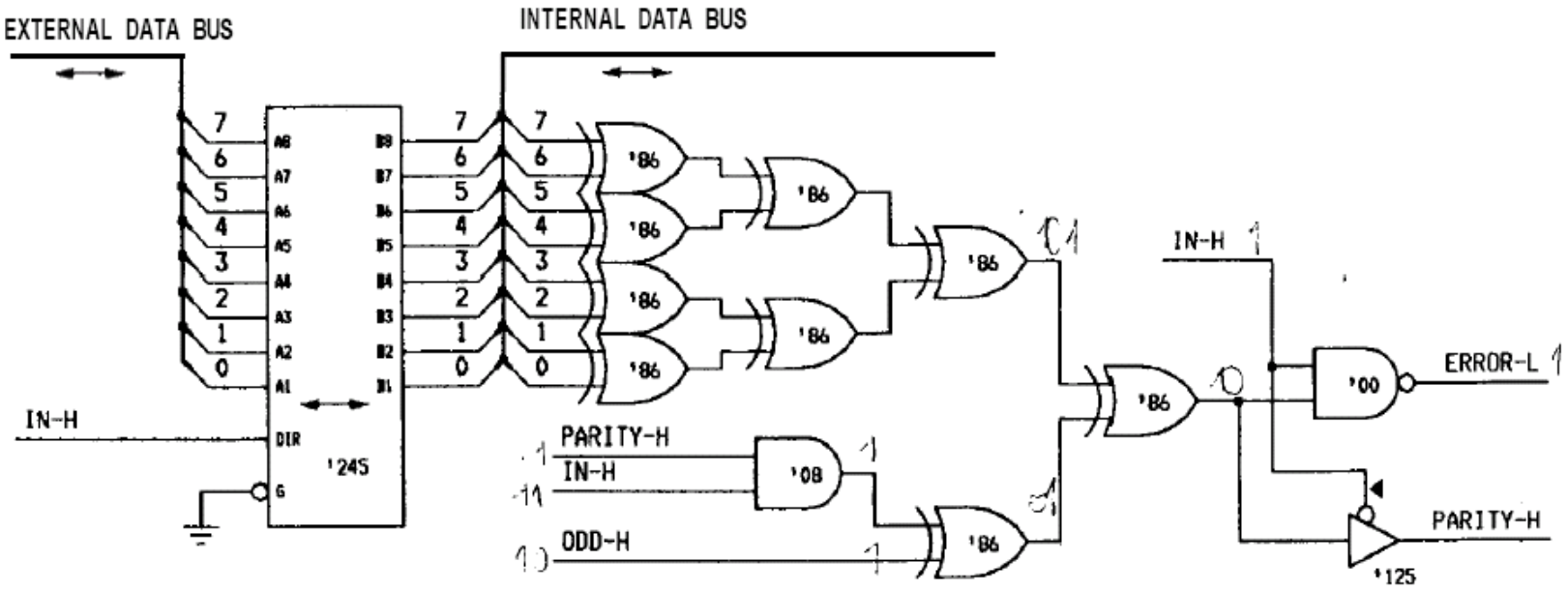
Paritás bit ellenőrzés

- Páros v. páratlan paritás: N bites információ egy kiegészítő bittel bővül → egyszeres hiba felismerése (bitek számát párosra v. páratlanra egészíti ki)
  - Pl: leragadásból: ‘0’-ból ‘1’-es lesz, vagy fordítva
  - Pl: ideiglenes, tranziens jellegű hiba
  - 8 adatbithez paritásbit generálás (XOR) (IC ‘280)



8 bites adatút kétirányú paritásbit ellenőrzéssel

- IN-H=1 → PARITY-H=1 paritás ell.
- Ha ODD-H=1 páratlan paritást használ
- Hiba esetén: ERROR-L=1

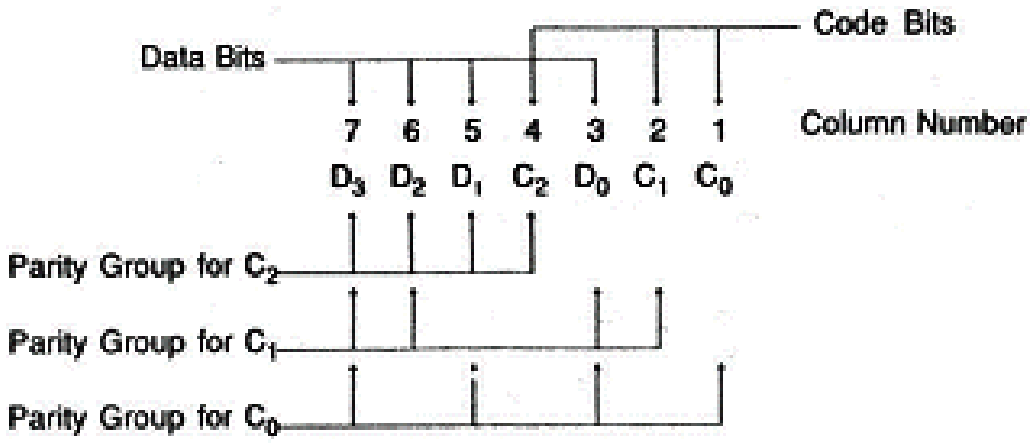


Hamming kód

- Több redundáns bittel nemcsak a hiba meglétét, és helyét tudjuk detektálni, hanem a hibás bitet javítani is tudjuk
- Hamming kód: egy biten tároljuk a bitmintázatok azonos helyiértékű bitjeinek különbségét, tehát egybites hibát lehet vele *javítani*.

Hamming kódú kódszó konstruálása (pl. 7 –bites kódszóra)

- **2^N-1 bites Hamming kód: N kódbit, 2^N-N-1 adatbit**
- Összesen 7 biten ***4*** *adatbitet (D0,D1,D2,D3),* ***3*** *kódbittel (C0,C1,C2)* kódolunk
- C_i kódbitek a bináris súlyuknak megfelelő bitpozíciókban
- A maradék pozíciókat rendre adatbitekkel töltjük fel (D_i)

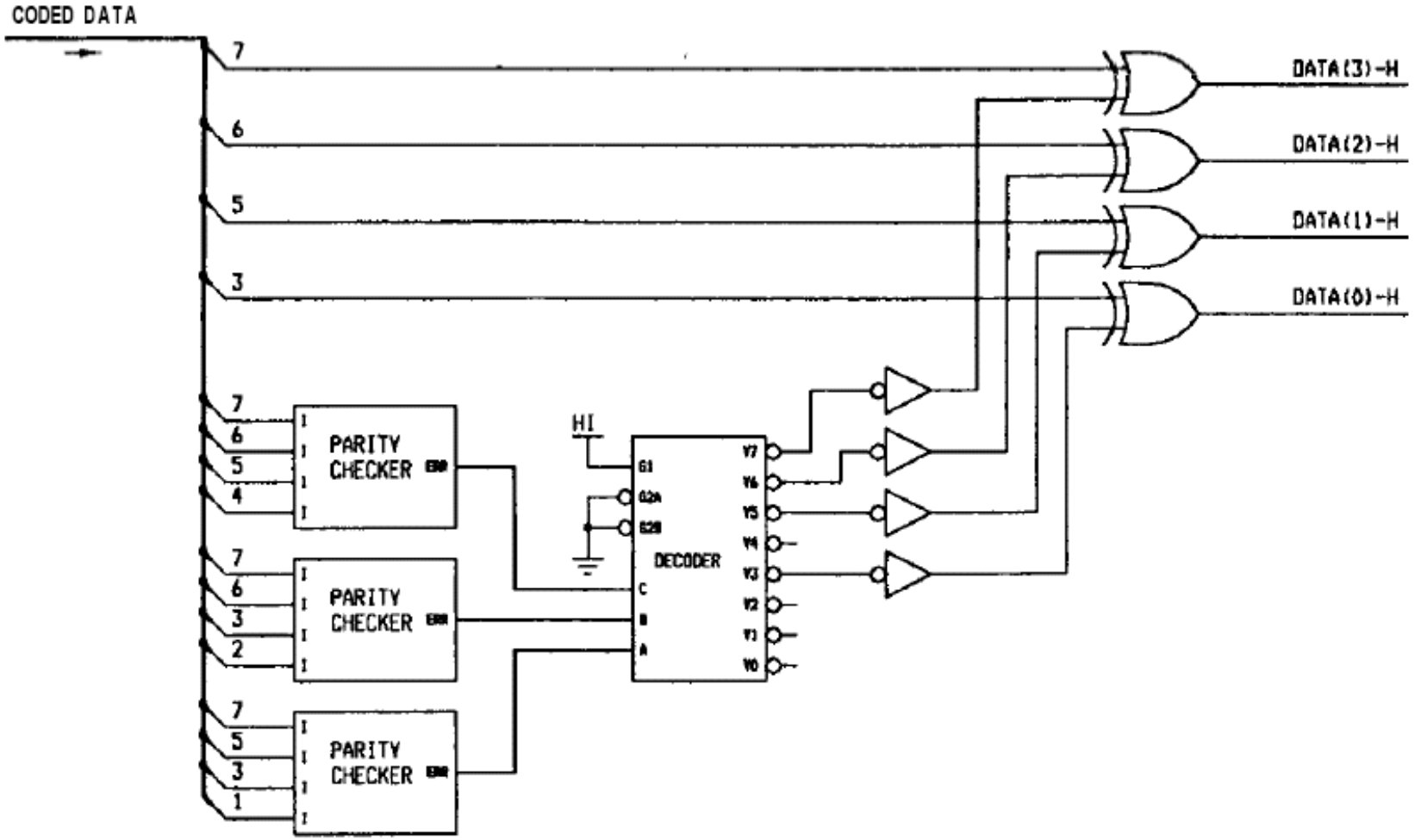


Paritás-csoportok	Bit pozíciók	Bitek jelölései
0	1, 3, 5, 7	C0, D0, D1, D3
1	2, 3, 6, 7	C1, D0, D2, D3
2	4, 5, 6, 7	C2, D1, D2, D3

Hamming kódú hibajavító áramkör tervezése

- 3 kódbitünk van, így írásnál 3 paritásbit generáló-, míg olvasásnál 3 paritás-ellenőrző áramkör kell.
- Példa: bemeneti adatbit-mintázatunk 0101.
  - Mivel páratlan paritást alkalmazunk, a megfelelő helyen szereplő kódbitekkel kiegészítve a következő szót kapjuk: 0100110. Ha nincs hiba, a paritásellenőrzők (C0, C1, C2) kimenete 000, minden egyes paritáscsoportra.
  - Hiba esetén például, ha az input mintázat 0100010, akkor a paritásellenőrző hibát észlel. Ugyan C2. paritásbitcsoport rendben, de a C1. és C0. hibás:
  - 011 az azonosított minta a harmadik oszlopban (**D0** helyén).
  - Javításként *invertáljuk* a 3. bitpozícióban lévő bitet. 0100010 $\Rightarrow$ 0100110. Ekkor a kódbitek a következőképpen módosulnak a páratlan paritásnak megfelelően: C0=1, C1=1 és C2=0.

7-bites Hamming kódú hibajavító áramkör felépítése



Példa:

Hamming kód (DEB-el) 8 adatbitre: mi a helyes ábrázolása 8 biten a 01011100 adatbit mintázatnak. Szükséges 8 adatbit (D0-D7), 4 kódbit (C0-C3) és egy kettős hibajelző bit (DEB). Páratlan paritást alkalmazunk. (BW-binary weight jelenti az egyes oszlopok bináris súlyát, 1,2, 4 ill 8 biten).

13	12	11	10	9	8	7	6	5	4	3	2	1	Oszlopszám
DEB	D7	D6	D5	D4	C3	D3	D2	D1	C2	D0	C1	C0	
	1	1	1	1	1	0	0	0	0	0	0	0	BW, 8bit
	1	0	0	0	0	1	1	1	1	0	0	0	BW, 4 bit
	0	1	1	0	0	1	1	0	0	1	1	0	BW, 2 bit
	0	1	0	1	0	1	0	1	0	1	0	1	BW, 1 bit

Paritáscsoportok	Bit pozíciók	Bitek jelölései
0	1, 3, 5, 7, 9 ,11	C0, D0, D1, D3, D4, D6
1	2, 3, 6, 7, 10, 11	C1, D0, D2, D3, D5, D6
2	4, 5, 6, 7, 12	C2, D1, D2, D3, D7
3	8, 9, 10, 11, 12	C3, D4, D5, D6, D7

13	12	11	10	9	8	7	6	5	4	3	2	1	Oszlopszám
DEB	D7	D6	D5	D4	C3	D3	D2	D1	C2	D0	C1	C0	
_	0	1	0	1	_	1	1	0	_	0	_	_	Adatbitek
1	0	1	0	1	1	1	1	0	1	0	0	0	Hozzáadott

kódbitek. Tehát a helyes ábrázolása 01011100-nek a következő:
1010111101000.