Pannon Egyetem
Villamosmérnöki és Információs Tanszék

Számítógép Architektúrák II.
(MIVIB344ZV)

2. előadás: Számrendszerek,
Nem-numerikus információ ábrázolása

Előadó: Dr. Vörösházi Zsolt
voroshazi.zsolt@mik.uni-pannon.hu

Jegyzetek, segédanyagok:

- Könyvfejezetek:
  - http://www.virt.uni-pannon.hu → Oktatás → Tantárgyak → Számítógép Architektúrák II.
  - (chapter02.pdf)
- Fóliák, óravázlatok .ppt (.pdf)
- Feltöltésük folyamatosan

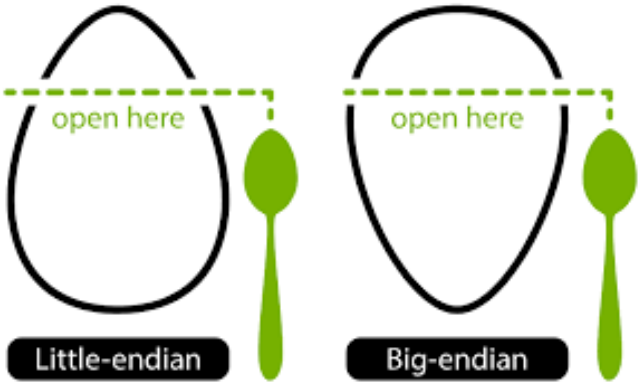
Információ ábrázolás:

- A) Számrendszerek (numerikus információ):
  - I.) **Egész típusú:**
    - előjel nélküli,
    - előjeles:
      - 1-es komplemens,
      - 2-es komplemens számrendszerek.
  - II.) **Fix-pontos,**
  - III.) **Lebegő-pontos** (IBM-32, DEC-32, IEEE-32),
    - Excess kód (exponens kódolására)
- B) Nem-numerikus információ kódolása
- C) Hiba-detektálás, és javítás (Hamming kód)
  - SEC-DED

A) Számrendszerek

Endianitás (endianness)

- A számítástechnikában, az **endianitás** („bit/byte-sorrend” a jó fordítása) az a tulajdonság, ami bizonyos adatok - többnyire kisebb adategységek egymást követő sorozata - tárolási és/vagy továbbítási sorrendjéről ad leírást (pl. két protokoll, vagy busz kommunikációja).
- Ez a tulajdonság döntő fontosságú az értékeknek a számítógép memóriájában bit/byte-onként való tárolása (egy memória címhez relatívan), ill. továbbítása esetében
- Két lehetséges sorrend:
  - Big-Endian (BE) formátum
  - Little-Endian (LE) formátum



📖 Háttér: Az eredeti angol kifejezés az *endianness* (1980) egy utalás arra a háborúra, amely a két szembenálló csoport között zajlik, akik közül az egyik szerint a lágytojás nagyobb/vastagabb végét (big-endian), míg a másik csoport szerint a lágytojás kisebb végét (little-endian) kell feltörni. Erről Swift ír a *Gulliver Kalandos Utazásai* című könyvében ☺

„Kicsi a végén” - Little-endian (LE)

Példa: 32-bites „`3A 4B 1C 2D`” értéket a `0x100`… címtől növekvő módon, **4×1byte**-os tárolási egységekben tároljuk:

`0x100 0x101 0x102 0x103 ...`

- Ekkor a kevésbé jellemző ("legkisebb") byte (az angol Least Significant Byte rövidítéséből *LSB* néven ismert) az első, ez a `2D`, tehát a *kis vége kerül „előre”, azaz a legkisebb címen van tárolva (`0x100`):*

0x103	**0x102**	**0x101**	**0x100**	*memóriacímek*	**[31:0]**
3A	**4B**	**1C**	**2D**		
MSB	←		**LSB**	*adategységek*	

- **LE = Hagyományos, általánosan** használt formátum: ha mást nem mondunk, nem kötik ki külön, ezt feltételezzük!
- pl. Intel, AMD, illetve ARM processzorok stb.

„Nagy a végén” - Big-endian (BE)

Példa: 32-bites értéket „`3A 4B 1C 2D`”, a `0x100` címtől kezdve tároljuk a memóriában, **4×1 byte-os** tárolási egységekben :

- 1 byte-onként növekvő címekkel rendelkezik

`0x100 0x101 0x102 0x103 ... ”2D 1C 4B 3A”...`

memóriacímek

0x103	**0x102**	**0x101**	**0x100**		**[0:31]**
2D	**1C**	**4B**	**3A**		
LSB	→	→	**MSB**	*adategységek*	

- Ekkor a „legjellemzőbb” byte - „Most Significant Byte" (*MSB)*, ami itt a „`3A`” - a memóriában az *legalacsonyabb címen van tárolva (`0x100`)*, míg a következő "jellemző byte" (`4B`) az egyel nagyobb címen van tárolva, és így tovább.
- *Bit/byte-reversed format! (fordított formátum)*
- PI: speciális, beágyazott rendszerek processzorai, pl. MCU – mikrovezérlők, programozható FPGA-k (MicroBlaze, PowerPC), stb

I.) Egész típusú számrendszerek

Bináris (*p=2*) számrendszer: ”1” / ”0” (I / H, T / F, …)

- **Általános szabály:**

  N biten → 2^N lehetséges érték ábrázolható!

- Példa: N = 2 byte-os (16 bites), vagy N = 4 byte-os (32-bites) bináris, **pozitív**, egész szám esetén:
  - 2^{16}-1 = 65535 vagy
  - 2^{32}-1 = 4 294 967 295 a maximálisan ábrázolható érték.

- **Negatív alak = Előjel kezelése**? legegyszerűbb mód, ha a szám legfelső helyiértékű (MSB) bitjét ***előjelbitnek*** (**S**) tekintjük:
  „+”: 0, „–”: 1 (de nem feltétlenül ez az általános!)
  - Ennek oka: célszerű olyan ábrázolási módot alkalmazni, ahol **a kivonás (-) művelete → összeadással (+) helyettesíthető.**

a.) előjel nélküli egész:

- Unsigned integer:

$$V_{\text{UNSIGNED INTEGER}} = \sum_{i=0}^{N-1} b_i \times 2^i$$

- 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!

- **<u>Pl:</u> (Legyen N:=6, LE, unsigned int)**

$10\ 1101_2 \Rightarrow$1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 45_{10}

b.) 1’s komplemens rendszer

- V értékű, N bites rendszer: **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!! (-0 / +0, ellenőrzés szükséges)

Példa:
1’s komplemens

Példa: **N:=6, LE**

$V = 01\ 0010_{2)} = 18_{10)}$

a.) Adja meg a fenti V szám 1’s komplemensét!

V(1’s)= $10\ 1101_{2)}$

b.) Milyen decimális, negatív, egész értéket ábrázol a fenti bináris 1’s komplemens szám?

V(1’s)= $10\ 1101_{2)} = ?$

$= -18_{10)}$

V érték	V(Unsigned int)	V(1's comp)
01 1111	31	31
01 1110	30	30
…	…	…
010 010	18	18
…	…	…
00 0010	2	2
00 0001	1	1
00 0000	0	+0
11 1111	63	-0
11 1110	62	-1
11 1101	61	-2
…	…	…
10 1101	45	-18
…	…	…
100001	33	-30
100000	32	-31

c.) 2's komplemens rendszer

- V jelöli a szám értékét, N bites, LE rendszer:

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

- Értékhatár: $-2^{(N-1)}$ –től … $+2^{(N-1)}-1$ –ig
  - ha MSB='1', a szám negatív
  - helyiértékes rendszer!

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.

Példa: 2’s komplemens

Legyen előjeles – 2’s komplemens rendszer, N:=6, LE

Milyen decimális, negatív, egész értéket ábrázol az alábbi bináris 2’s komplemens szám?

- **V(2’s) =** $10\ 1101_{2)}$ **= ?**

$$-1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$$
$$= -32 + 8 + 4 + 1 = -19_{10)}$$

- További számítási módszerek:

* azt jelöli, amikor az adott helyiértéken ‘1’-et kell kivonni még az Xi értékéből (borrow from Xi)

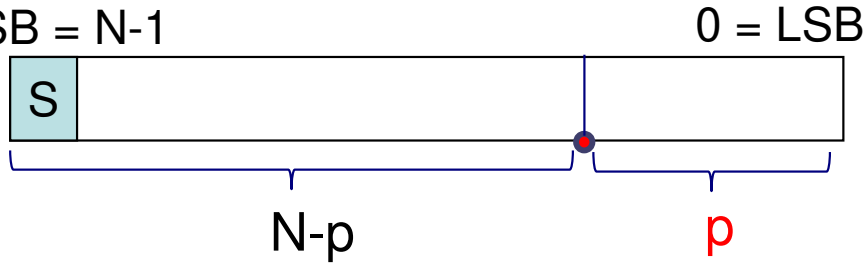
I.

	010011	a szám
	101100	szám 1’s kompl.
+	1	+1
	101101	–19 **=V(2’s)**

II.

	* * * * * *	*
X	~~1~~0~~0~~0~~0~~0~~0~~	~~6~~4
–Y	– 010011	– 19
Z	–19 ≡ 101101	45

II.) Fix-pontos számrendszerek



- Műveletek:
  - +, - : mint egész számrendszereknél
  - *, / : vizsgálni kell a tizedespont helyét
- V jelöli az előjeles, bináris, fixpontos szám értékét (LE):

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

- Paraméterek:
  - N: fixpontos szám hossza (egész + törtrész együttesen)
  - **p: radix (tizedes) pont** helye, törtrész hossza
  - ***differencia***, $\Delta r = 2^{-p}$ (számrendszer finomsága, azaz két szomszédos szám távolsága)
    - *Ha p=0 → Δr=1 → egész számrendszer* *pl: -1, 0, 1, 2, 3, …*
    - *Ha p=1 → Δr=1/2 → fixpontos számrendszer* *pl: -1/2, 0, 1/2, 1, 3/2, 2, …*
    - *Ha p=2 → Δr=1/4 → fixpontos számrendszer* *pl: -1/4, 0, 1/4, 1/2, 3/4, 1, …*

Példa: Fixpontos rendszer

Legyen egy N:=16 bites, LE, bináris, **2's komp. fixpontos rdsz**. ahol p:=8.

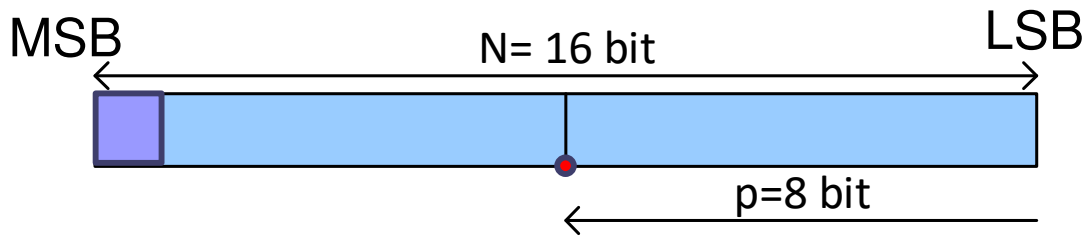
Kérdések:

- Ábrázoljuk az N, p paramétereknek megfelelő számformátumot
- V(legkisebb pozitív)=?
- V(legnagyobb pozitív)=?
- V(legkisebb negatív)=? //"leg-negatívabb"
- V(legnagyobb negatív)=? //"legkevésbé negatív"
- V(zéró),
- $\Delta r = ?$

Minden paramétert számoljunk ki, és ahol lehet decimális értékben / hatványalakban is megadva!

Megoldás: Fixpontos rendszer

- Számformátum:



- Differencia $\Delta r = 2^{-8} = 1/256 = (0{,}390625 * 10^{-2})$
- V(zero)= **0**0000000.00000000 = 0.0
- V(legkisebb poz)= **0**0000000.00000001 = $2^{-8} = 1/256 = \Delta r = (0{,}390625 * 10^{-2})$
- V(legnagyobb poz)= **0**1111111.11111111 ≈ $128 = 128 - \Delta r$
- V(legkisebb negatív) = **1**0000000.00000000 = $-2^{7} = -128$
- V(legnagyobb negatív)= **1**1111111.11111111 = $-2^{-8} = -1/256 = = -\Delta r = (-0{,}390625 * 10^{-2})$

III.) Lebegőpontos számrendszerek

- Nagyobb pontosság vs. több paraméter tárolása → nehezebb számolni
- 7 különböző paraméter: *a számrendszer alapja, előjele és nagysága, a mantissza alapja, előjele és hosszúsága, ill. a kitevő alapja*.
- **Egyszerű matematikai jelölés:**

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

- Fixpontosnál nagyságrendekkel kisebb, vagy nagyobb számok ábrázolására is lehetőség van:
  - Pl: Avogadro-szám: 6.022*10^23
  - Pl: proton tömege 1.673*10^-24 g
  - Normalizált lebegőpontos rendszerek: pl. DEC-32, IBM-32, IEEE-32
    - **32-bites, vagy egyszeres pontosságú – float (C)** vagy
    - 64-bites, vagy dupla pontosságú – double (C)

Normalizálás (mantissza)

Példa: adott decimális (r_b = 10-es) alapú lebegőpontos szám **normalizálása** esetén:

- $\mathbf{32\ 768_{10}} = \mathbf{0.32768 \times 10^5} =$
  $3.2768 \times 10^4 = 32.768 \times 10^3 =$
  $327.68 \times 10^2 = 3267.8 \times 10^1$

(mindegyik érvényes alak) DE

- $[\frac{1}{r_b}, 1[$ közé **normalizált** alak:

  $\mathbf{0.32768 \times 10^5}$

  Mantissza igazítása ↔ exponens változása!

Lebegőpontos rendszer (FPN) jellemző paraméterei

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

Excess-kód

Miért használják az Excess „kódolást” ?

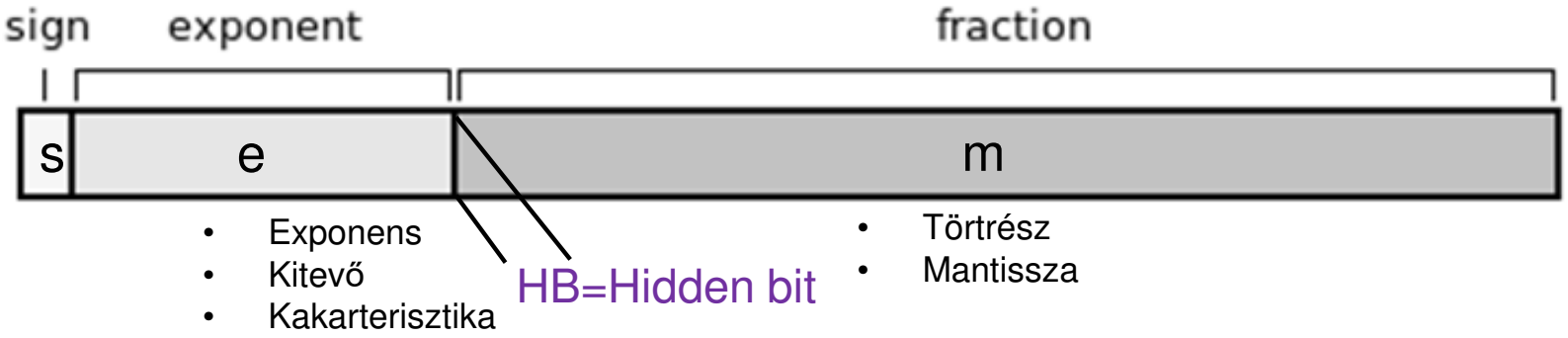
- **Lebegőpontos** számok ***kitevőjét*** **(*exponens*-ét) tárolják** / kódolják e módszerrel. Cél: a kitevő **NE legyen negatív**, ezért eltolással oldják meg, hogy a negatív kitevőhöz egy pozitív számot adnak hozzá.
- Megjegyzés: egy lebegőpontos szám előjelbitje, a mantissza előjelét, és nem pedig az exponens előjelét tárolja!
  - V: az exponens valódi értéke, (lehet pozitív, negatív)
  - E: az excess (offset/eltolás értéke)
  - S: a reprezentálni kívánt érték, azaz a kitevő kódolt értéke, melyet eltárolunk (ez csak pozitív lehet)
  - S = V + E

Lebegőpontos számrendszerek

- **DEC-32:** r_b=**2**, r_e=2, m=24 (nincs HB), p=24 (m=p), e=8, az exponenst Excess-128 kódolással tárolja. Normalizálás: $\left[\frac{1}{r_b}, 1\right[$
- **IBM-32:** r_b=**16**, r_e=2, m=p=6 (nincs HB), e=7, az exponenst Excess-64 kódolással tárolja. Normalizálás: $\left[\frac{1}{r_b}, 1\right[$
- **IEEE-32:** r_b=**2**, r_e=2, m=24, de p=23! **(HB='1')**, e=8, az exponenst Excess-127 kódolással tárolja. Normalizálás: $[HB, r_b[= [1, 2[$

IEEE 754-1985 számformátum

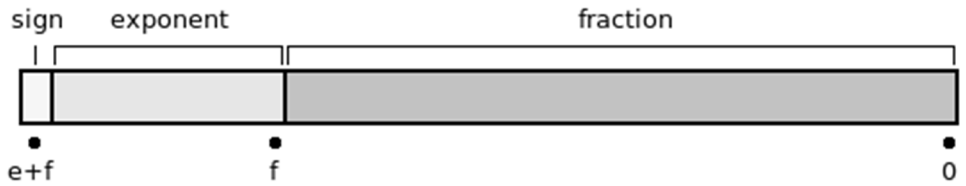
- **Nemzetközileg elfogadott szabvány a bináris lebegőpontos számok tárolására**, amely tartalmazza:
  - negatív zérust is: –0 = 1_000..0000_00…0000 (két zérus: +0)
  - normalizálatlan (denormált) számok (Hidden-Bit = 0)
  - NaN: nem szám (pl. $\frac{\pm 0}{\pm 0} = \mathrm{NaN};$ vagy $\pm 0 \times \pm\infty = \mathrm{NaN}$)
  - $\pm\infty$



Sign-magnitude format („előjel-hossz” formátum): az előjel (sign) külön biten van el tárolva (MSB), az exponens kódolt (excess-el „eltolt”), ill. a törtrész (mantissza) következik végül. Fontos a sorrendjük!

IEEE-32 bites normalizált lebegőpontos rendszer

- $V_{E\,[min,max]} = [-126, 127] \rightarrow [1, 254]$ az Excess127-el eltolt exponens tartomány
- Speciális jelentőség:
  - $V_E = 0$ értékénél (zérus ábrázolása)
  - $V_E = [255]$ értékénél lehetőség van bizonyos információk tárolására




```
(+∞) + (+7) = (+∞)
(+∞) × (−2) = (−∞)
(+∞) × 0 = NaN
0 / 0     = NaN
Sqrt(−1) = NaN
```

V_E [255]	S (előjel)	Ábrázolás jelentése
≠ 0	X	Nem egy szám (NaN)
0	0	+∞
0	1	-∞

Különböző pontosságú **IEEE** lebegőpontos rendszerek

Típus IEEE	Sign (s)	Exponens (e)	Excess-kód (Exc)	Mantissza (p < m)	Teljes szóhossz
Half (IEEE 754r)	1	5	15	10	16
Single	**1**	**8**	**127**	**23**	**32**
Double	1	11	1023	52	64
Quad	1	15	16383	112	128

Példa: DEC-32

Ábrázoljuk DEC-32 rendszerben a következő decimális számot: $-12.625_{10} = ?$

DEC-32 paraméterei: r_b=**2**, r_e=2, m=p=24, (nincs HB), e=8, Excess-128

$-12.625_{10} = 1100.101_2 =$ (normalizálás $[\frac{1}{r_b}, 1[$) $\Rightarrow$

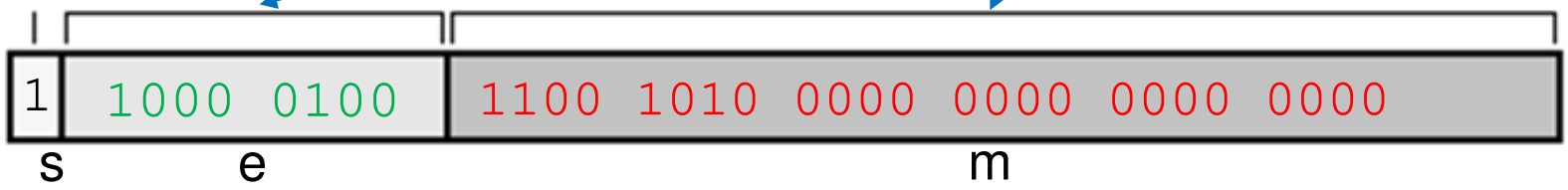
$= \mathbf{0}.1100101_2 \times 2^4$

Kitevő: $4_{10} \Rightarrow 100_2$ + Exc-128 =

$= 100 + 10000000 =$

$1000\ 0100_2$

S=1



Példa: IBM-32

Ábrázoljuk IBM-32 rendszerben a következő decimális számot: $-12.625_{10} = ?$

IBM-32 paraméterei: r_b=16, r_e=2, (m=p=6), (nincs HB), e=7, Excess-64

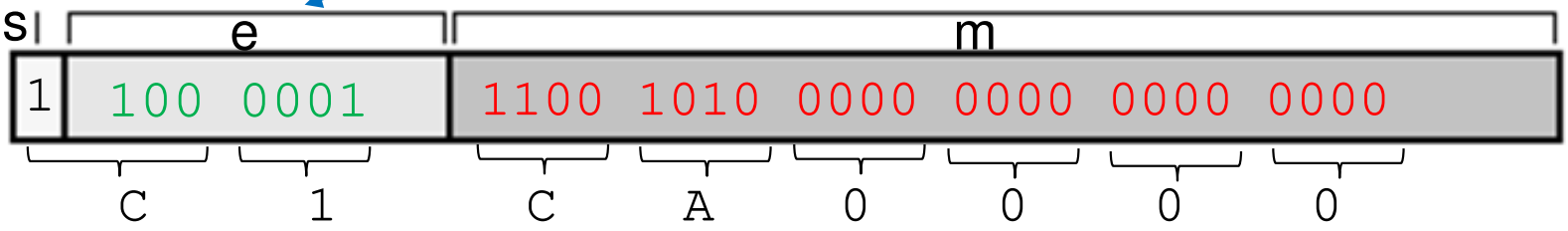
$-12.625_{10} = C.A_{16} =$ (normalizálás $[\frac{1}{r_b}, 1[) \Rightarrow$ $0.CA_{16} \times 16^1$

$12 = C_{16}$, $0.625 = \frac{10}{16} = \frac{A}{16} = 0.A_{16}$

Kitevő: $1_{10} \Rightarrow 1_2$ + Exc-64 =

= 1+100 0000 = $100\ 0001_2$

S=1



Példa: IEEE-32

Ábrázoljuk IEEE-32 rendszerben a következő decimális számot: $-12.625_{10} = ?$

IEEE-32 paraméterei: r_b=**2**, r_e=2, (m=24>p=23), (HB=1!), e=8, Excess-127

$-12.625_{10} = 1100.101_2 =$ (normalizálás [1,2[) $\Rightarrow$

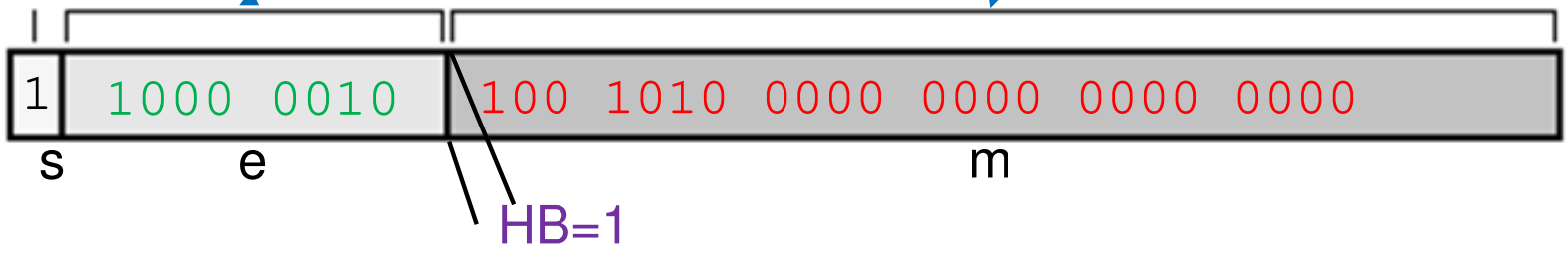
$= \mathbf{1}.100101_2 \times 2^3$

Kitevő: $3_{10} \Rightarrow 11_2$ + Exc-127 =

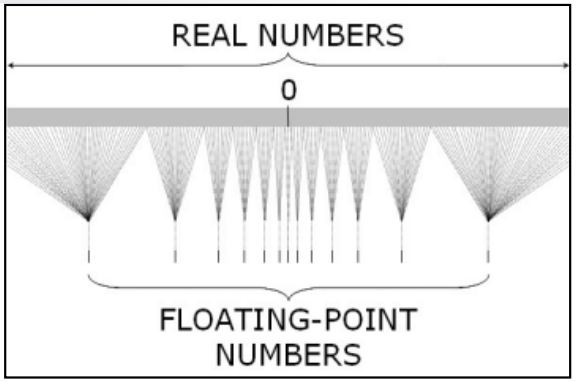
= 11+111 1111 =

$1000\ 0010_2$

S=1



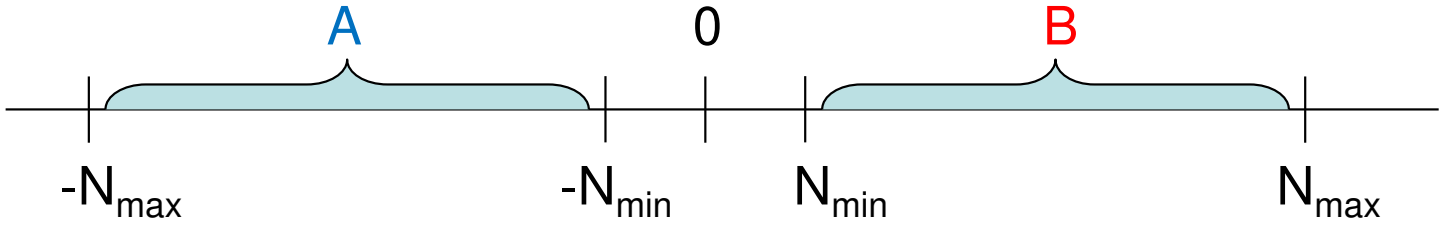
Lebegőpontos rendszer dinamika tartománya



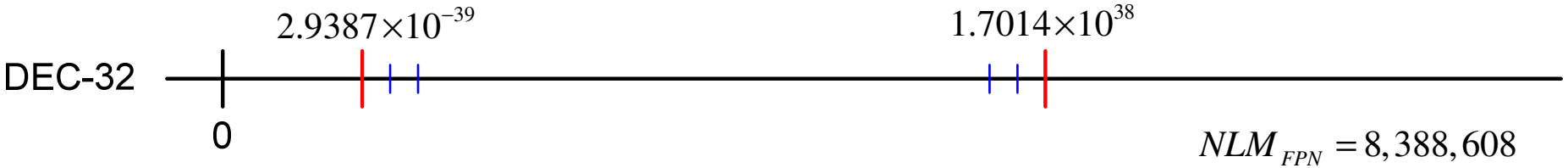
A lebegőpontos számábrázolás **A** és **B** számtartományt tudja (részben) ábrázolni.

- Ha $x < -N_{max}$ vagy $x > N_{max}$, akkor

  túlcsordulásról,
- Ha $-N_{min} < x < N_{min}$, akkor **alulcsordulásról** beszélünk, és ekkor x nem ábrázolható.
- Az ilyen esetek kezelése a programozó feladata.

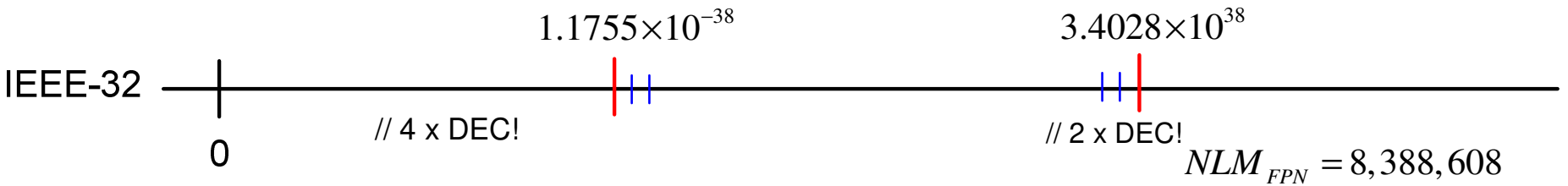


Lebegőpontos számrendszerek összehasonlítása (ha FPN előjele pozitív):



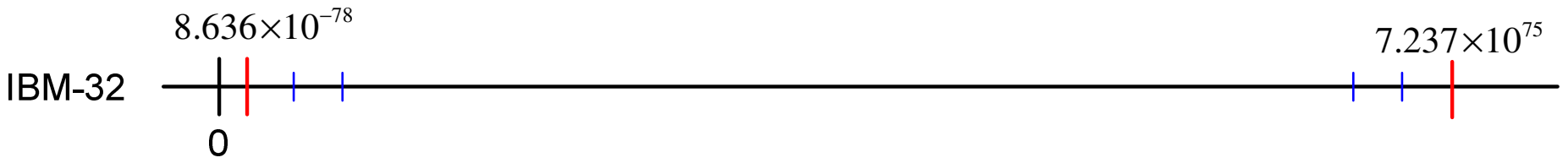
$NLM_{FPN} = 8,388,608$

$NRV_{FPN} = 2.139\times10^{9}$



$NLM_{FPN} = 8,388,608$

$NRV_{FPN} = 2.139\times10^{9}$



$NLM_{FPN} = 15,728,640$

$NRV_{FPN} = 1.9975\times10^{9}$

7%-al kevesebb mint a DEC!!

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

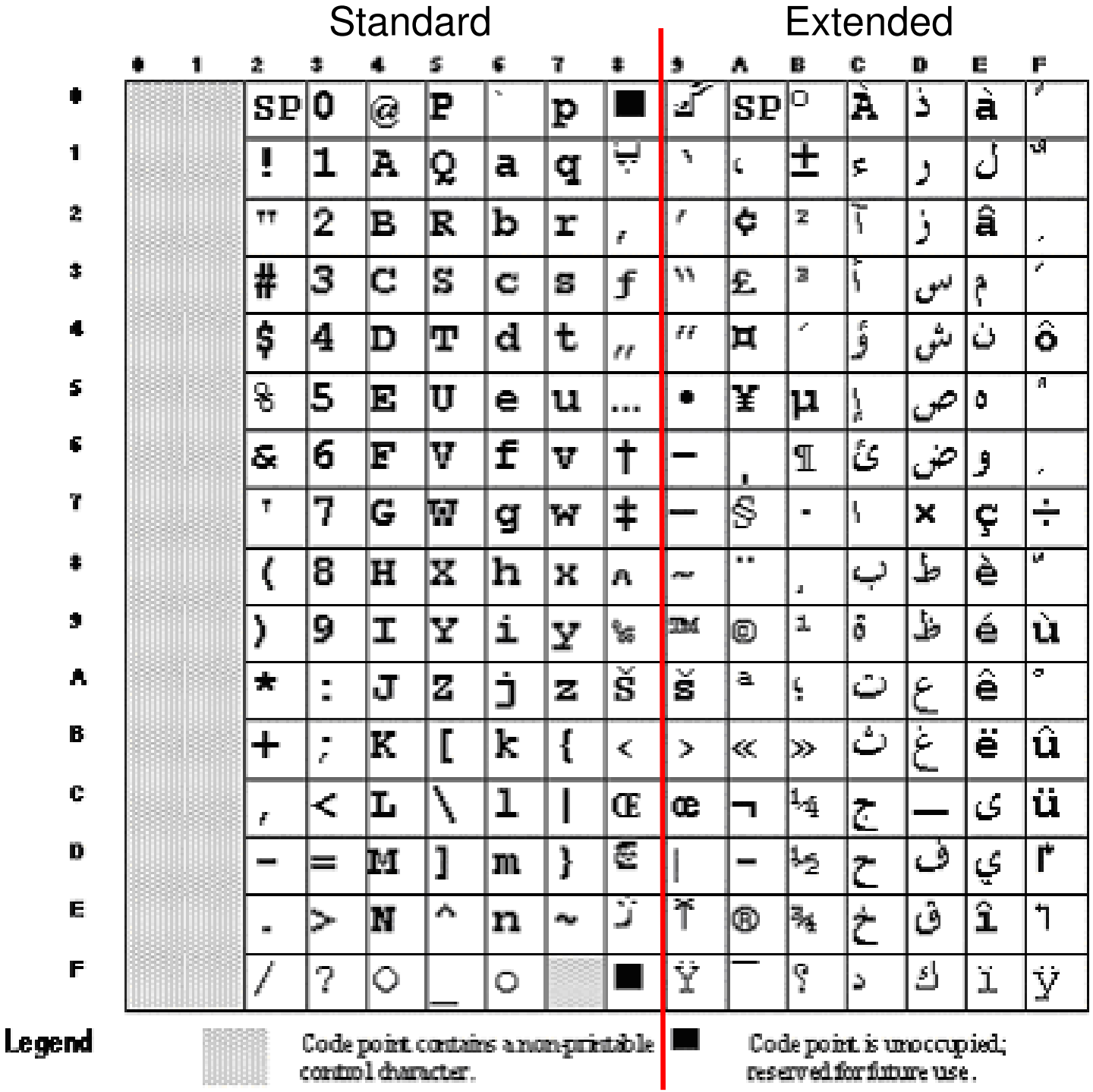
- BCD (Binary Coded Decimal): 6-biten
  - nagybetűk, számok, és speciális karakterek
- EBCDIC (Extended Binary Coded Decimal Interchange Code): 8-biten (A. Függelék)
  - + kisbetűs karaktereket és kiegészítő-információkat
  - 256 értékből nincs mindegyik kihasználva
  - Továbbá I és R betűknél szakadás 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	(NSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~	£	A 000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000			000	K LK020000	S LS000000	2 ND020000
-3	ã LA170000	ë LE170000	Ã LA180000	Ë LE180000	c LC010000	l LL010000			000	L LL020000	T LT000000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000			000	M LM020000	U LU000000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	' LV010000	≥ SM340000	⌐ LE020000	N LN020000	V LV000000	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 LO830000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	œ LO810000	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	$ SC030000	, SP060000	# SM010000	» SP180000	º SM220000	¿ SP160000	š LS210000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD830000	æ LA510000	Ð LD840000	¯ SD310000	õ LO170000	ü LU170000	Õ LO180000	Ü LU180000
-D	(SP080000	) SP070000	_ SP090000	' SP050000	ý LY110000	ž LZ210000	[SM080000	] SM090000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT810000	Æ LA520000	Þ LT840000	Ž LZ220000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	^ SD150000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

A
LA020000

Extended ASCII (1 byte)



Unicode

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

Általános írásjelek, matematikai, pénzügyi, mértani szimbólumok stb.

*részlet

C) Hamming hibakódolás – Hiba-detektálás, és javítás

Hibakódolás - Hibadetektálás és Javítás

- **N bit segítségével 2^N különböző érték, cím, vagy utasítás ábrázolható**
- 1 bittel növelve (N+1) bit esetén: $2^{\wedge N}$ -ről $2^{\wedge N+1}$ –re: tehát megduplázódik az ábrázolható értékek tartománya
- ***Redundancia:*** „többlet bitek” segítségével lehet a hibákat detektálni, ill. akár javítani is!
  - Redundáns többlet bitek a paritás, v. kódbitek.

Paritás bit

- Legegyszerűbb hibafelismerési eljárás, a paritásbit átvitele. Két lehetőség:

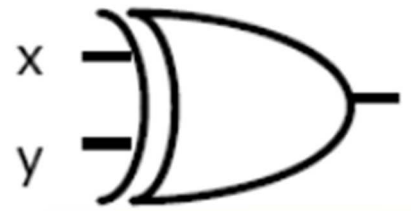
	Adatbitek	Paritásbit(kódbit)
páros paritás	1 1 0 1	1
páratlan paritás	1 1 0 1	0

- Páros paritás: az ‘1’-esek száma páros.
  - Az adatszóban lévő ‘1’-esek számát ‘1’ vagy ‘0’ hozzáadásával párossá egészítjük ki.
  - ‘0’ a paritásbit, ha az ‘1’-esek száma páros volt.
- Páratlan paritás: az ‘1’-esek száma páratlan.
  - A adatszóban lévő ‘1’-esek számát ‘1’ vagy ‘0’ hozzáadásával páratlanná egészítjük ki.
  - ‘1’ a paritásbit, ha az ‘1’-esek száma páros volt.

Paritás bit generáló áramkör

- Paritásbit képzése:
  - ANTIVALENCIA (XOR) művelet alkalmazása a kódszó bitjeire, pl. ‘n’ adatbit esetén „n-1”-szer!
- Példa:

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0



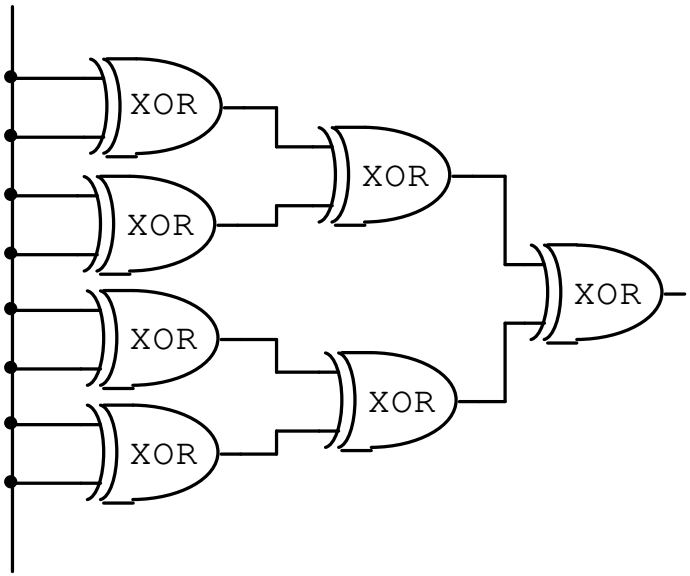
Kódszó Paritásbit(P)

0 0 0 1 ?→ 0 ⊕0 ⊕0 ⊕1 = 1
0 1 1 0 ?→ 0 ⊕1 ⊕1 ⊕0 = 0
1 1 1 0 ?→ 1 ⊕1 ⊕1 ⊕0 = 1

Páros paritás!

Paritás bit ellenőrzés

- Páros v. páratlan paritás: N bites információ egy kiegészítő bittel bővül → *egyszeres hiba* felismerése
- Hibák lehetséges okai:
  - leragadásból: ‘0’-ból ‘1’-es lesz, vagy fordítva
  - ideiglenes, tranziens jellegű hiba
  - áthallás (crosstalk)
- 8-adatbithez *páros* paritásbit generálás (IC 74‘180. http://alldatasheet.com 9 bites paritás ellenőrző)
  (XOR gate IC 74’86)

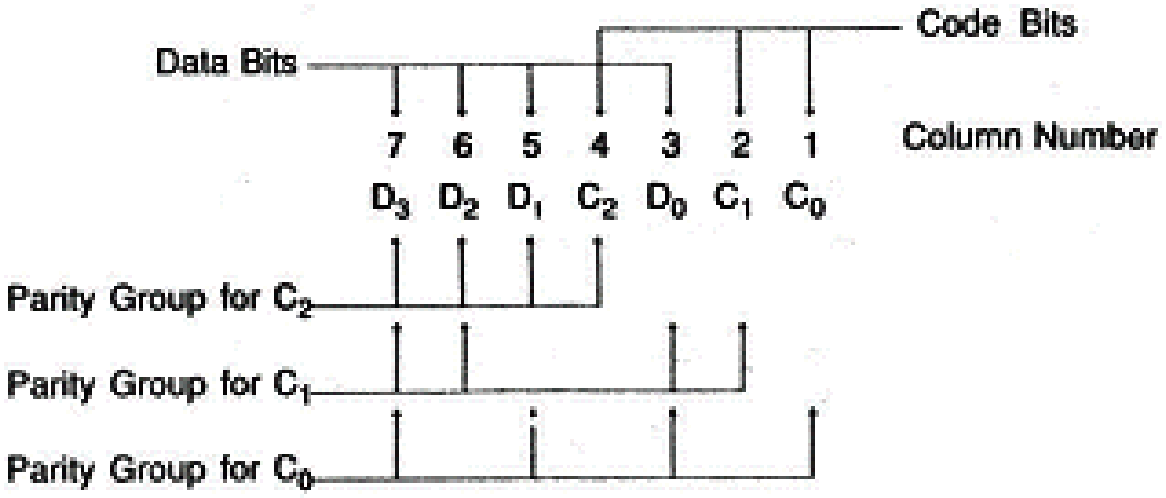


Hamming kód

- Háttér: több *redundáns* bittel nemcsak a hiba meglétét, és helyét tudjuk detektálni, hanem akár a hibás bitet javítani is tudjuk
- **Hamming kód**: egy biten tároljuk a bitmintázatok azonos helyiértékű bitjeinek különbségét, tehát egybites hibát lehet vele *javítani*.
  - Előny:
    - Bitcsoportokon történő paritás ellenőrzésen alapul: gyors művelet
    - Egy-szeres hibát tud javítani (adatokon)
    - SEC-DED: Single Error Correction / Double Error Detection
  - Hátrány:
    - Csoportos hibát nem képes javítani, ill. több egy-bites hibát sem
    - Adatbitekhez képest „túl sok” a javító ún. kód bitek száma

7-bites Hamming kódú kódszó konstruálása (pl. 4 –bites adatszóra)

- **$2^{\wedge N}$-1 bites Hamming kód: N kódbit → $2^{\wedge N}$-N-1 adatbit**
- Összesen pl. 7 biten ***4*** *adatbitet (D0,D1,D2,D3),* ***3*** *kódbittel (C0,C1,C2)* kódolunk (LE!)
- 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

Pl. 1/a) Hamming kódú hibajavító áramkör tervezése (Little Endian)

- Példa: bemeneti adatbit-mintázatunk 0101 (D_3-D_0). LE!
- 4 adatbit → 3 kódbitünk van
  - Alkalmazzunk páratlan paritást! A megfelelő helyen szereplő kódbitekkel kiegészítve a következő szót kapjuk: 010**0**1**10**. Ha nincs hiba, a paritásellenőrzők (C2, C1, C0) kimenete '000' (nem-létező bitpozíciót azonosít, azaz nincs hiba), így megegyezik a kódolt mintázat paritásbitjeinek értékével, minden egyes paritáscsoportra (küldött és az azonosított Ci-minták bitenkénti XOR kapcsolata).
  - Hiba szindróma: Hiba esetén például, tfh. az input mintázat 010**0**10 -re változik, ekkor a vevő oldali paritásellenőrző hibát észlel. Ugyan C2. paritásbitcsoport rendben ('0'), DE a C1 ('0') és C0 ('1') változott, tehát hiba van:
    - Ekkor 011 = 3 az azonosított minta, ami a 3. oszlopot jelenti (→ **D0** helyén).
    - Javításként *invertálni kell* a 3. bitpozícióban lévő bitet. 010**0**10 ⇒ 010**0**1**10**. Ekkor a kódbitek a következőképpen módosulnak a páratlan paritásnak megfelelően: C2=0, C1=1 és C0=0.

Pl. 1/a) folyt. Hamming kódú kódszó (Little Endian)

- 4 adatbithez (D_3-D_0)=0101 → 3 paritásbit (C_2-C_0)
  - azaz 7-bites Hamming kódú hibajavító kódszó

7	6	5	4	3	2	1
D3	D2	D1	C2	D0	C1	C0
0	1	0	?	1	?	?
0	1	0	0	1	1	0
|	.	|	.	|	.	|
|	|	.	.	|	|	
|	|	|	|			

poz

Error syndrome

	C2	C1	C0
Adó	0	1	0
Vevő	0	1	0
XOR	0	0	0

Azaz 000 = 0. pozíció (nem létezik, tehát nincsen hiba)

Pl. 1/a) folyt. Hamming kódú kódszó (Little Endian)

- Tfh. van **hiba**, a D0 megváltozik ’1’ → ’0’

7	6	5	4	3 *	2	1	poz
D3	D2	D1	C2	D0	C1	C0	
0	1	0	?	0	?	?	
0	1	0	0	0	0	1	
|	.	|	.	|	.	|	
|	|	.	.	|	|	
|	|	|	|			

Error syndrome

	C2	C1	C0
Adó	0	1	0
Vevő	0	0	1
XOR	0	1	1

Azaz 011 = 3. pozíció (tehát D0 a hibás!)

Javítás = hibás D0 invertálása ’0’ → ’1’

Pl. 1/b) Hamming kódú hibajavító áramkör tervezése (Big Endian)

- Példa: bemeneti adatbit-mintázatunk 0101 (D_0-D_3). BE!
- 4 adatbit → 3 kódbitünk van
  - Alkalmazzunk páratlan paritást! A megfelelő helyen szereplő kódbitekkel kiegészítve a következő szót kapjuk: **10**0**1**101. Ha nincs hiba, a paritásellenőrzők (C0, C1, C2) kimenete '000' (nem-létező bitpozíciót azonosít, azaz nincs hiba), így megegyezik a kódolt mintázat paritásbitjeinek értékével, minden egyes paritáscsoportra (azaz a küldött és azonosított Ci-minták bitenkénti XOR kapcsolata).
  - Hiba szindróma: Hiba esetén például, tfh. az input mintázat 1011101 -re változik, ekkor a vevő oldali paritásellenőrző hibát észlel. Ugyan C2. paritásbitcsoport rendben ('1'), DE a C1 ('1') és C0 ('0') változott, tehát hiba van:
    - Ekkor (110) azaz 011 = 3 az azonosított minta, ami a 3. oszlopot jelenti (→ **D0** helyén).
    - Javításként *invertálni kell* a 3. bitpozícióban lévő bitet. 1011101 ⇒ 1001101. Ekkor a kódbitek következőképpen módosulnak a páratlan paritásnak megfelelően: C0=1, C1=0 és C2=1.

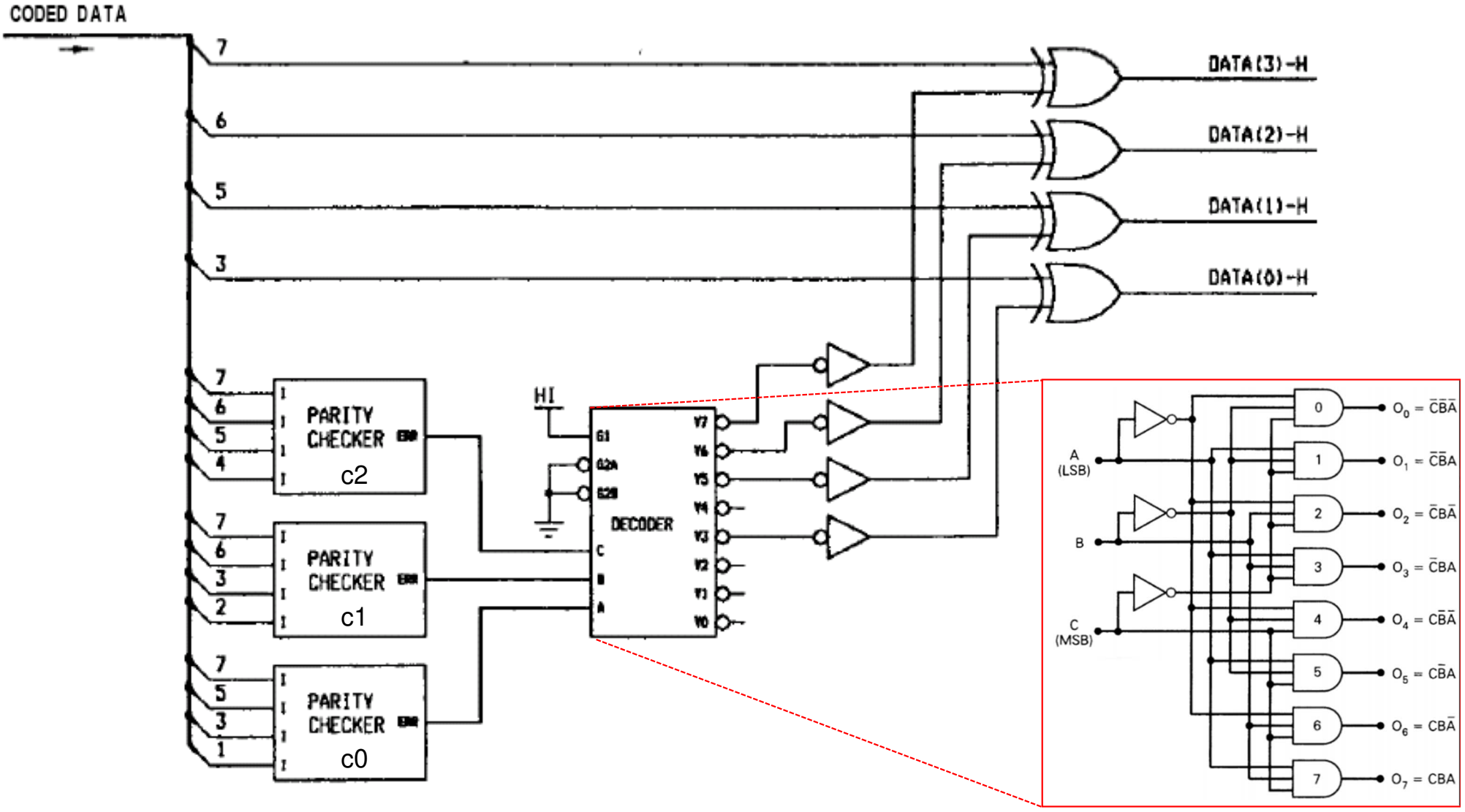
Pl 2.) Hamming kódú hibajavító áramkör tervezése (Little Endian)

- Példa: bemeneti adatbit-mintázatunk 0101 (D_3-D_0). LE!
- 4 adatbit → 3 kódbitünk van
  - Alkalmazzunk **páros paritást**! A megfelelő helyen szereplő kódbitekkel kiegészítve a következő szót kapjuk: 0101101. Ha nincs hiba, a paritásellenőrzők (C2, C1, C0) kimenete '000' (nem-létező bitpozíciót azonosít). Így megegyezik a kódolt mintázat paritásbitjeinek értékével, minden egyes paritáscsoportra (küldött és vett Ci-k bitenkénti XOR kapcsolata).
  - Hiba esetén például, ha az input mintázat 0111101 -re változik, ekkor a paritásellenőrző hibát észlel. Két paritásbit ellenőrző megváltozott: C2 ('0'), a C0 ('0'), de C1 ('0') változatlan.
    - Ekkor 101 = 5, az azonosított minta, ami a 5. oszlopot jelenti (→ **D1** helyén).
  - Javításként *invertálni kell* a 5. bitpozícióban lévő bitet. 0111101 ⇒ 0101101. Ekkor a kódbitek a következőképpen módosulnak a páratlan paritásnak megfelelően: C2=1, C1=0 és C0=1.

Pl 3.) Hamming kódú hibajavító áramkör tervezése (Little Endian)

- Példa: bemeneti adatbit-mintázatunk 0101 (D_3-D_0). LE!
- 4 adatbit → 3 kódbitünk van
  - Alkalmazzunk páratlan paritást! A megfelelő helyen szereplő kódbitekkel kiegészítve a következő szót kapjuk: 010**0**1**10**. Ha nincs hiba, a paritásellenőrzők (C2, C1, C0) kimenete '000' (nem-létező bitpozíciót azonosít, azaz nincs hiba), így megegyezik a kódolt mintázat paritásbitjeinek értékével, minden egyes paritáscsoportra (küldött és vett Ci-k bitenkénti XOR kapcsolata).
  - Hiba esetén például, tfh. az input mintázat 0100100 -re változik, akkor a paritásellenőrző hibát észlel. C2. paritásbit ellenőrző változatlan ('0'), és a C0 is ('0'), DE a C1 ('0') hibát észlel, tehát:
    - Ekkor 010 = 2 az azonosított minta önmaga, ami a 2. oszlopot jelenti (→ **C1** paritásbit! helyén).
  - Javításként itt már a dupla hibaellenőrzést (DEB / vagy SECDEC) kell alkalmazni, amely a **paritásbiteket is kódolja.**

7-bites Hamming kódú hibajavító áramkör felépítése



Példa: SEC-DED-dupla paritáshiba ellenőrzés (Little Endian)

DEB: extra bit, a teljes kódszóra vonatkozóan (Ci-ket is kódolja)

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.