

# Számrendszerek

## 1. Elméleti alapok

### 1.1. A kettes számrendszerről

Számlálás közben mi *tízeseivel csoportosítunk* (valószínűleg azért, mert 10 ujjunk van). Ezt a számírásunk is követi. A helyiértékek: egy, tíz, száz ( $10 \times 10 = 10^2$ ), ezer ( $10^3$ ), tízezer ( $10^4$ ), ...

De *nem volt ez mindig így*:

- ♦ Mezopotámia, sumérok: a 6, 12, 60 számoknak kitüntetett szerepe volt. Innen származik az időmérés: 1 óra az 60 perc, egy nappal pedig 12 óra (és az éjjel is 12 óra). 12 hónap egy év.
- ♦ Matematika, szögmérés: 1 fok = 60 perc
- ♦ Angol (és még néhány más) nyelvben a 11 és 12 számoknak külön nevük van.
- ♦ Köznyelvben a múlt századokban a tucat szó (12-t jelent) elterjedt volt.<sup>1</sup>
- ♦ római számok írása: 5-ös csoport is kitüntetett.
- ♦ Latin, olasz, francia: A számnevekben 20-as csoportosítást fedezhetünk fel.

Számítógép-generációk:

- ♦ Relés számítógépek (elektromechanikus elv)
- ♦ Elektroncső (elektromos elv)
- ♦ Tranzisztor (elektromos elv)

A relé, elektroncső, a tranzisztor<sup>2</sup> a számítógépekben kétféle állapotban tud működni, egy kapcsolóhoz hasonlóan: bekapcsolt/kikapcsolt; vezet/nem vezet; vagy van feszültség/nincs feszültség. (A mágneslemez is kétféle állapottal (észak-dél vagy dél-észak) működik.)

Mi 10-es számrendszert használunk, mert ez nekünk „testhezálló” (az ujjaink száma miatt<sup>3</sup>), de a számítógépeknek ez nem kényelmes, nekik a kétféle állapot esik jól. Tehát kétféle jellel is mindent le kell tudni írni.

Tíz-es számrendszerben 10 féle jelünk van (a számjegyek), tízeseivel csoportosítunk.

Kettes számrendszerről beszélünk, ha két jelet fogunk használni, azaz kettesével csoportosítunk. Ekkor 2-es, 4-es, 8-as, 16-os, ... csoportokat képezünk, ezek lesznek a helyiértékek:



1. ábra: Csoportosítás 2-es számrendszerben

1 darab 8-as csoport  
0 darab 4-es csoport  
1 darab 2-es csoport  
1 darab 1-es

Helyiértékek: 8 4 2 1  
A szám leírva: 1 0 1 1

Jelölés: Ha nem egyértelmű, hogy a „1011” szám az tízes számrendszerben értelmezett szám vagy pedig kettes számrendszerbeli szám (mint a példában), akkor a szám után szoktuk írni a számrendszert: 1011 az 10-es (decimális) számrendszert jelöl, a 1011 pedig kettes (bináris) számot.

AZ INFORMÁCIÓ MÉRTEKEGYSÉGE a *bit*, azaz egy darab kettes számrendszerbeli számjegy (binary digit).

A kettes számrendszer terjedős: a példában szereplő „tizenegyes” számot kettes számrendszerben egy 4 számjegyű számmal tudunk leírni (10-es számrendszerben pedig kétjegyű szám is bőven elég). Ezért a számítógépek alapegységének a 8 bitből álló bináris számot választották, ennek a neve BAJT (angolosan: BYTE). Ez azt jelenti, hogy a számítógép általában 8 bittel (vagy ennek többszörösével) végez műveletet egyszerre.

<sup>1</sup> Valószínűleg azért, mert a gyakorlatban könnyebb vele dolgozni. Például amikor a piacon néhány ember osztzkodik: egy tucat tojást lehet két-, három-, négy- és hatfelé is osztani. (A tíz csak kettő és öt felé osztható.)

<sup>2</sup> És a tranzisztorokból álló integrált áramkörök is.

<sup>3</sup> Általános iskola 2 osztályban sokan az ujjakon számolnak.

<i>Kettes számrendszerben</i>	<i>Hasonló gondolat 10-es számrendszerben</i>
A számok 0-1 közti számjegyekből állnak. A helyiértékek (jobbról balra haladva):	A számok 0-9 számjegyekből állnak. A helyiértékek (jobbról balra haladva):
$2^0 = 1;$ $2^3 = 8;$ $2^6 = 64;$ $2^9 = 512;$ $2^1 = 2;$ $2^4 = 16;$ $2^7 = 128;$ $2^{10} = 1024;$ $2^2 = 4;$ $2^5 = 32;$ $2^8 = 256;$ ...	$10^0 = 1;$ $10^3 = 1000;$ $10^1 = 10;$ $10^4 = 10\,000;$ $10^2 = 100;$ $10^5 = 100\,000;$ ...
Egy szám így nézhet ki: 11101 Ennek az értéke (jobbról balra olvasva): $1 \cdot 1 + 0 \cdot 2 + 1 \cdot 4 + 1 \cdot 8 + 1 \cdot 16 = 29$	Egy szám így nézhet ki: 10591 Ennek értéke (jobbról balra olvasva): $1 \cdot 1 + 9 \cdot 10 + 5 \cdot 100 + 0 \cdot 1000 + 1 \cdot 10000 = 10\,591$
A legnagyobb 4 jegyű szám: $1 \cdot 1 + 1 \cdot 2 + 1 \cdot 4 + 1 \cdot 8 = 15$ , ami nem más, mint $2^4 - 1$ . tehát négyjegyű számokkal $2^4 = 16$ féle számot tudunk leírni 0-tól 15-ig	A legnagyobb 4 jegyű szám: $9 \cdot 1 + 9 \cdot 10 + 9 \cdot 100 + 9 \cdot 1000 = 9999$ , ami nem más, mint $10^4 - 1$ . Tehát négyjegyű számokkal $10^4 = 10000$ féle számot lehet leírni 0-tól 9999-ig.
8 jegyű számokkal: $2^8 = 256$ féle számot tudunk leírni 0-tól $2^8 - 1$ -ig (azaz 255-ig) ( $255 = 11111111_B$ )	8 jegyű számokkal $10^8 = 100\,000\,000$ féle számot tudunk leírni 0-tól $10^8 - 1$ -ig (azaz 99 999 999-ig).
A hosszú számokat (a könnyebb olvashatóság kedvéért) 4-es csoportokban szoktuk leírni: 1010 0110 0101	A hosszú számokat (a könnyebb olvashatóság kedvéért) 3-es csoportokban szoktuk leírni: 193 326 042
A „kilo” prefixum értéke 1024 (informatikában)	A „kilo” prefixum értéke 1000
A számok után írt nulla 2-vel való szorzást jelent, mert ezzel a „kettedes vesszőt” mozgatjuk. <sup>4</sup>	A számok után írt nulla 10-zel való szorzást jelent, mert a tizedes vesszőt mozgatjuk.

### **Prefixumok (kilo, mega, giga) a számítástechnikában**

Prefixum=előtag. Prefixumok jelentése:

1000 méter = 1 kilométer

1000 gramm = 1 kilogramm

A „kilo” jelentése tehát 1000.

1 méter = 1000 milliméter

1 kilométer = 1000 méter = 1 000 000 milliméter

A „milli” jelentése ezred.

A függvénytáblázatokban megtalálhatóak a prefixumok ( $10^{-15}$ -től  $10^{18}$ -ig). Néhány példa:

kilo = 1000

Mega = 1000 kilo = 1 000 000

Giga = 1000 Mega = 1 000 000 kilo = 1 000 000 000

Tera = 1000 Giga = 1 000 000 Mega = ...

Informatikában a kettes számrendszert használjuk, amiben nem az 1000, hanem az 1024 a kerek szám. ( $1024 = 2^{10}$ ) Mivel ez a két szám közel egyenlő, ezért *informatikában* a kilo 1024-et jelent:

1024 bájt = 1 kilobyte (rövidítve: 1024 B = 1 KB)

Nagyobb számokkal:

1 MB = 1024 KB = 1024 \* 1024 B (KB = kilobyte, MB = Megabyte, GB = Gigabyte)

1 GB = 1024 MB = 1024 \* 1024 KB = 1024 \* 1024 \* 1024 B = 1 073 741 824 B

Ezért van az, hogy akinek hivatalosan 64 MB memóriája van, annak a számítógépe nem 64 000, hanem 65 536 KB-ot számol össze bekapcsoláskor.

Ez az oka, hogy egy 60 GB-os merevlemeznek valójában 56 GB-nál is kevesebb a kapacitása. (A „boltosok” és a gyártók 1000-rel számolnak 1024 helyett, mert így nagyobb számot írhatnak rá, hogy jobban el tudják adni.)

### **Átváltás 10-es és 2-es számrendszer között**

#### BINÁRISBÓL DECIMÁLISBA:

$2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$

$01101100_B = 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 2^6 + 2^5 + 2^3 + 2^2 = 64 + 32 + 8 + 4 = 108_D$

#### DECIMÁLISBÓL BINÁRISBA:

<sup>4</sup> Mert egy ahogy 10-es számrendszerben egy helyiértékkel léptetés 10-es szorzót jelent (pl.  $10^2$  helyett  $10^3$ ), úgy kettes kettes számrendszerben ez 2-es szorzót jelent (pl.  $2^2$  helyett  $2^3$ ).

$$122_D = 0 \cdot 128 + 1 \cdot 64 + 1 \cdot 32 + 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 01111010_B$$

A módszer lényege, hogy megmondjuk, hogy hány 1-es, 2-es, 4-es, ..., 128-as, ... bináris számjegy kell ahhoz, hogy kijöjjön a 122 decimális szám. Ez egy kicsit nehéz, mert sokat kell kivonni és számolni hozzá.

#### ÁTVÁLTÁS DECIMÁLISBÓL (PROFI MÓDON) BINÁRISBA:

Hogy megértsük, először vegyünk egy egyszerű feladatot: váltsuk át a 2836 decimális (10-es számrendszerű) számot 10-es számrendszerbe! Ez azt jelenti (azon felül, hogy semmit sem kell csinálni), hogy mondjuk meg, hány 1-es, 10-es, 100-as, ... kell hozzá.

Ez nem nehéz, ránézésre ki tudjuk olvasni. De ha az a feladat, hogy ne ránézzünk, hanem *számítsuk ki*, hogy melyik helyiértékből mennyi van, akkor így számolnánk:

		Maradék
A 2836-ot elosztjuk 10-zel, a maradékot (egyesek száma) a vonaltól jobbra, az egészrészt (ahányszor megvan) alá írjuk:	2836	6
Tovább osztunk, hogy megkapjuk a 10-esek számát:	283	3
Majd a 100-sok számát:	28	8
Az ezreseket:	2	2
Tízezres már nincs, mert elértük a 0-t:	0	

A maradékokat kiolvasva: kell 6 darab egyes, 3 darab 10-es, 8 darab 100-as, 2 darab ezres.

A számot pedig letről fölfelé lehet kiolvasni: 2836

Most váltsuk át a 122 decimális számot kettes számrendszerbe:

Elosztjuk 2-vel, hogy megkapjuk a egyesek számát (leírjuk alá), a maradékot (egyesek száma) pedig a vonaltól jobbra:	122	0
Majd ismét elosztjuk, hogy megkapjuk a 2-esek számát:	61	1
Majd ismét elosztjuk, hogy megkapjuk a 4-esek számát:	30	0
Majd a 8-asok számát:	15	1
Majd a 32-esek számát:	7	1
...	3	1
	1	1

Lentről fölfelé kiolvasva: 1111010, azaz ugyanaz, mint az előző feladatban. Még elé szoktunk írni 0-akat, hogy 8 számjegy hosszú legyen<sup>5</sup>: 0111 1010

#### **Összeadás, kivonás természetes számokkal**

<i>Kettes számrendszerben</i>	<i>Hasonló gondolat 10-es számrendszerben</i>
$1+1 = 10$	$9 + 1 = 10$
$111 + 1 = 1000$	$999 + 1 = 1000$
$\begin{array}{r} 1111\ 0100 \\ +\ 0110\ 0101 \\ \hline 10101\ 1001 \end{array}$	$\begin{array}{r} 99910900 \\ +\ 01100101 \\ \hline 101011001 \end{array}$
Magyarázat: kettes számrendszerben az „1” a legnagyobb számjegy, ha ezt megnöveljük, akkor az értéke 0 lesz, és keletkezik egy átvitel.	Magyarázat: tízes számrendszerben a „9” a legnagyobb számjegy. Ha ezt megnöveljük, akkor az értéke 0 lesz, és keletkezik egy átvitel. <sup>6</sup>

<sup>5</sup> Mert a byte-ban mindig 8 bit van, mint ahogy a vízórában is mindig 5 számjegy van, legfeljebb az eleje 0-ákból áll

<sup>6</sup> A példában azért szerepel 9-es és 1-es is, mert kettes számrendszerben az 1 egyben a legnagyobb számjegy is, tehát kicsit a 10-es rendszerbeli 9-eshez is hasonló (ha hozzáadunk 1-et, 10-et kaponk).

<i>Kettes számrendszerben</i>	<i>Hasonló gondolat 10-es számrendszerben</i>
<p>Az összeadás művelete a fenti példán: (jobbról balra)</p> <p>0+1 az egy, leírjuk az 1-et, nincs átvitel.</p> <p>0+0 az nulla, leírjuk a 0-t, nincs átvitel.</p> <p>1+1=10<sub>B</sub>,<sup>7</sup> leírjuk a 0-t, marad az 1.</p> <p>0+0=0, plusz az átvitel az 1, leírjuk az 1-et.</p> <p>1+0=1, leírjuk az 1-et, nincs átvitel.</p> <p>1+1=10, leírjuk a 0-t, maradt az 1.</p> <p>1+1=10, + átvitel, az 11, leírjuk az 1-et, marad 1.</p> <p>1+0=1, + átvitel az 10, leírjuk a 0-t, marad 1.</p> <p>semmi (azaz 0) + átvitel az 1, leírjuk.</p> <p>Ugyanez 10-es számrendszerben:</p> $\begin{array}{r} 11110100_B = 244_D \\ + 01100101_B = + 101_D \\ \hline 101011001_B = 345_D \end{array}$	<p>Az összeadás művelete a fenti példán: (jobbról balra)</p> <p>0+1 az egy, leírjuk az 1-et, nincs átvitel.</p> <p>0+0 az nulla, leírjuk a 0-t, nincs átvitel.</p> <p>9+1=10, leírjuk a 0-t, marad az 1.</p> <p>0+0=0, plusz az átvitel az 1, leírjuk az 1-et.</p> <p>1+0=1, leírjuk az 1-et, nincs átvitel.</p> <p>9+1=10, leírjuk a 0-t, maradt az 1.</p> <p>9+1=10, plusz az átvitel, az 11, leírjuk az 1-et, marad 1.</p> <p>9+0=9, plusz átvitel, az 10, leírjuk a 0-t, marad 1.</p> <p>semmi (azaz 0) + átvitel az 1, leírjuk.</p>
<p>Kivonás:</p> $\begin{array}{r} 1110\ 1011 \\ - 0111\ 0110 \\ \hline 0111\ 0101 \end{array}$	<p>Kivonás:</p> $\begin{array}{r} 19931031 \\ - 09940930 \\ \hline 09990101 \end{array}$
<p>Magyarázat: A kivonás hasonló, mint 10-es számrendszerben. Ha nagyobb számot kell kivonni a kisebből (azaz 0-ból 1-et), akkor keletkezik egy átvitel, és 10<sub>B</sub> = 2<sub>D</sub>-ből vonunk ki 1-et. (10-1 = 1)</p>	<p>Ha kisebb számból kell kivonni a nagyobbat: például 0-ból a 9-et, akkor nem 0-ból, hanem 10-ből vonjuk ki a 0-t, és lesz egy átvitel (kölcson).</p>

A világon 10-féle ember létezik: aki érti a kettes számrendszert és aki nem.

## 1.2. A tizenhatos (hexadecimális) és nyolcas (oktális) számrendszer

Hogy az informatikusoknak ne kelljen olyan hosszú bináris számokat írni, ezért a 16-os számrendszert is használják. A számok leírásához kettes számrendszerben 2, tízes számrendszerben 10, tizenhatos számrendszerben 16 különböző számjegy kell (azaz csak *kellene*, mert csak 10 számjegyünk van). Mivel nem akartak hat új jelet bevezetni az új számjegyeknek, ezért az angol ABC első hat betűjét használták fel erre a célra. A=10, B=11, C=12, D=13, E=14, F=15. Ha a számítógéppel kapcsolatban ilyen számokat látunk, hogy 2F8 vagy 7D5, akkor „az valószínűleg nem sült bolondság, hanem tizenhatos számrendszer béli szám”<sup>8</sup>.

Néha jelölni kell, hogy a „34” az 10-es vagy 16-os számrendszerbeli szám. Ezért gyakran így írják: 0x34 vagy 34h esetleg \$34.

Az egész 16-os számrendszernek az a nagy előnye, hogy minden négyjegyű kettes számrendszerbeli számot egy darab egyjegyű számmal le tudunk írni. Ráadásul nagyon könnyű átváltani a két számrendszer között oda vissza: Egy bájt az pont egy kétjegyű hexadecimális számmal ábrázolható, így a jobb oldali 4 és a jobb oldali 4 bináris számjegy is egy-egy hexa számjegyet jelent.

A helyiértékek: 1-es, 16-os, 16\*16-os, 16\*16\*16-os, ...  
Mennyit ér 10-es számrendszerben a 7D4?

10-es	2-es	16-os	10-es	2-es	16-os
0	0	0	16	1 0000	10
1	1	1	17	1 0001	11
2	10	2	18	1 0010	12
3	11	3	19	1 0011	13
4	100	4	20	1 0100	14
5	101	5	21	1 0101	15
6	110	6	22	1 0110	16
7	111	7	23	1 0111	17
8	1000	8	24	1 1000	18
9	1001	9	25	1 1001	19
10	1010	A	26	1 1010	1A
11	1011	B	27	1 1011	1B
12	1100	C	28	1 1100	1C
13	1101	D	29	1 1101	1D
14	1110	E	30	1 1110	1E
15	1111	F	31	1 1111	1F

1. táblázat: Átváltás 2-es és 16-os számrendszer között

<sup>7</sup> Természetesen 1+1=2, de a 2 kettes számrendszerben leírva 2<sub>D</sub>=10<sub>B</sub>

<sup>8</sup> Kezdő UHU-Linux Felhasználók Kódexe, 40. oldal

Az OKTÁLIS SZÁMOK pedig 0-7 közti számjegyekből állnak. Az oktális és a bináris számok is könnyen válthatóak át egymásba: egy oktális számjegy pont 3 biten ábrázolhatóak.

A 2. táblázat megadja a különféle számrendszerek egyes helyiértékeinek 10-es rendszerbeli értékét, így segítséget nyújthat az átváltáshoz. Tört számok esetén is segíthet, mert bár a tizedes (vagy kettedes) vesszőtől jobbra levő értékeket nem tünteti fel a táblázat, könnyen kiszámolható az  $1/x$  képlettel (ahol  $x$  a megfelelő helyiérték a táblázatból).<sup>9</sup>

Számjegy	Helyiérték	Helyiértékek a különféle számrendszerekben			
		10-es számr. (decimális)	2-es számr. (bináris)	8-as számr. (oktális)	16-os számr. (hexadecimális)
1	0	$10^0 = 1$	$2^0 = 1$	$8^0 = 1$	$16^0 = 1$
2	1	$10^1 = 10$	$2^1 = 2$	$8^1 = 8$	$16^1 = 16$
3	2	$10^2 = 100$	$2^2 = 4$	$8^2 = 64$	$16^2 = 256$
4	3	$10^3 = 1000$	$2^3 = 8$	$8^3 = 512$	$16^3 = 4096$
5	4	$10^4 = 10000$	$2^4 = 16$	$8^4 = 4096$	$16^4 = 65536$
6	5	$10^5 = 100000$	$2^5 = 32$	$8^5 = 32768$	$16^5 = 1048576$
7	6	$10^6 = 1000000$	$2^6 = 64$	$8^6 = 262144$	$16^6 = 16777216$
8	7	$10^7 = 10000000$	$2^7 = 128$	$8^7 = 2097152$	$16^7 = 268435456$
9	8	$10^8 = 100000000$	$2^8 = 256$	$8^8 = 16777216$	$16^8 = 4294967296$
10	9	$10^9 = 1000000000$	$2^9 = 512$	$8^9 = 134217728$	$16^9 = 68719476736$
11	10	$10^{10} = \dots$	$2^{10} = 1024$	$8^{10} = 1073741824$	$16^{10} = \dots$
12	11	$10^{11} = \dots$	$2^{11} = 2048$	$8^{11} = 8589934592$	$16^{11} = \dots$

Az első oszlop a számjegyek helye a számban, jobbról balra számolva. (Például az „508”-as számból a „8”-as az első számjegy (egyese), és az „5”-ös a harmadik számjegy (aminek helyiértéke 10-es rendszerben 100-as, 8-as rendszerben 64-es, 16-os rendszerben pedig 256-os))

2. táblázat: A helyiértékek 10-es, 2-es, 8-as és 16-os számrendszerekben

### 1.3. BCD számok

Ez a Binary Coded Decimal (azaz Binárisan Kódolt Decimális) számokat jelöli. Az ilyen szám nem egy 8 bites szám, hanem két darab 4 bites szám, melyek a 0-9 értékeket vehet fel. Tehát a számjegyeket tároljuk. Mivel nem egy szám, hanem két független számjegy, ezért az alpműveleteket nem a „hagyományos” módon kell elvégezni.

Például a 17-es szám BCD-ben így néz ki: 0001 0111 (az első 4 bit értéke 1, a második 4 bité 7).

### 1.4. Szövegek tárolása

FELADAT: A számítógép tudjon tárolni betűket is.

A PROBLÉMA: A számítógép csak számokat tud tárolni, csak számokkal tud műveleteket végezni. (Azt is leginkább bináris számokkal: vezet/nem vezet, van feszültség/nincs feszültség)

A MEGOLDÁS: A betűket, írásjeleket stb. (egy szóval karaktereket) kódolni fogjuk. A kódolás azt jelenti, hogy egy-egy betűhöz egy-egy számot rendelünk. Például a számok jelentsék a következőt:

1=A, 2=Ä, 3=B, 4=C, 5=CS, 6=D, 7=DZ, 8=DZS, 9=E, 10=É, 11=F, 12=G ...

Ezeket a kódokat és jelentéseket egy táblázatban felírjuk egy lapra, hogy el ne felejtjük. Ezt a táblázatot hívják *kódlap*nak.

Ezután ha például azt látjuk leírva, hogy 9, 3, 10, 6 akkor (az előbbi kódlap alapján értelmezve) ki tudjuk találni, hogy miről van szó.<sup>10</sup> Persze többféle módon lehet számokat rendelni a karakterekhez, a fenti az csak egy találmányra kiválasztott példa.

Ezzel a hozzárendeléssel az a baj, hogy önkényesen történik. Lehet, hogy a világ más részén valaki úgy is gondolhatja, hogy az egyjegyű számokhoz (0-9) az írásjeleket és a szünetet rendeli, és 10-től kezd az A-betűt.<sup>11</sup> Ha két program nem ugyanazt a betű-szám összerendelést használja, akkor amit az egyik programmal elkészítettünk, a másik programból megnézve zagyvaságnak tűnik. Ez nem jó.

<sup>9</sup> Például a második tizedesjegyhez kinézzük a  $10^2 = 100$ -at, tehát értéke  $1/100$  lesz.

Ugyanez 16-os számrendszerben: kinézzük a  $16^2 = 256$ -ot, tehát értéke  $1/256$  lesz.

<sup>10</sup> Dél körül.

<sup>11</sup> Egy orosz programozónak pedig valószínűleg nem a magyar ABC jut az eszébe...

Közösen meg kell állapodni, hogy melyik szám melyik betűt jelenti. A legelterjedtebb ilyen megállapodás az Amerikai Szabványos Kódolás (American Standard Code for Information Interchange, röviden ASCII).<sup>12</sup>

Mivel az alapegység a byte, ami 8 bitből áll, ehhez a számtartományhoz rendelték karaktereket. Ebből egy bitet akkoriban a kommunikáció ellenőrzésére használták, ezért csak a maradék 7 biten tárolt 128 számhoz (0-127 közötti értékek) rendelték karaktereket. Íme néhány karakter kódja:

Szám	Binárisan	Hex	Mit jelent?	Szám	Binárisan	Hex	Mit jelent?
32	0010 0000	20	Szünet (space)	48	0011 0000	30	0
33	0010 0001	21	Felkiáltójel	49	0011 0001	31	1
...	...	...	... (írásjelek és matematikai jelek)	50	0011 0010	32	2
47	0010 1111	2F		...	...	...	...
				57	0011 1001	39	9
				58-63		...	További jelek: ; : < = > ?
64	0100 0000	40	@	97	0110 0000	60	`
65	0100 0001	41	A	98	0110 0001	61	a
66	0100 0010	42	B	99	0110 0010	62	b
67	0100 0011	43	C	100	0110 0011	63	c
...	...	...	...	...	...	...	...
90	0101 1010	5A	Z	122	0111 1010	7A	z
91-96			További írásjelek: [ \ ] ^ _	123-127			További írásjelek: {   } ~

A 0-31 közötti számkódok különleges célra lettek meghatározva<sup>13</sup>, a „rendes” karakterek 32-vel (hexa 20) kezdődnek.

Amit érdemes megjegyezni az, hogy 48-tól (hexa 30) kezdődnek a számjegyek, és 65 (hexa 41) a nagy „A”, és 65+32 (hexa 41+20) a kis „a” betű).

A másik fontos észrevétel, hogy nem tartalmaz ékezetes (magyar) karaktereket, mivel eredetileg az USA számára (angol nyelvhez) készült. Ebből elég sok bonyodalom adódik, melyeket most nem részletezünk. A lényeg, hogy a 128-255 tartományban minden ország meghatározta, hogy melyik szám melyik ékezetes betűt jelentse. Így alakultak ki (csak Magyarországot tekintve) a CWI, IBM852, Windows-1250, ISO-8859-2 kódlapok.<sup>14</sup> (Jelenleg az ISO-8859-2 az elterjedten használt kódlap.) A magyar kódlap és a többi ország kódlapjai is mind ugyanazokhoz a számkódokhoz (128-255) rendelik a saját ékezetes betűiket.<sup>15</sup> Ezért egyszerre csak 1 kódlap használható. Így viszont nem lehet például egy szövegben leírni magyar, francia cirill és görög betűket.

Ezért megterveztek egy nagy egyesített (unified) kódlapot (code page), amiben a világ összes betűje szerepel. Ez lett az UNICODE. Ez a szabvány a karaktereket – hogy a sok ezer kínai, indiai, afrikai betű is beleférjen -- nem 8, hanem 21 biten ábrázolja.<sup>16</sup> A szabványt néhány évente újra tárgyalják, és ha szükséges, új karaktereket vesznek fel a szabványba.

Mivel a UNICODE 8 bitnél hosszabb értékekkel dolgozik (a mai számítógépek pedig a byte-ot szeretik), ezért csak lassan terjedt el.<sup>17</sup>

Megemlíteném még, hogy az MS-Word már 1997 óta a UNICODE egyik 16-bites részalmazát használja. Itt egy betűt egy (vagy két) darab 16 bites „karakter” ír le. Azaz egy karakter két egymás utáni byte-ot foglal el. Ha egy szöveget úgy látunk, hogy minden második betűje egy négyzet, az azért van, mert azt nem byte-onként (8 bitenként) kellene értelmezni (ahogy a programunk teszi), hanem kétbyte-onként (16 bitenként).

Mostanában (2004) kezdik széleskörűen használni *UTF-8*-nak nevezett kódolást<sup>18</sup>, ami a 21 bites UNICODE egy speciális 8 bites kódolása. Így a magyar szövegek alig lettek hosszabbak, mint korábban (a legtöbb betű 1 byte-os), de mégis van lehetőség ugyanabban a szöveggörnyezetben leírni az „egzotikus” betűket is (görög, cirill, kínai).<sup>19</sup>

12 Az IBM ekkor már belsőleg használt egy EBCDIC nevű kódolást. Az IBM nagygépek még most is így működnek belsőleg.

13 Mint például: lapdobásjel (nyomtatónak); tárcsázásjel (modemnek), újsor-jel (táv-írógépnek); üzenet/fájl vége stb.

14 Azért nem 1 darab van, hanem ennyi, mert az IBM és Microsoft is a saját (több országra kiterjedő) szempontjaik szerint saját kódlapokat határoztak meg. (Az, hogy egyes országok már addigra kitaláltak valamit, az nem számított.) Ebből volt is kavardás az elmúlt időszakban. A 852-es kódlap magyar szabvány lett: MSZ 7795-3:1992 néven. Mai napig ez a magyar nyelvű DOS betűinek kódolása.

15 Például a 165 az jelenthet (magyar kódlap szerint) „á” betűt, de egy svéd vagy francia kódlap szerint egy másik betűt jelent.

16 Néhány éve kérték, hogy a Star Trek-ből ismert harcias Klingon nép ABC-jét is vegyék bele az UNICODE kódlapba. De a mérnökökből álló döntőbizottság végül komolytalannak tartotta az ötletet. Az ősi magyar rovásírás még elbírálás alatt áll.

17 Két ok lassítja a terjedését: egyrészt, hogy a 21 bites kód nem 256, hanem kb. 2 millió különböző karaktert jelent, és ennyit nehéz kezelni egy programnak. A másik baj, hogy mivel 4 byte-on ábrázolnak egy betűt, ez 4-szeres helyfoglalást jelent.

18 Az ajánlás azRFC3629 (2003 november). Ez az 1998-as RFC2279 újragondolt, módosított verziója.

19 De az UNICODE sem „tökéletes”, mert a magyar nyelvtan szerint az „ny” olyan *egyetlen* betű, ami két betűjegyből (n és y) áll, ez elválasztásnál lehet fontos. (Nem beszélve a „rég” kettős betűkről, mint például az eő (Weöres) vagy cz (Losonczy))