Gyakorló feladatok

Számrendszerek:

Feladat: Ábrázold kettes számrendszerbe a 639_{10}, 16-os számrendszerbe a 311_{10}, 8-as számrendszerbe a 483_{10} számot!

Megoldás:

/2	Maradék
639	1
319	1
159	1
79	1
39	1
19	1
9	1
4	0
2	0
1	1

/16	Maradék
311	7
19	3
1	1

/8	Maradék
483	3
60	4
7	7

$639_{10} = 1001111111_2$ $311_{10} = 137_{16}$ $483_{10} = 743_8$

Feladat: Ábrázold kettes számrendszerben a 3.125_{10} és a 12.5_{10} törteket!

Megoldás:

/2	Maradék	Egész	*2
3	1		.125
1	1	0	.25
		0	.5
		1	.0

$3.125_{10} = 11.001_2$

/2	Maradék	Egész	*2
12	0		.5
6	0	1	.0
3	1		
1	1		

$12.5_{10} = 1100.1_2$

Feladat: 2-es, 8-as és 16-os számrendszerben levő számok átalakítása decimálissá.

Megoldás: A decimális értéket úgy számoljuk ki, hogy összeadjuk a szám alaki értékének helyi értékével képzett szorzatát.

Kettes számrendszer

8	4	2	1	Decimális érték
1	0	1	1	11

8-as számrendszer

512	64	8	1	Decimális érték
6	3	7	5	3325

16-os számrendszer

4096	256	16	1	Decimális érték
A	5	2	B	42283

Feladat: Ábrázold a következő számok előjeles abszolút értékét: 25, -46, -18, 87 !

Megoldás:

25 = 00011001	46 = 00101110	18 = 00010010	87 = 01010111
25 = 00011001	-46 = 10101110	-18 = 10010010	87 = 01010111

Feladat: Ábrázold a következő számok egyes komplemensét: 25, -46, -18, 87 !

Megoldás:

25 = 00011001 25 = 00011001	46 = 00101110 -46 = 11010001	18 = 00010010 -18 = 11101101	87 = 01010111 87 = 01010111

Feladat: Ábrázold a következő számok kettes komplemensét: 25, -46, -18, 87 !

Megoldás:

25 = 00011001 25 = 00011001	46 = 00101110 -46 = 11010010	18 = 00010010 -18 = 11101110	87 = 01010111 87 = 01010111

Feladat: Ábrázold a következő számok 128-többletes értékét: 25, -46, -18, 87 !

Megoldás:

25 = 00011001 128+25 = 153 = 10011001	46 = 00101110 128-46 = 82 = 01010010	18 = 00010010 128-18 = 110 = 01101110	87 = 01010111 128+87 = 215 = 11010111

Feladat: Az alábbi bitsorozatot Hamming-kód ellenőrzi. Melyik bit a hibás? (A bitsorozat tartalmazza a paritásbiteket is. A paritásbit 1, ha az ellenőrzött 1-esek száma páratlan.)
Bitsorozat: 111110011101101101

Megoldás:
A paritásbitek a 2-hatvány pozíciókon helyezkednek el.

1111100111011011101
pozíciók: 123456789012345678

A paritásbitek a következő pozíciókat ellenőrzik:
1. bit: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, ...
2. bit: 2, 3, 6, 7, 10, 11, 14, 15, 18, 19, ...
4. bit: 4, 5, 6, 7, 12, 13, 14, 15, ...
8. bit: 8, 9, 10, 11, 12, 13, 14, 15, ...
16. bit: 16, 17, 18, 19, ...

Ellenőrizzük le, hogy mely paritásbitek mutatnak rossz értéket!
Az 1. bit jó. A 2. bit rossz, ezért a 2, 3, 6, 7, 10, 11, 14, 15, 18 bit valamelyike rossz. A 4. bit rossz, ezért a 4, 5, 6, 7, 12, 13, 14, 15 bit valamelyike rossz. A 8. bit jó. A 16. bit jó.
A következő pozíciókon lehet a hibás bit: 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15.
Néhányat ezek közül kizárhatunk: Az 1. és a 8. bit jó volt, ezért az általuk ellenőrzött pozíciókon biztos, hogy nem romlott el az érték. 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15. Maradt a 2, 4 és a 6.
Ezek közül a 2. és a 4. bit is rossz, de csak egy bit romlott el, vagyis valamelyik jó. Csak a 6. bit az, amelyiket mindkét bit ellenőrzi, ezért ez volt a hibás bit.

Feladat: Az alábbi bitsorozatot Hamming-kód ellenőrzi. Melyik bit a hibás? (A bitsorozat tartalmazza a paritásbiteket is. A paritásbit 1, ha az ellenőrzött 1-esek száma páratlan.)
Bitsorozat: 00111111110110011

Megoldás:
A paritásbitek a 2-hatvány pozíciókon helyezkednek el.

00111111110110011
pozíciók: 12345678901234567

A paritásbitek a következő pozíciókat ellenőrzik:
1. bit: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, ...
2. bit: 2, 3, 6, 7, 10, 11, 14, 15, 18, 19, ...
4. bit: 4, 5, 6, 7, 12, 13, 14, 15, ...
8. bit: 8, 9, 10, 11, 12, 13, 14, 15, ...
16. bit: 16, 17, 18, 19, ...

Ellenőrizzük le, hogy mely paritásbitek mutatnak rossz értéket!
Az 1. bit jó. A 2. bit is jó. A 4. bit is jó. A 8. bit hibás, ezért a 8, 9, 10, 11, 12, 13, 14, 15 pozíciókon lévő bitek valamelyike rossz. A 16. bit jó.
A lehetséges hibás bit a 8, 9, 10, 11, 12, 13, 14, 15 pozíciók valamelyikén van, de néhányat ki lehet zárni. Mivel az 1., a 2. és a 4. bit jó, ezért a 9, 10, 11, 12, 13, 14, 15 bitek nem lehetnek hibásak. Így egyedül a 8. bit romolhatott el.

Feladat: Az alábbi bitsorozatot Hamming-kód ellenőrzi. Melyik bit a hibás? (A bitsorozat tartalmazza a paritásbiteket is. A paritásbit 1, ha az ellenőrzött 1-esek száma páratlan.)
Bitsorozat: 11000111011001111

Megoldás:
A paritásbitek a 2-hatvány pozíciókon helyezkednek el.

11000111011001111
pozíciók: 12345678901234567

A paritásbitek a következő pozíciókat ellenőrzik:
1. bit: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, ...
2. bit: 2, 3, 6, 7, 10, 11, 14, 15, 18, 19, ...
4. bit: 4, 5, 6, 7, 12, 13, 14, 15, ...
8. bit: 8, 9, 10, 11, 12, 13, 14, 15, ...
16. bit: 16, 17, 18, 19, ...

Ellenőrizzük le, hogy mely paritásbitek mutatnak rossz értéket!
Az 1. bit hibás, ezért a 1, 3, 5, 7, 9, 11, 13, 15, 17 pozíciók valamelyike hibás. A második is hibás, ezért a 2, 3, 6, 7, 10, 11, 14, 15 pozíciók valamelyikén hibás bit van. A 4. bit helyes, ezért az általa ellenőrzött pozíciókon nincs hiba. A 8. bit hibás, ezért a 8, 9, 10, 11, 12, 13, 14, 15 pozíciókon lévő bitek egyike hibás. A 16. bit jó, ezért a 16 és 17. bitek nem romlottak el. Mivel több paritásbit is hibát jelzet, így képezzük a lehetséges hibás bitek metszetét, vagyis azokat a pozíciókat, amelyeket több bit is ellenőriz, és hibásnak sejt. Ezek a 11, 15 bitpozíciók. Így leszűkült a kör. A 15. bitet ellenőrzi a 4. bit, ami jó, ezért ezt kizárhatjuk. Így maradt a 11. bit. Ez romlott el.

Digitális elektronika

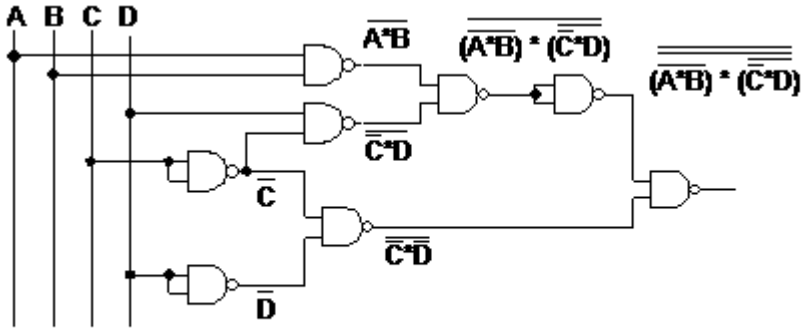
Feladat: Valósítsd meg az F kimeneti függvényt kétbemenetű NAND kapukkal!

$$F=(A\wedge B)\vee(\overline{C}\wedge D)\vee(\overline{A\vee C})$$

Megoldás:

$$F=(A\wedge B)\vee(\overline{C}\wedge D)\vee(\overline{A\vee C})=\overline{\overline{(\overline{A\wedge B})\vee(\overline{C\wedge D})}\vee(\overline{A\vee C})}=$$
$$=\overline{\overline{(\overline{A\wedge B})\wedge(\overline{C\wedge D})}\vee(\overline{A\vee C})}=\overline{(\overline{A\wedge B})\wedge(\overline{C\wedge D})\vee(\overline{A\vee C})}=$$
$$=\overline{\overline{(\overline{A\wedge B})\wedge\overline{C}\wedge D\wedge(\overline{A}\wedge\overline{C})}}$$

Mivel a három kifejezés VAGY kapcsolattal áll, ezért először vehetjük az első két zárójeles kifejezést. Először a zárójeles kifejezéseket alakítjuk át, majd a köztük lévő VAGY kapcsolatra alkalmazzuk a De Morgan azonosságokat. A De Morgan azonosságokat a legkönnyebb úgy végrehajtani, ha először kétszeresen negáljuk az egész műveletet, majd, alkalmazzuk az átalakítást. Ezután következhet a harmadik zárójeles tag átalakítása, majd az utolsó külső VAGY kapcsolatot alakítjuk át. Mivel jelen esetben a zárójeles tagok sorrendje felcserélhető, ezért másfajta csoportosítás/átalakítás is helyes lehet.

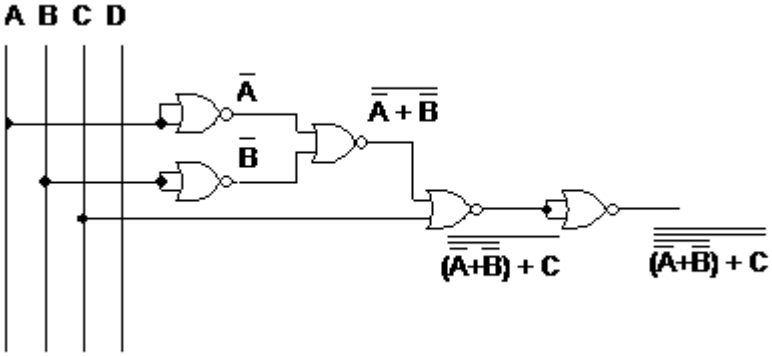


Feladat: Valósítsd meg az F kimeneti függvényt kétbemenetű NOR kapukkal!

$$F=(A\wedge B)\vee C$$

Megoldás:

F = $(A\wedge B)\vee C=(\overline{\overline{A\wedge B}})\vee C=(\overline{\overline{A}\vee\overline{B}})\vee C=\overline{\overline{(\overline{A}\vee\overline{B})\vee C}}$



Assembly programozás:

Feladat: Írj eljárást, amely egy hexadecimális számot kiír a képernyőre.
Megoldás:

```
hexat_kiir proc
  push ax        ;
  push bx        ; lementjuk az regiszterek tartalmat

  mov al, bl     ; AL tartalmat atmasoljuk BL-be

                 ; egy szamjegyet ugy irunk ki, hogy
  cmp bl, 9      ; ha a szamjegy
  jle szamjegy   ; kisebb-egyenlo 9, akkor ugras "szamjegy"-re
  betu:          ; kulonben betut ir ki
    mov al, bl   ; AL-be atmasolom a szamot
    sub al, 10   ; kivonok belole 10-et
    add al, 'A'  ; es ehhez hozzaadom az 'A' karakterkodjat
    mov ah, 14   ; BIOS parameterezes
    int 010H     ; kiiratas
    jmp hexa_vege

  szamjegy:
    mov al, '0'    ; a 0 karakterkodjahoz hozzaadjuk a szamjegyet
    add al, bl
    mov ah, 14
    int 010H
  hexa_vege: pop bx   ; visszatoltjuk a lementett tartalmakat
             pop ax
             ret
hexat_kiir endp
```

Magyarázat: Az eljárások elején érdemes lementeni a használandó regiszterek tartalmát. Egy hexadecimális számjegy vagy betű vagy szám lehet. Először meg kell nézni, hogy az értéke, az kisebb vagy egyenlő-e mint 9, ha igen, akkor számjegyként kell kiírni, ha nem, akkor betűként. Ha számjegyként írjuk ki, akkor a regiszter tartalmát hozzá kell adni a '0' karakterkódjához és ezt a karaktert kiírni a képernyőre. Ha betű, akkor a számból ki kell vonni 10-et, és ezt hozzá kell adni az 'A' betű karakterkódjához, majd a kapott karaktert kiíratni. Az eljárás végén a regiszterek tartalmát visszamentjük.

Feladat: Írj kódrészletet, amely kiírja 6! (6 faktoriális) értékét!
Megoldás:

```
  mov cx, 6
  mov bx, 1
  mov ax, 1
  ciklus: cwd                  ; AX helyett (DX:AX)-ben taroljuk a szamot
          imul bx              ; (DX:AX)-et megszorozzuk BX aktualis
                               ; ertekevel
          inc bx               ; megnoveljuk BX erteket eggyel
          dec cx               ; csokkentjuk CX-et
          cmp cx, 0            ; CX == 0 ?
          jz vege              ; ha igen, akkor vege
          jnz ciklus           ; ha nem, akkor ugrunk ciklusra
  vege:
  call tobbjegyu_szamot_kiir  ; kiirja az AX-ben levo szamot
```

Magyarázat: A ciklusszámlálóba (CX) eltároljuk azt a számot, amelynek a faktoriálisát keressük. Egy másik regisztert (BX) arra használunk, hogy ebben tároljuk azt az elemet, amellyel megszorozzuk az eddigi részeredményt. A részeredményt egy külön regiszterben (AX) tároljuk. A számok faktoriálisa nagy lehet, ezért nem 16 biten, hanem 32 biten (DX:AX) tároljuk a számot. A részeredmény így (DX:AX)-ben keletkezik, ezt kell megszorozni, a soron következő számmal. Ha a ciklusszámláló 0-ra csökken, akkor vége a számításnak.

Feladat: Írj kódrészletet, amely egy megadott kisbetűs szöveget alakítsunk át nagybetűssé! Feltehetjük, hogy (DS:SI) az átalakítandó szöveg elején áll. A szöveg végét a $ karakter jelzi.

Megoldás:

```
    mov si, offset msg
    cld
    ciklus:
            lodsb           ; beolvasunk egy karaktert
            cmp al, '$'     ; megnézzuk, hogy a vegjel-e
            je vege         ; ha igen, akkor a szo vegere ertunk es ugrunk a
                            ;  "vege" cimkere
            mov bl, al      ; ha nem akkor atmasoljuk a karaktert bl-be
            sub bl, 'a'     ; megnezzuk, hogy hanyadik betu az abc-ben
            mov al, 'A'     ; AL-be beirjuk az 'A' karakter kodjat
            add al, bl      ; es hozzaadjuk a beolvasott betu sorszamat
                            ; igy a nagybetus valtozatat kapjuk meg a beolvasott
                            ; betunek
            call kiir       ; vegul kiirjuk
            jmp ciklus
```

Magyarázat: A megoldás eléggé egyszerű, ha kihasználjuk, hogy az ábécé betűi egymást követő karakterkódok. Nem kell mást tenni, mint beolvasni egy karaktert, megnézni hogy az a végjel-e, ha igen, akkor a szó végére értünk, ha nem, akkor úgy beolvasott karakter kódjából kivonjuk a 'a' karakter kódját. Ez egy szám lesz, amely megmondja, hogy a beolvasott betű hányadik helyen áll az ábécében. Ezt a számot hozzá kell adni a 'A' karakterkódjához, így megkapjuk a beolvasott karakter nagybetűs változatát. Ez a megoldás csak az angol ábécé betűire alkalmazható.