

# Számítógépes alkalmazások Alapok

Soós Sándor

Nyugat-magyarországi Egyetem  
Simonyi Károly Műszaki, Faanyagtudományi és Művészeti Kar  
Informatikai és Gazdasági Intézet

E-mail: [soossandor@inf.nyme.hu](mailto:soossandor@inf.nyme.hu)

Sopron, 2015.



# Tartalomjegyzék

- 1 Bevezetés
- 2 Alapfogalmak
  - Kettes számrendszer
  - Logikai és aritmetikai műveletek
  - A rendszerszemlélet
- 3 A számítógép működése
  - Kódolás, kódrendszerek
  - Számok kódolása
- 4 Befejezés



# Miről lesz szó a mai órán?

- A számítógépek működésének alapjai
- Kettes számrendszer
- Logikai alpműveletek
- A rendszer fogalma, rendszerszemlélet
- Kódolás, kódrendszerek, számok kódolása



# Outline

- 1 Bevezetés
- 2 Alapfogalmak
  - Kettes számrendszer
  - Logikai és aritmetikai műveletek
  - A rendszerszemlélet
- 3 A számítógép működése
  - Kódolás, kódrendszerek
  - Számok kódolása
- 4 Befejezés



# Kettes számrendszer

- Mi az a kettes számrendszer?
- Miért jó, ha ezt használjuk a számítógép működésének alapjául?
- Két számjegy (0, 1)  $\Leftrightarrow$  Kétállapotú jelenségek (áram, mágnesség)
- Az áramkörök modellezhetők a matematikai logika elméletével

Bináris számjegy	Áram	Mágnesség
0	nincs áram (0V)	pozitív
1	van áram (5V, 3V)	negatív



# Outline

- 1 Bevezetés
- 2 Alapfogalmak
  - Kettes számrendszer
  - Logikai és aritmetikai műveletek
  - A rendszerszemlélet
- 3 A számítógép működése
  - Kódolás, kódrendszerek
  - Számok kódolása
- 4 Befejezés



# Logikai műveletek igazságtáblája I

- ❶ Negáció – NOT (egyváltozós) ellentettjére változtatja a logikai értéket

$A$	$\neg A$
0	1
1	0

- ❷ És – AND (kétfváltozós) igaz, ha mindkét operandus igaz

$A$	$B$	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1



## Logikai műveletek igazságtáblája II

- 3 Vagy – OR (kétváltozós) igaz, ha valamelyik operandus igaz

$A$	$B$	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1





## Logikai műveletek igazságtáblája III

- 4 Kizáró vagy – XOR (kétváltozós) igaz, ha pontosan az egyik operandus igaz

$A$	$B$	$A \text{ xor } B$
0	0	0
0	1	1
1	0	1
1	1	0



# Aritmetikai műveletek

## 1 Egybites szorzás kettes számrendszerben

$A$	$B$	$A \times B$
0	0	0
0	1	0
1	0	0
1	1	1

## 2 Egybites összeadás kettes számrendszerben

$A$	$B$	$A + B$
0	0	00
0	1	01
1	0	01
1	1	10



## Kapcsolat a logikai és az aritmetikai műveletek között

- 1 A szorzás műveleti táblája megegyezik az És – AND művelet igazságtáblájával
- 2 Az összeadás felső bitjét az És – AND, az alsó bitjét a Kizáró vagy – XOR igazságtáblája szolgáltatja
- 3 A több-bites aritmetikai műveleteket hasonlóképpen összerakhatjuk a logikai műveletekből
- 4 A logikai műveletek könnyen megvalósíthatók elektronikai alkatrészekkel (ÉS-kapu, VAGY-kapu, stb.)
- 5 Így tudunk számolni elektronikus áramkörökkel
- 6 A processzor logikai kapuk hálózatából épül fel



# Outline

- 1 Bevezetés
- 2 Alapfogalmak
  - Kettes számrendszer
  - Logikai és aritmetikai műveletek
  - A rendszerszemlélet
- 3 A számítógép működése
  - Kódolás, kódrendszerek
  - Számok kódolása
- 4 Befejezés



# A rendszer fogalma

- Rendszernek nevezzük alkotóelemek és ezek kapcsolatainak olyan együttesét, amelyek az adott vizsgálat szempontjából összetartoznak
- A rendszerek kisebb alrendszerekből, és tovább nem bontható alapelemekből épülnek fel
- Alapelemnek nevezzük a rendszer azon alkotóelemeit, amelyeket az aktuális vizsgálatban nem bontunk tovább kisebb elemekre
- Példák: autó, számítógép, ember
- Vizsgáljuk meg ezeket a rendszereket!
- Rendszerszemlélet



# A rendszerek bonyolultsága

- Bonyolultság szempontjából két csoportba oszthatjuk a rendszereket:
  - ① egyszerű rendszer:
    - kisszámú alapelemből felépülő rendszer
    - könnyen áttekinthető, vizsgálható
  - ② összetett rendszer:
    - nagyszámú alkotóelemből és/vagy alrendszerekből felépülő rendszer
    - a rendszer vizsgálatához célszerű részrendszerekre bontani
- A mérnöki rendszerek vizsgálatának alapvető módszere a részekre bontás
- Minden rendszer vizsgálható ezzel a módszerrel?



# A rendszerek bonyolultsága

- A rendszer nem egyenlő a részeinek összegével!
- Minél összetettebb egy rendszer, annál nagyobb a különbség a kettő között
- Mi a különbség? - KAPCSOLATOK!!!
- Hogyan vizsgálhatók az összetett rendszerek?
  - Modellezés - Absztrakció
  - Kiválasztjuk a rendszer azon jellemzőit, amelyek a vizsgálat szempontjából fontosak
  - Felépítünk egy másik rendszert (modell), ami egyszerűbb a vizsgált rendszerénél, de a kiválasztott jellemzőkben megegyezik azzal
  - Ha jól választottuk ki a fontos jellemzőket, és jól építjük fel a modellt, akkor annak vizsgálatával fontos információkat szerezhetünk a vizsgált rendszerről



# A számítógép, mint rendszer

- A számítógép alkotóelemeit két nagy csoportba soroljuk:
- Hardver:
  - hardware: *"kemény áru"*
  - a számítógépet alkotó kézzel fogható alkatrészek összefoglaló neve
  - elektronikus áramkörök, mechanikus eszközök, kábelek, perifériák, stb.
  - önmagában működésképtelen
- Szoftver:
  - software: *"lágy áru"*
  - a számítógépen futó algoritmusok, programok és adatok összessége
  - a szoftver működteti a számítógép hardver eszközeit
- Ebben a tárgyban általában a szoftverrel fogunk foglalkozni





# Analóg-digitális technika

- **Analóg**nak nevezzük azokat az eszközöket, eljárásokat, amelyek folytonos mennyiségekkel dolgoznak
- **Digitális**nak nevezzük azokat az eszközöket, eljárásokat, amelyek az adatokat diszkrét (nem folytonos) értékekkel, véges sok számjeggyel közelítik
- **Digitális ábrázolás** valamely változó értékének diszkrét ábrázolása véges sok számjeggyel
- A természetben az adatok általában analóg formában vannak jelen, hőmérséklet, áramerősség, feszültség, színek (a fény hullámhossza)
- A számítógépben általában digitális adatokkal dolgozunk
- Szükség van az adatok átalakítására



# Adatok átalakítása

- ❶ Analóg-digitális (A/D) átalakítás, A/D konverter:
  - mintavételezés: a folytonos jelből véges sok helyen mintát veszünk
  - kvantálás: a minta értéktartományát diszkrét intervallumokra osztjuk és minden intervallumot egy kijelölt elemével reprezentálunk
- ❷ Digitális-analóg (D/A) átalakítás, D/A konverter:
  - a digitális jelből analóg (folytonos) jelet állít elő
- ❸ Hol használunk minden nap analóg-digitális átalakítást mindkét irányban?
  - Modem, kábelmodem
  - A telefon és a kábeltevé analóg jelet továbbít a két fél között



# Adatok átalakítása

- ❶ Analóg-digitális (A/D) átalakítás, A/D konverter:
  - mintavételezés: a folytonos jelből véges sok helyen mintát veszünk
  - kvantálás: a minta értéktartományát diszkrét intervallumokra osztjuk és minden intervallumot egy kijelölt elemével reprezentálunk
- ❷ Digitális-analóg (D/A) átalakítás, D/A konverter:
  - a digitális jelből analóg (folytonos) jelet állít elő
- ❸ Hol használunk minden nap analóg-digitális átalakítást mindkét irányban?
  - Modem, kábelmodem
  - A telefon és a kábeltevé analóg jelet továbbít a két fél között



# Outline

- 1 Bevezetés
- 2 Alapfogalmak
  - Kettes számrendszer
  - Logikai és aritmetikai műveletek
  - A rendszerszemlélet
- 3 A számítógép működése
  - Kódolás, kódrendszerek
  - Számok kódolása
- 4 Befejezés



# Kódolás, kódrendszerek

- Az adatok feldolgozásához szükség van arra, hogy a számítógép számára érthető formára hozzuk azokat
- Ezt a folyamatot nevezzük kódolásnak
- Általában olyan kódolásokat használunk, amikor az adat és a kódja között kölcsönösen egyértelmű megfeleltetés áll fenn
- Ismerünk olyan kódrendszert, ami nem kölcsönösen egyértelmű?
  - természetes nyelvek
  - természetes nyelvek fordítása
  - többek között ezért sem tudunk beszéddel kommunikálni a számítógépekkel (folytak erre kísérletek, de még messze vagyunk a sci-fi-kben látható eredményektől)
  - veszteséges tömörítés, pl. JPEG képformátum
- Milyen kódrendszereket ismerünk?



# Kódolás, kódrendszerek

- Az adatok feldolgozásához szükség van arra, hogy a számítógép számára érthető formára hozzuk azokat
- Ezt a folyamatot nevezzük kódolásnak
- Általában olyan kódolásokat használunk, amikor az adat és a kódja között kölcsönösen egyértelmű megfeleltetés áll fenn
- Ismerünk olyan kódrendszert, ami nem kölcsönösen egyértelmű?
  - természetes nyelvek
  - természetes nyelvek fordítása
  - többek között ezért sem tudunk beszéddel kommunikálni a számítógépekkel (folynak erre kísérletek, de még messze vagyunk a sci-fi-kben látható eredményektől)
  - veszteséges tömörítés, pl. JPEG képformátum
- Milyen kódrendszereket ismerünk?



# Elterjedt karakterkódolási rendszerek

## 1 ASCII:

- 8 bites kódrendszer
- $2^8 = 256$  jelet tud megkülönböztetni
- eredetileg 7 bites volt, az első 128 jel szabványos, egységes
- a második 128 karakterre több különböző kód kiosztás létezik

## 2 UNICODE:

- 16 bites kódrendszer
- $2^{16} = 65536$  jelet tud megkülönböztetni
- „minden” nyelv összes jele elfér benne

## 3 UTF8:

- Általános kódolási rendszer
- Minden karaktert képes kódolni 1-4 byte hosszan
- Változó hosszúságú kódokat használ
- A kódolás leírása: <http://en.wikipedia.org/wiki/UTF-8>
- Az UTF-8 kódok: <http://www.utf8-chartable.de/>



# Karakterosztályok

- alfabetikus: az angol ábécé kis és nagybetűi (a, b, c, ..., A, B, C, ...)
- numerikus: számjegyek: (0, 1, 2, ..., 9)
- alfanumerikus: alfabetikus vagy numerikus, időnként beleértünk néhány írásjelet is, pl. \_
- egyéb jelek: írásjelek, nemzeti karakterek, stb.





# Outline

- 1 Bevezetés
- 2 Alapfogalmak
  - Kettes számrendszer
  - Logikai és aritmetikai műveletek
  - A rendszerszemlélet
- 3 A számítógép működése
  - Kódolás, kódrendszerek
  - Számok kódolása
- 4 Befejezés



# Számok kódolása

- Kettes számrendszer (bináris)
- Tizenhatos számrendszer (hexadecimális)
- Átváltás kettes, tízes és tizenhatos számrendszer között
- BCD (Binary Coded Decimal/binárisan kódolt decimális)



# Kettes számrendszer használata

128	64	32	16	8	4	2	1		10-es számrendszer
0	1	0	0	0	1	0	0	=	$1 \times 64 + 1 \times 4 = 68_{(10)}$
0	0	0	0	0	0	0	1	=	$1 \times 1 = 1_{(10)}$
0	0	0	0	0	0	1	0	=	$1 \times 2 = 2_{(10)}$
0	0	0	0	0	0	1	1	=	$1 \times 2 + 1 \times 1 = 3_{(10)}$
0	0	0	0	0	1	0	0	=	$1 \times 4 = 4_{(10)}$
...									
1	1	1	1	1	1	0	0	=	$252_{(10)}$
1	1	1	1	1	1	0	1	=	$253_{(10)}$
1	1	1	1	1	1	1	0	=	$254_{(10)}$
1	1	1	1	1	1	1	1	=	$255_{(10)}$



## Egész számok átalakítása tízes-ről kettes számrendszerre

- az egész számot addig osztjuk 2-vel, amíg a hányados 0 nem lesz
- a hányadost írjuk a bal oldali oszlopba
- a maradékot a jobb oldaliba
- az eredményt a jobb oldali oszlopban kapjuk alulról felfelé

hányados	maradék
74	$\div 2$
37	0
18	1
9	0
4	1
2	0
1	0
0	1

$$74_{(10)} = 1001010_{(2)}$$



## A törtrész átalakítása

- a törtrészt addig szorozzuk 2-vel, amíg a szorzat 1,0 nem lesz
- ha ez nem következik be, akkor a kívánt pontosságig folytatjuk a szorzást
- a szorzat egészrészét írjuk a bal oldali oszlopba
- a törtrészt a jobb oldaliba
- az eredményt a bal oldali oszlopban kapjuk felülről lefelé

egészrész	törtrész
$\times 2$	0,125
0	0,25
0	0,5
1	0

$$0,125_{(10)} = 0,001_{(2)}$$



# Átalakítás bináris, oktális és hexadecimális számrendszer között

- korábban láttuk, hogy a számítógép számára a kettes számrendszer a legalkalmasabb
- a bináris számok leírása nagyon hosszú, ezért körülményesen használhatók
- az átalakítás tízes számrendszerről elég körülményes
- ezen a problémán segít a 8-as (oktális) és a 16-os (hexadecimális) számrendszer használata
- Miért?
  - vegyük észre, hogy  $8_{(10)} = 1000_{(2)}$  és  $16_{(10)} = 10000_{(2)}$ , azaz a 8 és a 16 „kerek” számok kettes számrendszerben, ahogyan a tízes számrendszerben a száz, az ezer, vagy a millió
  - Hogyan írnánk át egy számot tízes számrendszerből ezresbe?



## Ezres tagolás

- tízes számrendszerben így szoktuk leírni a nagy számokat:

123 456 789

- ha a háromjegyű számcsoportokat egy-egy számjegynek tekintjük, akkor ezzel átírtuk a számot ezres számrendszerbe:

$[123] [456] [789]_{(1000)}$

- ha ugyanezt meg tesszük a bináris számokkal, akkor ugyanilyen egyszerűen átírhatunk számokat oktális, vagy hexadecimális számrendszerbe



## Bináris-oktális átalakítás

- Nyolcas számrendszerben 8 számjegyre van szükségünk, ezért használhatjuk a megszokott számjegyeket 0-tól 7-ig
- például:
  - decimális:  $42_{(10)}$
  - bináris:  $101\ 010_{(2)}$
  - oktális:  $52_{(8)}$

oktális számjegy	bináris szám
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111





## Bináris-hexadecimális átalakítás

- Tizenhatos számrendszerben 16 számjegyre van szükség 0-tól 15-ig
- 10-től 15-ig az ábécé első 6 betűjét használjuk (A-F)

hexa szjegy	bináris szám
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

hexa szjegy	bináris szám
8	1000
9	1001
A (10)	1010
B (11)	1011
C (12)	1100
D (13)	1101
E (14)	1110
F (15)	1111



## Bináris-hexadecimális átalakítás

decimális	$42_{(10)}$
bináris	$00101010_{(2)}$
bináris tagolva	$0010 \quad 1010_{(2)}$
hexadecimális tagolva	$2 \quad A_{(16)}$
hexadecimális	$2A_{(16)}$

- Melyiket használjuk az oktális, vagy a hexadecimális számrendszert?
- A ma használatos számítógépek 8 bites szervezésűek, ezért a hexadecimális írásmód a legcélszerűbb
- Egy bájtnyi (8 bit) adatot két hexadecimális számjeggyel írunk le



# BCD - Binary Coded Decimal

- Binárisan kódolt decimális számaábrázolás
- az előző ötletet alkalmazhatjuk bináris és decimális számrendszerek között is

decimális	1789 <sub>(10)</sub>			
decimális tagolva	1	7	8	9 <sub>(10)</sub>
bináris tagolva	0001	0111	1000	1001 <sub>(2)</sub>

- ez egy kicsit pazarló ábrázolás, mert a 4 biten elérhető 16 lehetőség közül csak 10-et használunk fel
- Pakolt BCD kód esetén két-két BCD számjegy 1 bájtba kerül
- Zónázott BCD kód esetén minden számjegyet külön bájtban tárolunk, ez tovább csökkenti a kód tömörségét, de egyszerűbbé teszi az aritmetikát



## Negatív számok kezelése

- az eddig tárgyalt módszerekkel tetszőleges pozitív egész számokat tudunk kezelni
- a használt bitek száma meghatározza az ábrázolható számok méretét, de bármekkora számokat megvalósíthatunk
- Mit tegyünk a negatív számokkal?
  - 1 minden számhoz külön tároljuk az előjelet
    - minden alkalommal, amikor használni akarjuk a számot, meg kell vizsgálni az előjelet, és attól függően kell használni az értéket
  - 2 építsük be az előjelet a számok ábrázolásába, lehetőleg úgy, hogy jól működjön az aritmetika
    - a számok ábrázolásához felhasználható bitek közül egyet lefoglalunk (általában a legmagasabb helyiértékűt), és ezen tároljuk az előjelet, 0: pozitív, 1: negatív
    - az ábrázolható számok abszolútértékét ezzel megfeleztük
    - pl.  $0, \dots, +255$  helyett  $-128, \dots, +127$



# Komplementképzés

- ezzel a módszerrel úgy ábrázoljuk a negatív számokat, hogy azokkal ugyanúgy végezhesünk műveleteket, mint a pozitív számokkal
- *komplement*: az a szám, amelyik az ábrázolható legnagyobb számnál eggyel nagyobbra egészíti ki az adott számot
- jele:  $\bar{x}$ 
  - ha tízes számrendszerben háromjegyű számokkal dolgozunk
  - a legnagyobb ábrázolható szám: 999
  - egy  $a$  szám komplemente:  $\bar{a} = 1000 - a$
  - pl.  $\overline{196} = 1000 - 196 = 804$



# Kettes komplement

- Kettes számrendszerben a következőképpen képezhetjük a kettes komplementet:
  - 1 a legkisebb helyiértéktől indulva, végigmegyünk a biteken
  - 2 amíg 0 biteket találunk, azokat változatlanul leírjuk
  - 3 amikor 1-est találunk, azt még szintén leírjuk, de a további biteket invertáljuk
- *Egyes komplement*: a bináris szám bitenkénti negáltja, azaz minden bitjét az ellenkezőjére állítjuk
- ha az egyes komplementhez egyet hozzáadunk, megkapjuk a kettes komplementet



## Példa komplementképzésre

- Például 8 bites számokkal dolgozunk,

$$11111111_{(2)} + 1 = 100000000_{(2)} = 256_{(10)}$$

-ra kell kiegészíteni

$x$	$10111000_{(2)}$	$184_{(10)}$
$\bar{x}$	$01001000_{(2)}$	$72_{(10)}$
$x$ egyes komplemente	$01000111_{(2)}$	$71_{(10)}$
$x$ egyes kompl. +1	$01001000_{(2)}$	$72_{(10)}$
$x + \bar{x}$	$100000000_{(2)}$	$256_{(10)}$



# Előjeles számok ábrázolása kettes komplementum kódban

- vegyünk egy kétbájtos számot
- a legmagasabb helyiértékű biten (16.) tároljuk az előjelet, 0: pozitív, 1: negatív
- a legnagyobb ábrázolható szám:  $N_{max} = 2^{15} - 1 = 32767$

16.	15.	14.	...	6.	5.	4.	3.	2.	1.	
0	0	0	...	1	1	0	1	0	0	= 52 <sub>(10)</sub>
1	1	1	...	0	0	1	0	1	1	egyes kompl.
0	0	0	...	0	0	0	0	0	1	+1
1	1	1	...	0	0	1	1	0	0	-52 kódolva





# Számolás kettes komplementes kódban

16.	15.	14.	...	6.	5.	4.	3.	2.	1.	
1	1	1	...	0	0	1	1	0	0	$-52_{(10)}$
0	0	0	...	1	1	0	1	1	1	$+55_{(10)}$
0	0	0	...	0	0	0	0	1	1	$-52 + 55 = 3$



# Összefoglalás

- 1 Bevezetés
- 2 Alapfogalmak
  - Kettes számrendszer
  - Logikai és aritmetikai műveletek
  - A rendszerszemlélet
- 3 A számítógép működése
  - Kódolás, kódrendszerek
  - Számok kódolása
- 4 Befejezés



# Befejezés

Köszönöm a figyelmet!

