Számítógép architektúrák

- **Számítógépek felépítése**
- **Digitális adatábrázolás**
- **Digitális logikai szint**
- **Mikroarchitektúra szint**
- **Gépi utasítás szint**
- **Operációs rendszer szint**
- **Assembly nyelvi szint**
- **Probléma orientált (magas szintű) nyelvi szint**
- **Perifériák**

Ajánlott irodalom

- **http://it.inf.unideb.hu/~halasz**
- **S. Tanenbaum: *Structured computer organization* (Prentice Hall, 2006) (T). Magyarul: Számítógép-architektúrák 2. átdolgozott, bővített kiadás (Panem 2006).**
- Patterson D.A., Henessy J.L.: Computer organization & Design, Morgan Kaufmannn Publ. (2 ed.) 1998.
- Rob Williams: Computer System Architecture (A Networking Approach), Addison Wesley, 2001.
- Sima D., Fountain T. , Kacsuk, P.: Korszerű számítógép architektúrák tervezési tér megközelítésben, Szak Kiadó, 1998.
- **Randall Hyde: The Art of Assembler Language, Randall Hyde, 2003**.
- Osborne: 80386/80286 Assembly Language Programming, Mc Graw-Hill, 1986.

Fixpontos számok

Pl.: előjeles kétjegyű decimális számok :

- Ábrázolási tartomány: [-99, +99].
- Pontosság (két „szomszédos” szám különbsége): 1.
- Maximális hiba: (az ábrázolási tartományba eső) tetszőleges valós szám, és a hozzá legközelebb lévő ábrázolható szám különbsége: 1/2.

Számolási pontatlanságok:

a = 70, b = 40, c = - 30 esetén

a + (b + c) = 80, (a+b) + c = -20.

túlcsordulás

Helyértékes ábrázolás

Pl.: $521{,}25_{10} = 5 * 10^2 + 2 * 10^1 + 1 * 10^0 +$

$2 * 10^{-1} + 5 * 10^{-2}$.

Általában (q alapú számrendszer esetén):

$$a_n\, a_{n-1} \ldots a_0, b_1 b_2 \ldots b_m =$$

$$a_n * q^n + a_{n-1} * q^{n-1} + \ldots + a_0 + b_1 * q^{-1} + b_2 * q^{-2} + \ldots + b_m * q^{-m}$$

$$0 \leq a_i, b_j < q$$

Átszámolás számrendszerek között

B: Bináris, O: Oktális, D: Decimális H: Hexadecimális

B	O	D	H	B	O	D	H
0	0	0	0	1000	10	8	8
1	1	1	1	1001	11	9	9
10	2	2	2	1010	12	10	A
11	3	3	3	1011	13	11	B
100	4	4	4	1100	14	12	C
101	5	5	5	1101	15	13	D
110	6	6	6	1110	16	14	E
111	7	7	7	1111	17	15	F

A.3. ábra része

Pl. $23{,}375_{10}$ átszámítása kettes számrendszer-be.

Egész rész osztással: Tört rész szorzással:

/2	marad	egész	*2
23	1		0.375
11	1	0	.750
5	1	1	.500
2	0	1	.000
1	1		

10111_2 $0{,}011_2$

$$23{,}375_{10} = 10111{,}011_2.$$

Véges tizedes tört nem biztos, hogy binárisan is véges!

Példa bináris összeadásra:

1. összeadandó: 0 1 0 1 1 0 1 0 (= 90_{10})
2. összeadandó: 0 1 1 1 1 1 0 0 (=124_{10})

Átvitel: 0 1 1 1 1 0 0 0

Eredmény: 1 1 0 1 0 1 1 0 (=214_{10})

Átszámítás 10-es számrendszerbe

q alapú számrendszerből legegyszerűbben a Horner elrendezéssel alakíthatunk át:

$$a_n*q^n+a_{n-1}*q^{n-1}+\ldots+a_0+$$
$$b_1*q^{-1}+b_2*q^{-2}+\ldots+b_m*q^{-m}=$$
$$(\ldots(a_n*q+a_{n-1})*q+\ldots+a_1)*q+a_0$$
$$(\ldots(b_m/q+b_{m-1})/q+\ldots+b_1)/q$$

$a_5f67=(((10*16+5)*16+15)*16+6)*16+7

%1_0110_0101=((((((((1*2+0)*2+1)*2+1)*2+0)*2+0)*2+1)*2+0)*2+1

$abcd – **hexadecimális** (16-os) számrendszerben megadott szám

%0100 – **bináris** (kettes számrendszerben megadott) szám

A számítógép kettes számrendszerben dolgozik, 10-es számrendszerből a Horner elrendezéssel alakítja át a számokat. A formulában a_i -t, b_j -t és q -t kettes számrendszerben kell megadni.

Kettes számrendszerből 10-es számrendszerbe 10-zel való ismételt osztással állítja elő az egész részt, és 10-zel való ismételt szorzással állítja elő a tört részt – hasonlóan ahhoz, ahogy korábban bemutattuk a 10-es számrendszerből 2-esbe való átszámítást.

Bit: egy bináris számjegy, vagy olyan áramkör, amely egy bináris számjegy ábrázolására alkalmas.

Bájt (Byte): 8 bit, 8 bites szám.

Előjeles fixpontos számok:

$2^8 = 256$ különböző 8 bites szám lehetséges.
Melyek jelentsenek negatív értékeket?

Előjeles számok szokásos ábrázolásai:

- előjeles abszolút érték,
- egyes komplemens,
- kettes komplemens,
- többletes.

Előjeles abszolút érték:

előjel és abszolút érték,

az első bit (balról) az előjel: 0: +, 1: -

Pl.: $+25_{10} = 00011001_2$, $-25_{10} = 10011001_2$.

\+ -

Jellemzők (8 bites szám esetén):

- a legkisebb szám -127, a legnagyobb 127,
- a nulla kétféleképpen ábrázolható.

Egyes komplemens:

az első bit az előjel (0: pozitív, 1: negatív).

Egy szám -1-szerese (negáltja) úgy kapható meg, hogy a szám minden bitjét negáljuk (ellenkezőjére változtatjuk).

Pl.: $+25_{10} = 00011001_2$,

$-25_{10} = 11100110_2$.

Jellemzők (8 bites szám esetén):

- a legkisebb szám -127, a legnagyobb 127,
- a nulla kétféleképpen ábrázolható.

Előjeles számok, egyes komplemens (Példa összeadásra)

	%1011_1100	(-67)	
+	%1111_1010	(-5)	
	%1011_0110	(-73)	

	%1111_1001	(-6)	
+	%0000_1001	(+9)	
	%0000_0010	(+2)	

Kettes komplemens:

az első bit az előjel (0: pozitív, 1: negatív).

Egy szám negáltja úgy kapható meg, hogy az egyes komplemenshez egyet hozzáadunk.

Pl.: $+25_{10} = 00011001_2$,

$-25_{10} = 11100110_2$ egyes komplemens,

$-25_{10} = 11100111_2$ kettes komplemens.

Jellemzők (8 bites szám esetén):

- a legkisebb szám -128, a legnagyobb 127,
- a nulla egyértelműen ábrázolható.

Előjeles számok, kettes komplemens (Példák összeadásra)

%1011_1101 (-67)
+ %1111_1011 (-5)
%1011_1000 (-72)

- %0100_0011=67
- %0000_0101=5
- %0100_1000=72

%1111_1010 (-6)
+ %0000_1001 (+9)
%0000_0011 (+3)

- %0000_0110=6
- %0000_1001=9

Kettes komplemens

- Képzése
  - Egyes komlemens +1
  - ….
- Kódolás (előjeles egész számok kódolására használt módszer, elterjedten használják erre a célra)
  - MSB --- előjel
    - 0 --- pozitív
    - 1 --- negatív
  - Ha pozitív, ugyanaz az érték, mint a „normál” számok esetén
  - Ha negatív, az abszolút érték meghatározásához kettes komplemenst kell képezni.
  - Ugy is megfogalmazható a kódolás, hogy:

A legmagasabb helyiértékű bit **$(+ 2^{n-1})$** helyett **(-2^{n-1})**–et ér

- Ek(x,n) --- x egyes komplemense n biten ábrázolva
- Kk(x,n) --- x kettes komplemense n biten ábrázolva
- $x+Ek(x)=\mathbf{2^n-1}$

  %1011_0010
  +%0100_1101
  %1111_1111=2^8-1
- $x+Kk(x)=\mathbf{2^n}$ (kivéve, ha x=0)
- $\mathbf{2^n}$ --- n biten ábrázolva **0** lesz, azaz x+Kk(x)=0
- Az ábrázolt érték =

  x, *ha MSB=0 és*

  $-(2^n-x)=x-2^n$ *ha MSB=1.*

Példák ketten komplemens kódolásra

- 23=%0001_011**1**
- Ek(23,8)=%1110_1000
- Kk(23,8)=%1110_100**1**
- Áé(+, %1110_1001)=233
- Áé(-, %1110_1001)=-23
- Áé(±,%0110_1001)=105
- 233=105+(+128)
- -23=-(256-233)=105+(-128)

- 6=%01**10**
- Kk(6,4)=%10**10**
- Kk(6,8)=%1111_1010
- Áé(+, %1111_1010)=250
- Áé(-, %1111_1010)=-6
- Áé(± ,%0111_1010)=122
- 250=122+(+128)
- -6=-(256-250)=122+(-128)

Többletes: a szám és a többlet összegét ábrázoljuk előjel nélkül (ez már pozitív!). **m** bites szám esetén a többlet általában $\mathbf{2^{m-1}}$ vagy $\mathbf{2^{m-1} - 1}$

Pl.: $+25_{10} = 10011001_2$, 128-többletes ábrázolás

$-25_{10} = 01100111_2$ 128-25=103

Jellemzők (128 többlet esetén):

- a legkisebb szám -128, a legnagyobb 127,
- a nulla egyértelműen ábrázolható.

Megjegyzés: a $\mathbf{2^{m-1}}$ többletes ábrázolás azonos a kettes komplemenssel – eltekintve az előjel bittől, amely épp ellentétes.

Használata: a lebegőpontos számok kitevő-részénél.

BCD (Binary Coded Decimal) ábrázolás: minden decimális számjegyet négy biten ábrázolunk.

Negatív számok BCD ábrázolása:

- Előjeles abszolút érték (Pentium: 80 bites egész)
- 9 vagy 10 komplemens kóddal.

Pl.: $+301_{10}$ = 0000 0011 0000 0001,

-301_{10} = 1001 0110 1001 1000 (9 komplemens),

-301_{10} = 1001 0110 1001 1001 (10 komplemens).

Lebegőpontos számok

előjel karakterisztika törtrész

Sok ekvivalens ábrázolási lehetőség, a leggyakrabban a törtrész első számjegye az adott szám első, nullától különböző számjegye (normalizált alak).

Példa: $154 = 0{,}0154 \times 10^4 = \mathbf{0{,}154 \times 10^3}$ ($= \mathbf{1{,}54 \times 10^2}$).

Illetve: $154 = \%0{,}10011010 \times 2^8 = \%1{,}0011010 \times 2^7$

Megjegyzések:

- A nulla ábrázolásához külön megállapodásra van szükség (általában csupa nulla számjegyből áll).
- A lebegőpontos ábrázolásoknál is meghatározható a legkisebb és a legnagyobb ábrázolható szám, továbbá a legkisebb és legnagyobb hiba.

IEEE 754 standard

single 32 bites, double 64 bites (, extended 80 bites).

típus	előjel	kitevőrész	| törtrész |
single	1 bit	8 bit 127-többletes	23 bit
double	1 bit	11 bit 1023-többletes	52 bit

single: Ha 0 < a kitevőrész < 255, a szám normalizált.

Normalizált tört vezető 1-es bitje nincs ábrázolva!

Normalizált számok (IEEE 754, single)

$0 <$ kitevőrész < 255

kitevőrész = kitevő + 127, 127 többletes.

Lehetséges kitevők: -126, -125, ... , +127.

Közvetlenül a törtrész elé kell képzelni egy *1*-est (**implicit bit**) és a bináris pontot.

Az ábrázolt szám: $\pm (1 + \text{törtrész}) * 2^{\text{kitevő}}$

```
Pl.: 1   0011 1111 1000 ... 0000₂ = 3F80 0000₁₆
    0,5  0011 1111 0000 ... 0000₂ = 3F00 0000₁₆
   -1,5  1011 1111 1100 ... 0000₂ = BFC0 0000₁₆
```

± kitevőrész *1*. törtrész

Normalizálatlan számok (IEEE 754, single)

Ha a kitevőrész = 0

Ilyenkor a kitevő -126 (! és nem -127),

a bináris pontot implicit *0* előzi meg (nincs ábrázolva).

Az ábrázolt szám: $\pm$ 0.(törtrész) * 2^{-126}

Pl.: $2^{-127} = 2^{-126} * 0{,}100\ldots0000_2 =$

$= 0000\ 0000\ 0100\ldots0000_2 = 0040\ \ 0000_{16}$

$\pm$ kitevőrész *0.* törtrész (2^{-1})

$-2^{-149} = -2^{-126} * 0{,}000\ldots0001_2 =$

$= 1000\ 0000\ 0000\ldots0001_2 = 8000\ \ 0001_{16}$

$\pm$ kitevőrész *0.* törtrész (2^{-23})

A legkisebb pozitív (single) normalizált szám:

$2^{-126} = \quad 2^{-126} * \ 1{,}000 \ldots 0000_2 =$

$= 0000\ 0000\ 1000 \ldots 0000_2 = 0080\ \ 0000_{16}$

□□□ ± kitevőrész *1*. törtrész

A legnagyobb pozitív (single) normalizálatlan szám:

$2^{-126} * \ 0{,}111 \ldots 1111_2 =$

$= 0000\ 0000\ 0111 \ldots 1111_2 = 007F\ FFFF_{16}$

□□□ ± kitevőrész *0*. törtrész

□□□$\approx 2^{-126}$

A különbségük csupán 2^{-149}.

Normalizálatlan számok (IEEE 754, single)

Ha a kitevőrész = 255

Túl nagy számok (túlcsordulás):

- ∞ (végtelen): pl. 1/0,
- **NaN** (Not a Number): pl. ∞ / ∞

Normalizát	±	0<kitevőrész<Max	bitminta
Nem normalizált	±	0	nem nulla bitminta
Nulla	+	0	0
Végtelen	±	111…1	0
Nem szám	±	111…1	nem nulla bitminta

8.5. ábra (IEEE 754, single)

Adattípusok

Alapkérdés: mit támogat a hardver (milyen utasítások vannak)? Ami nincs (pl. dupla pontosságú egész aritmetika), azt szoftveresen kell megcsinálni.

Numerikus típusok:

- előjel nélküli és előjeles egész számok (**8**, **16**, **32**, **64 bites**).
- lebegőpontos számok (**32**, **64**, néha **128 bites**),
- binárisan kódolt decimális számok: decimális aritmetika (**COBOL --> Y2K = 2000.** év probléma).

Az egyes gépek által támogatott numerikus típusok

P: Pentium 4, **U**: UltraSPARC III, **I**: I-8051

típus	1 bit	8 bit	16 bit	32 bit	64 bit	128 bit
bit	I					
előjeles egész		P U I	P U	P U	U	
előjel nélküli egész		P U	P U	P U	U	
BCD		P				
lebegőpontos				P U	P U	U

5.7-9. ábra

Karakterkódolás

ASCII (American Standard Code for Information Interchanges), 7 bites: vezérlőkarakterek, az angol abc kis és nagy betűi, szimbólumok, **2.43. ábra**

Latin-1 kód: 8 bites.

IS 8859: kódlap, **IS 8859-2:** magyar betűk is.

UNICODE (IS 10646), 32 bites: kódpozíciók (code point). Általában egy nyelv jelei egymás után vannak – a rendezés könnyű.

- Kínai, japán, koreai: fonetikus szimbólumok, Han ideogramok (20992 jel, nincsenek szótár szerint rendezve). ... Japán íráshoz kevés (> 50000 kanji jel van). (De pl. Braille sincs benne.)
- Elvileg 2^{31} különböző karakter, jel
- „Tárolás” pl. UTF-8 szerint:

```
00000000 00000000 00000000 0xxxxxxx <-> 0xxxxxxx
00000000 00000000 00000xxx xxxxxxxx <-> 110xxxxx 10xxxxxx
00000000 00000000 xxxxxxxx xxxxxxxx <-> 1110xxxx 10xxxxxx 10xxxxxx
00000000 000xxxxx xxxxxxxx xxxxxxxx <-> 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
000000xx xxxxxxxx xxxxxxxx xxxxxxxx <-> 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
0xxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx <-> 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
                                                                    10xxxxxx
```

További nem numerikus típusok

Logikai érték (boolean): igaz, hamis.
Leggyakrabban egy bájtban (szóban) ábrázolják.
Bit térkép.

Mutató (pointer): memória cím.

Bit: kapcsolók, lámpák beállítására, lekérdezésére beágyazott rendszerekben.

Adatábrázolás, adattípusok (HLA)

- bit, nible, byte
- Assembly alaptípusok
  - byte, word, dword, qword, tbyte, lword
- memória elérés (align)
- Utasítások
- „Magas szintű” adattípusok
  - Egész számok (int32, uns8)
    - bináris <=> BCD (Binary Coded Decimal)
    - pozitív <=> előjeles
  - Valós számok (real32, real64, real80)
  - Karakterek (char), karakterfüzérek (string)
  - Mutatók - memória címek (pointer)
  - Tömbök (tomb:int16[3,2];tomb2:byte:=[5,8,4,9];)

„Magas szintű” adattípusok

- Pozitív egészek (uns8, uns16, uns32, uns64, uns128)
  $0 \ldots 2^n-1$
- Előjeles egészek ábrázolása
  - kettes komplemens (int8, int16, int32, int64, int128)
    $-2^{n-1} \ldots +2^{n-1}-1$ ($int=uns-MSB*2^n$)
- Valós számok (real32, real64, real80)
  érték=$(-1)^S*2^{k-t}*$(1.mantissza)
  (IEEE 754 szabvány)
  real32: 1+ 8+23 10^{38} 6-7 jegy
  real64: 1+11+52 10^{308} 15-16 jegy
  real80: 1+15+64 10^{4800} 19-20 jegy
- Mutatók (pointer)
- Karakterek, karakter füzérek (char, string)
- Összetett típusok (tömbök, rekordok, uniók)

Utasítások ábrázolása (Pentium)

prefixum	***operációs kód***	***címzési mód***	***Operandus(ok)***
0 - 2 byte	1 byte	0 - 2 byte	0 - 8 byte

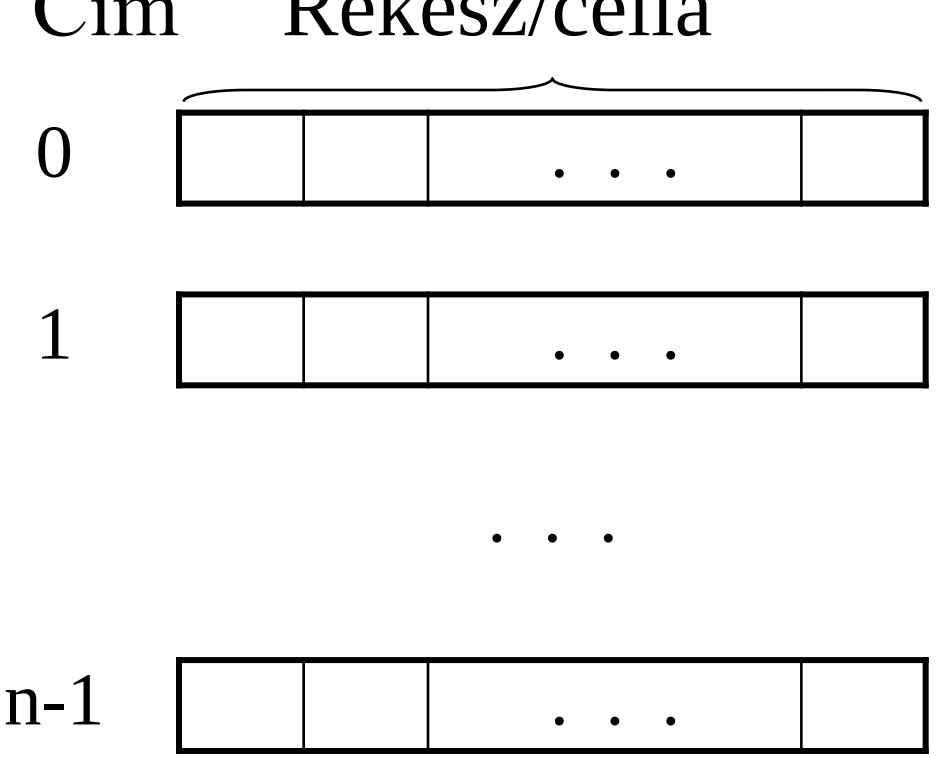
- ***Prefixum*:**
  - utasítás (ismétlés / LOCK),
  - explicit szegmens megadás: **MOV AX, CS:S ;** S nem a DS-ben, hanem CS-ben van**,**
  - Cím méret módosítás (16/32 bit)
  - Operandus méret módosítás (16/32 bit)
- ***Operációs kód*:** szimbolikus alakját mnemonic-nak nevezzük
- ***Címzési mód*:** hogyan kell az operandust értelmezni
  - ***Mod-r/m byte***
  - ***SIB byte***
- ***Operandus*:** mivel kell a műveletet elvégezni
  - Memória cím / eltolás
  - Azonnali operandus – konstans

Központi memória (2.9. ábra)

A programok és adatok tárolására szolgál.

Bit: a memória alapegysége, egy 0-t vagy 1-et tartalmazhat.

Memória rekesz (cella): több bit együttese. Minden rekesz ugyanannyi bitből áll. Minden rekeszhez hozzá van rendelve egy szám, a **rekesz címe**. Egy rekeszre a címével hivatkozhatunk. A rekesz a legkisebb címezhető egység.



Központi memória (2.9. ábra)

A bitek száma rekeszenként néhány számítógép-történetileg érdekes, kereskedelmi forgalomba került gépen (**2.10. ábra**)

Számítógép	Bit
Burroughs B1700	1
IBM PC	8
DEC PDP-8	12
IBM 1130	16
DEC PDP-15	18
XDS 940	24
Electrologica X8	27
XDS Sigma 9	32
Honeywell 6180	36
CDC 3600	48
CDC Cyber	60

Bájtsorrend

A legtöbb processzor több egymás utáni bájttal is tud dolgozni (**szó – word, …**).

A legmagasabb helyértékű bájt a szóban a

legalacsonyabb címen: **nagy (big) endian** **MSB first**

legmagasabb címen: **kis (little) endian** **LSB first**

(Most/Least **S**ignificant **B**yte **first)**

Ha egy 32 bites szó bájtjainak értéke rendre:

M[x]=a, M[x+1]=b, M[x+2]=c, M[x+3]=d, akkor a szó értéke:

$a*256^3+b*256^2+c*256+d$ $a+b*256+c*256^2+d*256^3$

Bájtsorrend (2.11. ábra)

A memória címek úgy vannak fölírva, hogy a legmagasabb helyértékű bájt van bal oldalon.

Cim	Nagy endian				Kis endian				Cím
0	0	1	2	3	3	2	1	0	0
4	4	5	6	7	7	6	5	4	4
8	8	9	10	11	11	10	9	8	8
12	12	13	14	15	15	14	13	12	12
	⟵ 32 bites szó ⟶				⟵ 32 bites szó ⟶				

Bájtsorrend (12. ábra)

A szövegek karaktereit mindkét esetben növekvő bájt sorrendben helyezik el

Cím — nagy endian

Cím				
0	*0* **T**	*1* **E**	*2* **X**	*3* **T**
4	*4* **12**	*5* **34**	*6* **56**	*7* **78**

kis endian — Cím

0	*3* **T**	*2* **X**	*1* **E**	*0* **T**
4	*7* **12**	*6* **34**	*5* **56**	*4* **78**

A **TEXT** szöveg és az **$12345678** hexadecimális szám elhelyezése a két géptípuson

Cím

Cím				
0	*0* **T**	*1* **E**	*2* **X**	*3* **T**
4	*4* **78**	*5* **56**	*6* **34**	*7* **12**

Problémák a gépek közötti kommunikációban!

Kódolás: adat + ellenőrző bitek = **kódszó**.

Két kódszó Hamming távolsága: az eltérő bitek száma.
Pl.: 11001 és 11011 (Hamming) távolsága = 1.

Hibaérzékelő kód: bármely két kódszó távolsága > 1:
paritás bit.

d hibás bit javítása: a kódszavak távolsága > 2*d*.

Egy hibát javító kód (**2.13. ábra**):
m adat, *r* ellenőrző bit, összesen $n = m + r$.
2^m „jó” szó, + minden „jó” szónak (legalább) *n* db „egyhibás” szomszédja van, ezért

$$(1+n)2^m \leq 2^n = 2^{m+r},$$

2^m -mel egyszerűsítve: $(1+n)=m + r +1 \leq 2^r$,
vagy másképp: $n=m + r < 2^r$ szükséges.

Például

- 8 bites adatok esetén legalább 4 ellenőrző bit szükséges.
  - $8+3>2^3$, de $8+4<2^4$
- 16 bites adatok esetén legalább 5 ellenőrző bit szükséges.
  - $16+4>2^4$, de $16+5<2^5$
- 32 bites adatok esetén legalább 6 ellenőrző bit szükséges.
  - $32+5>2^5$, de $32+6<2^6$
- stb…
- 2^k bites adatok esetén legalább $k+1$ ellenőrző bit szükséges. (Elég nagy k esetén.)
  - $2^k+k>2^k$, de $2^k+(k+1)<2^{(k+1)}$

Kódolás: adat + ellenőrző bitek = **kódszó**.

Két hibát javító kód:

m adat, r ellenőrző bit, összesen $n = m + r$.

2^m „jó” szó, + minden „jó” szónak n db „egyhibás” és $n*(n-1)/2$ „kéthibás” szomszédja van, ezért

$$(1+n*(n+1)/2)* 2^m \leq 2^n = 2^{m+r},$$

2^m -mel egyszerűsítve:

$$1+n*(n+1)/2 \leq 2^{r+1},$$

vagy másképp: $n*(n+1)/2 < 2^r$ szükséges.

Három hibát javító kód: ….