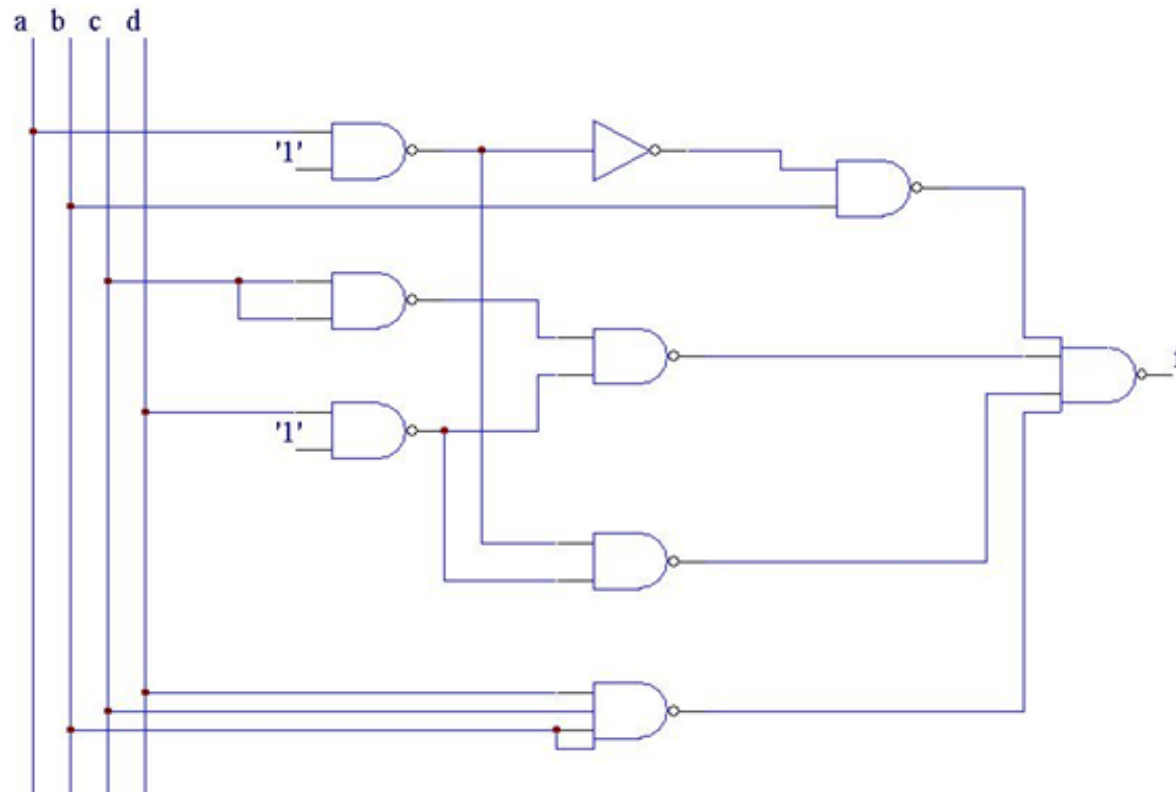


# **APLICAȚII – zona tematică 5**

## **-TST-ID-**

1. Să fie reprezentată funcția  $f$  din schema următoare cu numărul minim de porți logice SI-NU (2 sau 4 intrări):

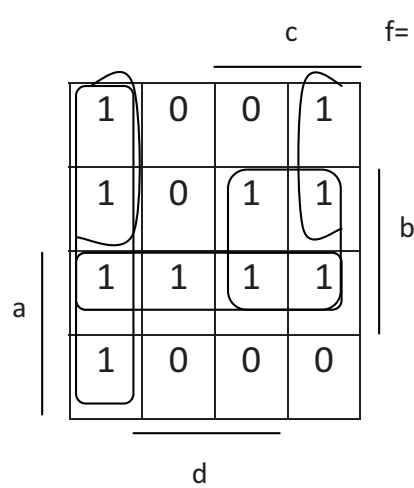


Soluție:

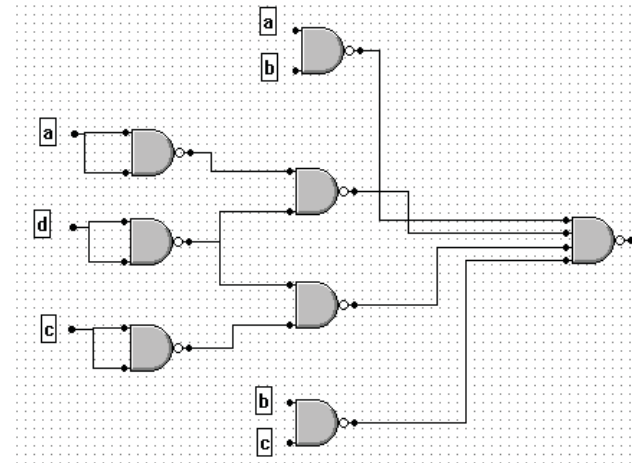
$$f = \overline{a \cdot 1 \cdot b \cdot c \cdot c \cdot d \cdot 1} + \overline{a \cdot 1 \cdot d \cdot 1 \cdot b \cdot b \cdot c \cdot d} = \overline{a \cdot b \cdot c \cdot d} + \overline{a \cdot d \cdot b \cdot c \cdot d}$$

Aplicând relațiile lui De Morgan obținem:  $f = a \cdot b + \overline{c} \cdot \overline{d} + \overline{a} \cdot \overline{d} + b \cdot c \cdot d$

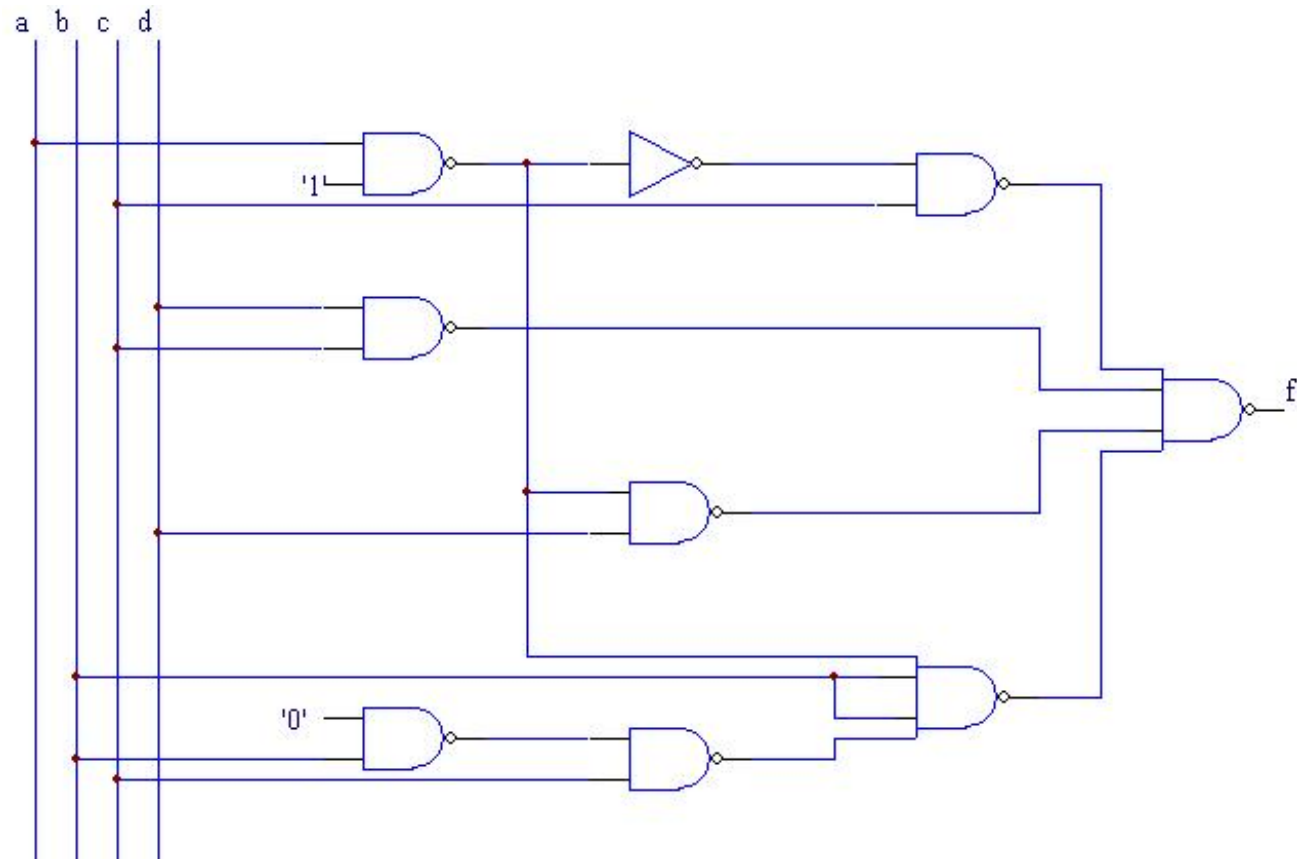
Pentru a obține forma minimă folosim diagrame Veitch-Karnaugh:



c  $f = a.b + \overline{a}.\overline{d} + \overline{c}.\overline{d} + b.c$  Aplicând din nou teoremele lui De Morgan obținem:



2. Să fie reprezentată funcția  $f$  din figura următoare folosind un circuit multiplexor cu 2 intrări de adresă, 4 de date și porți logice:



Soluție:

$$f = \overline{a \cdot 1 \cdot c \cdot c \cdot d} \cdot \overline{a \cdot 1 \cdot d} \cdot \overline{a \cdot 1 \cdot b \cdot b \cdot 0 \cdot c} = \overline{a \cdot c \cdot c \cdot d} \cdot \overline{a \cdot d} \cdot \overline{a \cdot b \cdot c}$$

Aplicând relațiile lui De Morgan obținem:  $f = a \cdot c + c \cdot d + \bar{a} \cdot d + \bar{a} \cdot b \cdot \bar{c}$

Pentru a reprezenta funcția cu multiplexor o scriem în formă tabelară. Pentru a scrie funcția în tabel folosim diagrame Veitch-Karnaugh:

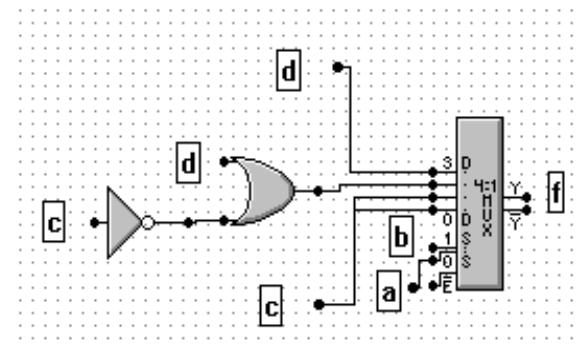
|   |   |   |   |   |
|---|---|---|---|---|
|   |   | c |   | b |
| a | 0 | 1 | 1 | 0 |
|   | 1 | 1 | 1 | 0 |
|   | 0 | 0 | 1 | 1 |
|   | 0 | 0 | 1 | 1 |
|   |   | d |   |   |

| a | b | c | d | f |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Vom pune la intrările de adresă ale multiplexorului variabilele a și b. Variabilele c și d sunt puse la intrările de date în funcție de valorile funcției. În fapt, fiecare intrare de date devine o funcție de cele 2 variabile c și d. obținem expresiile:

$$D_{00} = d, D_{01} = \bar{c} + d, D_{02} = D_{03} = c$$

O schemă posibilă este reprezentată în figura următoare (se acceptă și reprezentare simplificată pentru circuitul multiplexor cu punctarea doar a intrărilor și ieșirilor folosite):



1. Folosind un numărător sincron, binar, cu ștergere sincronă și porți convenabil alese, să se realizeze un numărător modulo 11 având secvența de numărare: 0, 1, 2, ..., 10, 0, 1, 2, ... .

*Soluție:* Schema numărătorului modulo 11 este prezentată în figura 1. Pentru obținerea secvenței 0, 1, 2, ..., 10, 0, 1, ... se utilizează o poartă ȘI-NU, cu două intrări, care detectează starea 10 ( $Q_3Q_2Q_1Q_0 = 1010$ ) și, prin activarea intrării  $\overline{CLR}$ , determină ca următoarea stare să fie 0000. Deoarece ștergerea se face sincron, acest numărător nu necesită un circuit suplimentar pentru memorarea impulsului de ștergere.

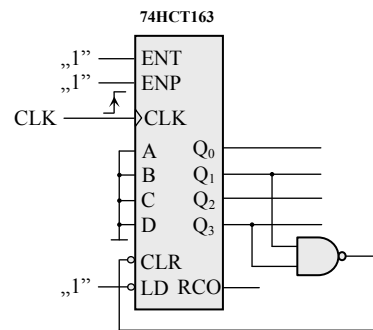


Figura 1. Schema numărătorului sincron modulo 11 cu secvența de numărare 0, 1, 2, ..., 10, 0, 1, ...

2. Folosind memorii EPROM de tip 27C256 (32k x 8 biți) și un număr minim de porți logice, să se obțină o memorie de 64k x 8 biți.

*Soluție:* a). Numărul necesar de circuite 27C256 este:

$$N = \frac{64k \times 8 \text{ biți}}{32k \times 8 \text{ biți}} = 2.$$

b). Memoria de 32k are  $2^5 \cdot 2^{10} = 2^{15}$  locații de memorie care pot fi accesate utilizând 15 linii de adresă ( $A_0, \dots, A_{14}$ ).

Memoria de 64k are  $2^6 \cdot 2^{10} = 2^{16}$  locații de memorie, adică 16 linii de adresă.

Adresa suplimentară,  $A_{15}$ , se folosește pentru validarea celor două memorii conform tabelului 1.

Tabelul 1. Tabelul de validare a memoriei de 64k x 8 biți.

| $A_{15}$ | $A_{14} - A_0$ | Memoria validată | Condiția de validare |                   |
|----------|----------------|------------------|----------------------|-------------------|
|          |                |                  | $\overline{CS}_0$    | $\overline{CS}_1$ |
| 0        | X.....X        | 0                | 0                    | 1                 |
| 1        | X.....X        | 1                | 1                    | 0                 |

Schema memoriei de 64k x 8 biți este prezentată în figura 2.

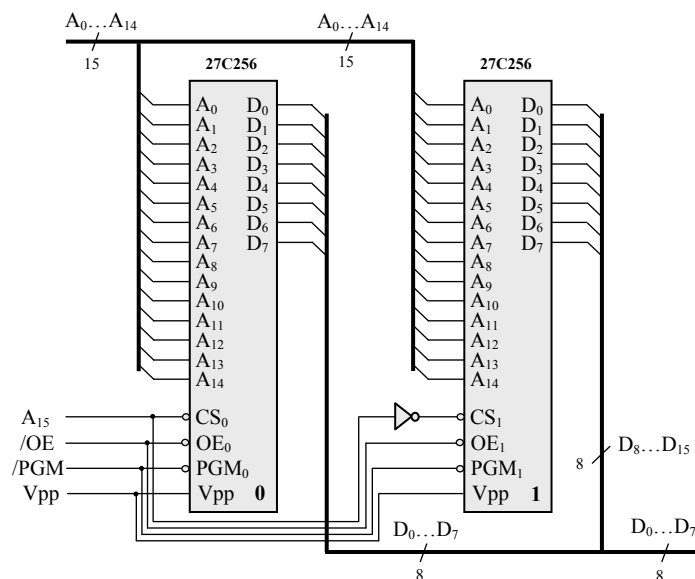
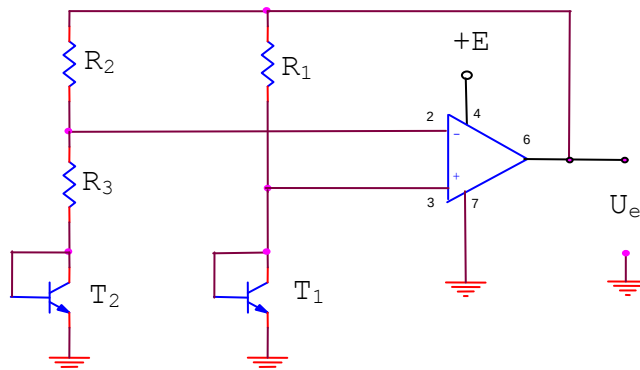


Figura 2. Memorie de capacitate 64k x 8 biți.

**Problema 1.**

- a) Să se demonstreze faptul că circuitul din figură poate fi o sursă de tensiune de referință de tip “band gap” ( $T_1$  și  $T_2$  fiind identice).
- b) Să se determine valoarea rezistenței  $R_3$  în acest caz, având  $R_1=10k\Omega$ ,  $R_2=27,2k\Omega$ .
- c) Pentru  $U_{BE} \approx 0,6V$  să se calculeze valorile curenților de colector ai tranzistoarelor, neglijând pe  $I_B$ .

**Soluție**

Se pot scrie ecuațiile

$$U_{BE1} - U_{BE2} = I_{C2} R_3 \quad (1)$$

$$I_{C2} R_2 = I_{C1} R_1 \quad \text{sau} \quad \frac{I_{C1}}{I_{C2}} = \frac{R_2}{R_1} \quad \text{deci} \quad I_{C2} = I_{C1} \frac{R_1}{R_2} \quad (2)$$

Din (1) și (2) rezultă  $U_T \ln \frac{I_{C1}}{I_{C2}} = I_{C1} \frac{R_1 R_3}{R_2}$  sau  $U_T \ln \frac{R_2}{R_1} = I_{C1} \frac{R_1 R_3}{R_2}$  deci

$$I_{C1} = \frac{R_2}{R_1 R_3} U_T \ln \frac{R_2}{R_1}$$

Tensiunea de ieșire se scrie acum

$$U_e = I_{C1} R_1 + U_{BE1} = \frac{R_2}{R_1 R_3} R_1 U_T \ln \frac{R_2}{R_1} + U_{BE1} \quad (3)$$

Se constată că  $U_e$  are forma:



$$U_e = U_{BE1} + \left( \frac{R_2 \ln \frac{R_2}{R_1}}{R_3} \right) U_T = U_{BE1} + N \cdot U_T$$

unde  $N = \left( \frac{R_2 \ln \frac{R_2}{R_1}}{R_3} \right)$  este o constantă realizabilă cu precizie datorită rapoartelor de rezistențe. Se știe că este o sursă de tip “band gap” și că pentru  $N=23$  apare compensarea termică a lui  $U_e$  și că  $U_e \cong 1,2V$

Calculul curenților:

$$I_{C1} R_1 \cong 0.6V \quad \text{deci} \quad I_{C1} = \frac{0.6}{R_1} = \frac{0.6}{10} = 0.06 \text{ mA} = 60 \mu A$$

$$I_{C2} = I_{C1} \frac{R_1}{R_2} = 60 \frac{10}{27.2} = 22.1 \mu A$$

Pentru compensare termică:  $N=23$  și pentru  $R_2=27,1k\Omega$  și  $R_1=10k\Omega$

$$\frac{R_2 \ln \frac{R_2}{R_1}}{R_3} = 23 \quad \frac{R_2}{R_3} = 23 \quad \text{deci} \quad R_3 = \frac{R_2}{23} = 1.178 K\Omega$$

### Problema 2.

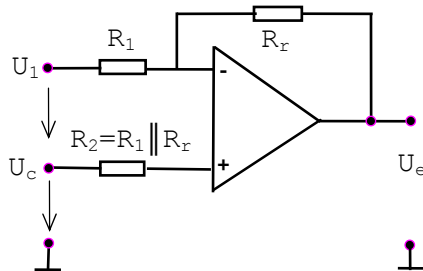
**5.15** Să se dimensioneze rezistențele amplificatorului operațional și tensiunea  $U_1$  astfel încât, dispunând de o tensiune de intrare  $U_c = -3 \dots +3V$ , să se obțină  $U_e = 0 \dots 12V$ . Tensiunea fixă se va realiza apoi cu ajutorul uneia din sursele de alimentare  $\pm E = \pm 5V$  (bine stabilizate) ale amplificatorului. Rezistența de la intrarea inversoare se va adopta de  $5k\Omega$ .

### Soluție

Tensiunile  $U_c$  și  $U_e$  au același sens de variație deci trebuie utilizat un amplificator neinversor. Tensiunea variabilă  $U_c$  se aplică la intrarea neinversoare iar tensiunea fixă o considerăm la cealaltă intrare pentru simplitate.

Se verifică dacă amplificarea necesară este mai mare decât 1 (pentru că altfel la intrarea + este necesară divizarea tensiunii  $U_c$  întrucât pentru această întare  $A_{ur+} \geq 1$ ).

Astfel



$$A_{ur+} = \frac{\Delta U_e}{\Delta U_c} = \frac{U_{emax} - U_{emin}}{U_{cmax} - U_{cmin}} = \frac{12 - 0}{3 - (-3)} = 2 > 1$$

prin urmare nu este necesar divizor.

Notînd  $A_{ur-} = R_r / R_1$  (amplificarea pentru intrarea -) se scrie tensiunea de ieșire la una din cele două limite, de exemplu:

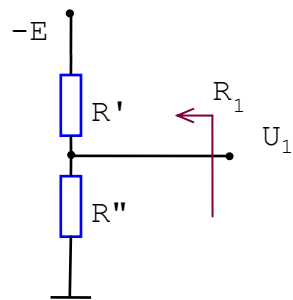
$$U_{emin} = -U_1 \cdot A_{ur-} + U_{cmin} \cdot (1 + A_{ur-})$$

unde  $A_{ur-} = A_{ur+} - 1 = 1$ . Se obține astfel

$$U_1 = [U_{cmin} (1 + A_{ur-}) - U_{emin}] / A_{ur-} = \frac{-3 \cdot 2 - 0}{1} = -6V$$

Adoptînd  $R_1 = 5k\Omega$  rezultă  $R_r = A_{ur-} \cdot R_1 = 5k\Omega$ .

Tensiunea  $U_1$  se realizează cu un divizor, astfel că  $R_1$  nu se mai folosește:



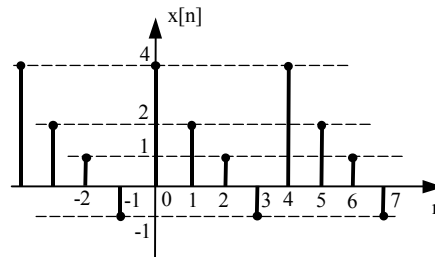
$$\begin{cases} U_1 = E \frac{R''}{R' + R''} \\ R_1 = R' \parallel R'' = R' R'' / (R' + R'') \end{cases}$$

Făcînd raportul ecuațiilor se obține

$$R_1 / U_1 = R' / E \Rightarrow R' = 12,5k\Omega \text{ și } R'' = 8,33k\Omega$$

Toate cele patru rezistențe trebuie să fie de precizie.

1. Fie semnalul în timp discret cu graficul din figură:



- a.) Să se determine coeficienții  $a_k$  ai seriei Fourier exponențiale atașate semnalului  
b.) Sa se evalueze puterea semnalului pe baza eșantioanelor  $x[n]$  și apoi pe baza coeficienților  $a_k$ .

Rezolvare:

- a.) Perioada semnalului este  $N = 4$ , iar seria Fourier exponențială are expresia:

$$x[n] = \sum_{k=0}^3 a_k e^{jk \frac{2\pi}{N} n}$$

Pentru calculul coeficienților avem:

$$a_0 = \frac{1}{4} \sum_{n=0}^3 x[n] e^{j0} = \frac{1}{4} (x[0] + x[1] + x[2] + x[3]) =$$

$$= \frac{1}{4} (4 + 2 + 1 - 1) = \frac{6}{4} = \frac{3}{2}$$

$$a_1 = \frac{1}{4} \sum_{n=0}^3 x[n] e^{-j \frac{\pi}{2} n} = \frac{1}{4} \left( 4e^0 + 2e^{-j \frac{\pi}{2}} + 1e^{-j\pi} - 1e^{-j \frac{3\pi}{2}} \right) =$$

$$= \frac{1}{4} (4 - 2j - 1 - j) = \frac{3 - 3j}{4}$$

$$\begin{aligned}
 a_2 &= \frac{1}{4} \sum_{n=0}^3 x[n] e^{-jn\frac{\pi}{2}} = \\
 &= \frac{1}{4} (x[0]e^0 + x[1]e^{-j\pi} + x[2]e^{-2j\pi} + x[3]e^{-3j\pi}) = \\
 &= \frac{1}{4} (4 + 2(-1) + 1 + (-1)(-1)) = \frac{4}{4} = 1 \\
 a_3 &= \frac{1}{4} \sum_{n=0}^3 x[n] e^{-j3\frac{\pi}{2}n} = \frac{1}{4} (3 + 3j)
 \end{aligned}$$

Întrucât  $x[n] \in \mathbb{R}$  coeficienții au proprietatea:

$$a_3 = a_1^* = \frac{1}{4} (3 + 3j).$$

$$\text{b.) } P_x = \frac{1}{N} \sum_{n=\langle N \rangle} |x[n]|^2 = \frac{1}{4} (16 + 4 + 1 + 1) = \frac{22}{4} = \frac{11}{2}$$

$$\begin{aligned}
 P_x &= \sum_{k=\langle N \rangle} |a_k|^2 = \frac{9}{4} + \frac{|3-3j|^2}{16} + \frac{|3+3j|^2}{16} + 1 = \\
 &= \frac{9}{4} + \frac{9 \cdot 2}{16} \cdot 2 + 1 = \frac{11}{2}
 \end{aligned}$$

2. Fie sistemul caracterizat de ecuația cu diferențe finite:

$$y[n] + a \cdot y[n-1] = x[n]$$

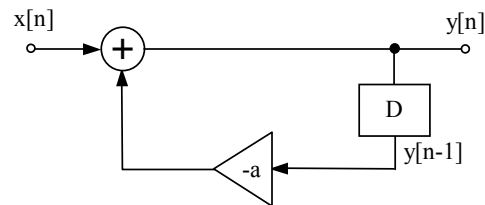
- Să se găsească o schemă de implementare a filtrului utilizând sumatoare, multiplicatoare și linii de întârziere.
- Cum se numeste un astfel de sistem?

Rezolvare:

a) Explicitând pe  $y[n]$  se obține:

$$y[n] = -a \cdot y[n-1] + x[n]$$

ce conduce la următoarea schemă:



Se remarcă prezența în cadrul structurii a unei bucle de reacție ca o consecință a naturii recursive a filtrului.

b) Sistemul este un filtru cu răspuns infinit la impuls (IIR)

## Problema 1

Sa se programeze (in limbaj C) Timer 1, in modul 3, pentru a genera intreruperi la 200 microsecunde. Frecventa de tact este 12MHz.

Se scriu 3 functii: functia de initializare timer, functia de intrerupere si o functie pricipala (main).

*Pentru rezolvare se urmareste exemplul din curs ID de la pagina 59.*

## Problema:

Se cere sa se utilizeze Timer 0 pentru a genera pe pinul 0 al portului P1 un semnal logic cu perioada de 2ms si factor de umplere 1/2. Frecventa de tact este 12MHz.

## Solutie:

Problema se poate rezolva programind Timer 0 ca si temporizator pentru 1ms. Astfel, la fiecare milisecunda, procesorul executa functia de tratare a intreruperii in care se comuta nivelul logic la portul P1.

Timer 0 ca si temporizator numara cicluri masina ( $F_{osc}/12$ ). Frecventa de incrementare  $F_i$  (perioada  $T_i$ ) a registrului de numarare (RN) este:

$$F_i = F_{osc}/12 = 1\text{MHz} \rightarrow T_i = 1\mu\text{s}$$

Pentru a obtine o intrerupere la fiecare milisecunda, RN trebuie incarcat cu o valoare de incarcare (VI) mai mica cu 1000 decit valoarea maxima (VM) + 1:

$$1000 * 1\mu\text{s} = 1000\mu\text{s} = 1\text{ms}$$

$$VI = (VM + 1) - 1000 = 65536 - 1000 = 64536 (0xFC18)$$

// Programul foloseste Timer 0 in mod 1 ca temporizator

```
#include <reg552.h>
```

```
#include <..\..\lib\ap_lib.h>
```

// Variabile

```
sbit P1.0 = P1 ^ 0; // bitul de port care se comuta
```

```
// .....
```

// Initializeaza Timer 0

```
void InitTimer0(void)
```

```
{
    TMOD = TMOD | 0x01; // modul 1(16 biti), temporizator, control soft
    TL0 = 0x18; TH0 = 0xFC; // valoarea de incarcare
    ET0 = 1; // validare intrerupere Timer 0
    TR0 = 1; // porneste Timer 0
    EA = 1; // autorizeaza global intreruperile
}
```

// Rutina de intrerupere pentru Timer0

```
Timer0() interrupt 1
```

```
{
    // Reincarca valoarea
    TL0 = 0x18; TH0 = 0xFC;
    P1.0 = ~P1.0 // comutare nivel logic
}
```

```
void main(void)
```

```
{
    InitTimer0();
```

```
while( 1 );
}
// .....
// end
```

În modul 2, rata de transfer depinde de valoarea bitului SMOD (PCON.7). Dacă SMOD=0 (după reset), rata de transfer este 1/64 din frecvența de oscilație, iar dacă SMOD=1, rata este 1/32 din frecvența de oscilație.

În modulele 1 și 3 rata de transfer (RT) este determinată de Timer 1 (rata determinată de depășirea de capacitate - "overflow rate") și de valoarea bitului SMOD, astfel :

$$RT = \frac{2^{SMOD}}{32} * T1(overflow - rate)$$

În acest caz, Timer 1 poate fi programat ca temporizator în oricare din cele trei moduri de funcționare, iar generarea întreruperilor trebuie invalidată. Cel mai frecvent, Timer 1 este configurat ca temporizator în modul 2 (8 biți cu autoîncărcare). În acest caz (vezi și tabelul 3.5.4\_1) rata de transfer este :

$$RT = \frac{2^{SMOD}}{32} * \frac{F_{osc}}{12 * [256 - (TH1)]}$$

**Tabelul 3.5.4\_1.** Rate de transfer utilizând Timer 1

| Rata de transfer | Frecvența de tact | SMOD<br>( PCON.7 ) | Timer T1  |              |                      |
|------------------|-------------------|--------------------|-----------|--------------|----------------------|
|                  |                   |                    | bit C / T | Mod de lucru | Valoare de încărcare |
| 19,2K            | 11,0592MHz        | 1                  | 0         | 2            | FDH                  |
| 9,6K             | 11,0592MHz        | 0                  | 0         | 2            | FDH                  |
| 4,8K             | 11,0592MHz        | 0                  | 0         | 2            | FAH                  |
| 2,4K             | 11,0592MHz        | 0                  | 0         | 2            | F4H                  |

**Problema:**

Se cere să se recepționeze, pe linia serială, câte un octet (caracter ASCII), să se afișeze pe LCD și să se transmită înapoi. Frecvența de tact este de 12MHz.

Alegem modul 3 de programare al interfetei seriale cu o rată de transfer de 9600 bauds. Se lucrează fără întreruperi seriale.

*Functia pentru receptie octet* așteaptă ca interfata să recepționeze un octet (RI=1) și returnează valoarea acestuia din registrul de receptie.

### Problema 2

Să se programeze (în limbaj C) interfata serială UART, în modul 1, fără facilități multiprocesor, cu rată de transfer 4.8 Kbauds, pentru a transmite continuu pe linia serială, caracterul ASCII 'A' (cod hexazecimal 0x41) la aproximativ 2 secunde. Frecvența de tact este de 11.0592MHz.

Se scriu 3 funcții: o funcție de inițializare interfata serială, o funcție de transmisie octet și o funcție principală (main).

*Pentru rezolvare se urmărește exemplul din curs ID de la pagina 79.*

*În funcția de inițializare, se programează S0CON conform temei iar pentru obținerea ratei de transfer se urmărește tabelul de la pagina 78.*

*În funcția principală (main), în buclă "while(1){}" se face doar transmiterea caracterului 'A' și temporizare 2 secunde folosind funcția Delay(2000).*



*Functia pentru transmisie octet asteapta ca octetul anterior sa fie transmis (TI=1), dupa care scrie noul octet in registrul de transmisie.*

```
// Recepie si transmisie pe portul serial fara intreruperi( polling )
#include <reg552.h>
#include <stdio.h>
#include <..\..\lib\ap_lib.h>
```

```
// Buffer pentru afisare caracter pe LCD
unsigned char buffer[ 2 ] = "0";
```

```
// Initializeaza interfata seriala UART
void InitSerial(void)
{
    S0CON = 0xF0;    // modul 3, multiprocesor, receptie validata
    TMOD  = 0x20;    // Timer 1 in mod 2, temporizator
    TH1 = 0xFD;      // rata de transfer 9600 bauds
    TL1 = 0xFD;
    TR1 = 1;         // porneste Timer 1
    ET1 = 0;         // NU genereaza intrerupere Timer 1
    ES0 = 0;         // NU genereaza intrerupere UART
    TI = 1;          // se poate transmite
}
```

```
// Functia de transmisie octet
void Transmit(unsigned char Data)
{
    // Transmite un caracter pe linia seriala daca TI = 1
    while(! TI );    // asteapta TI=1
    TI = 0;
    TB8 = 0;         // bitul 9 pe zero
    S0BUF = Data;
}
```

```
// Functia de receptie octet
unsigned char Receive(void)
{
    // Receptioneaza un caracter de pe linia seriala daca RI = 1
    while(! RI );    // asteapta RI=1
    RI = 0;
    return S0BUF;
}
```

```
// .....
void main(void)
{
    lcd_InitDisplay();
    InitSerial();

    while( 1 ){
        buffer[0] = Receive();    // receptie octet

        lcd_Write( 1, buffer );   // afisare pe LCD
        Transmit( buffer[0] );    // transmisie octet
    }
}
// .....
// end
```