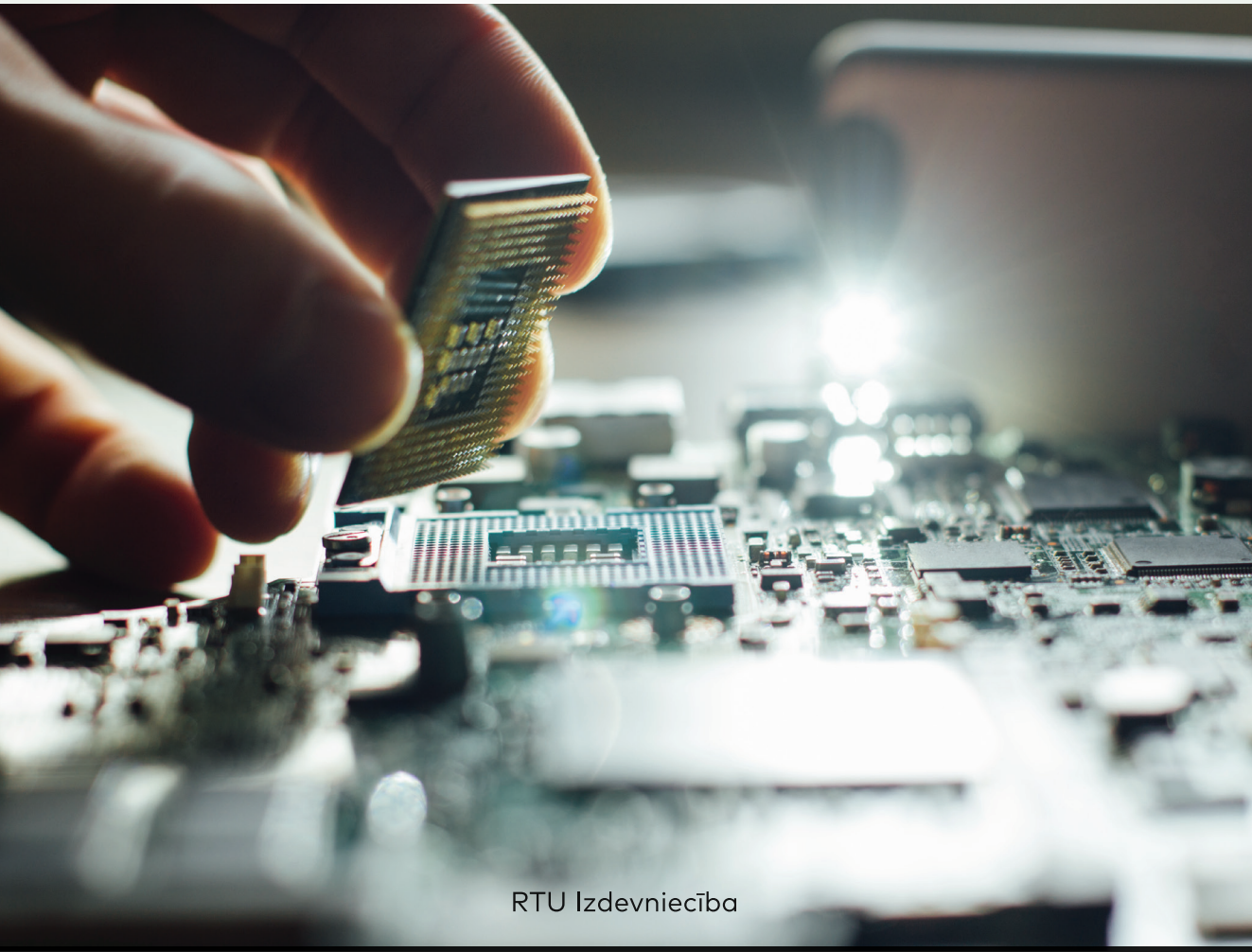
Ansis Klūga, Jānis Klūga

TRANSPORTA MIKROPROCESORU SISTĒMAS

Laboratorijas darbi un metodiskie norādījumi



RTU Izdevniecība

Rīgas Tehniskā universitāte
Elektronikas un telekomunikāciju fakultāte

Ansis Klūga, Jānis Klūga

TRANSPORTA MIKROPROCESORU SISTĒMAS

Laboratorijas darbi un metodiskie norādījumi

RTU Izdevniecība
Rīga 2017

Transporta mikroprocesoru sistēmas. Laboratorijas darbi un metodiskie norādījumi. Ansis Klūga, Jānis Klūga. — Rīga Izdevniecība: RTU, 2017. 66 lpp.

Metodiskie norādījumi paredzēti studentiem, kas apgūst transporta mikroprocesoru sistēmas dažādos līmeņos: bakalaura, profesionālā bakalaura un profesionālā maģistra. Metodiskie norādījumi sastāv no trīs daļām. Pirmajā daļā paskaidrotas galvenās mikroprocesora *Intel 8085* (*I8085*) atšķirības no metodiskā līdzeklī (Klūga, 2007) aprakstītā mikroprocesora *Intel 8080*. Aprakstīta mikroprocesora *I8085* apmācības ierīce *Primer*, kas izgatavota firmā *EMAC*, kā arī doti paskaidrojumi laboratorijas darbu un studiju darba izpildei, izmantojot minēto ierīci. Otrā daļā aprakstīta programmas izveide mikrokontrollerim *8051*, kā arī skaidrota apmācības ierīces *MTS-51* izmantošana laboratorijas darbos un studiju projektā. Aprakstīta programmas asamblēšana, linkēšana un pārbaude gan DOS, gan *Windows* vidē. Metodisko norādījumu trešā daļa veltīta mikroprocesora *Intel 8086* (*I8086*) programmēšanas iemaņu apgūšanai un studiju projekta veidošanai, izmantojot ierīci *MTS-86C*.

Metodiskos norādījumus sagatavoja A. Klūga. J. Klūga piedalījās laboratorijas darbu izstrādē mikroprocesoram *I8085* un mikroprocesoram *I8086*.

Apstiprināta Transporta elektronikas un telemātikas katedras sēdē 2016. gada 20. maijā, protokols Nr. 2-16.

Atbildīgā par izdevumu Natālija Čina
Literārā redaktore Lilita Vīksna
Datorsalikums Jekaterina Lukina
Tehniskā redaktore Irēna Skārda
Vāka dizains Jekaterina Lukina

Vāka attēls no Shutterstock.com

Izdevējs RTU Izdevniecība
Kaļķu iela 1, Rīga, LV-1658
E-pasts: izdevnieciba@rtu.lv

Tiražētājs RTU Digitālās drukas centrs

ISBN 978-9934-10-875-4

SATURA RĀDĪTĀJS

I. MIKROPROCESORA *INTEL 8085* LIETOŠANA

1. Vispārīgā informācija par mikroprocesoru *Intel 8085*

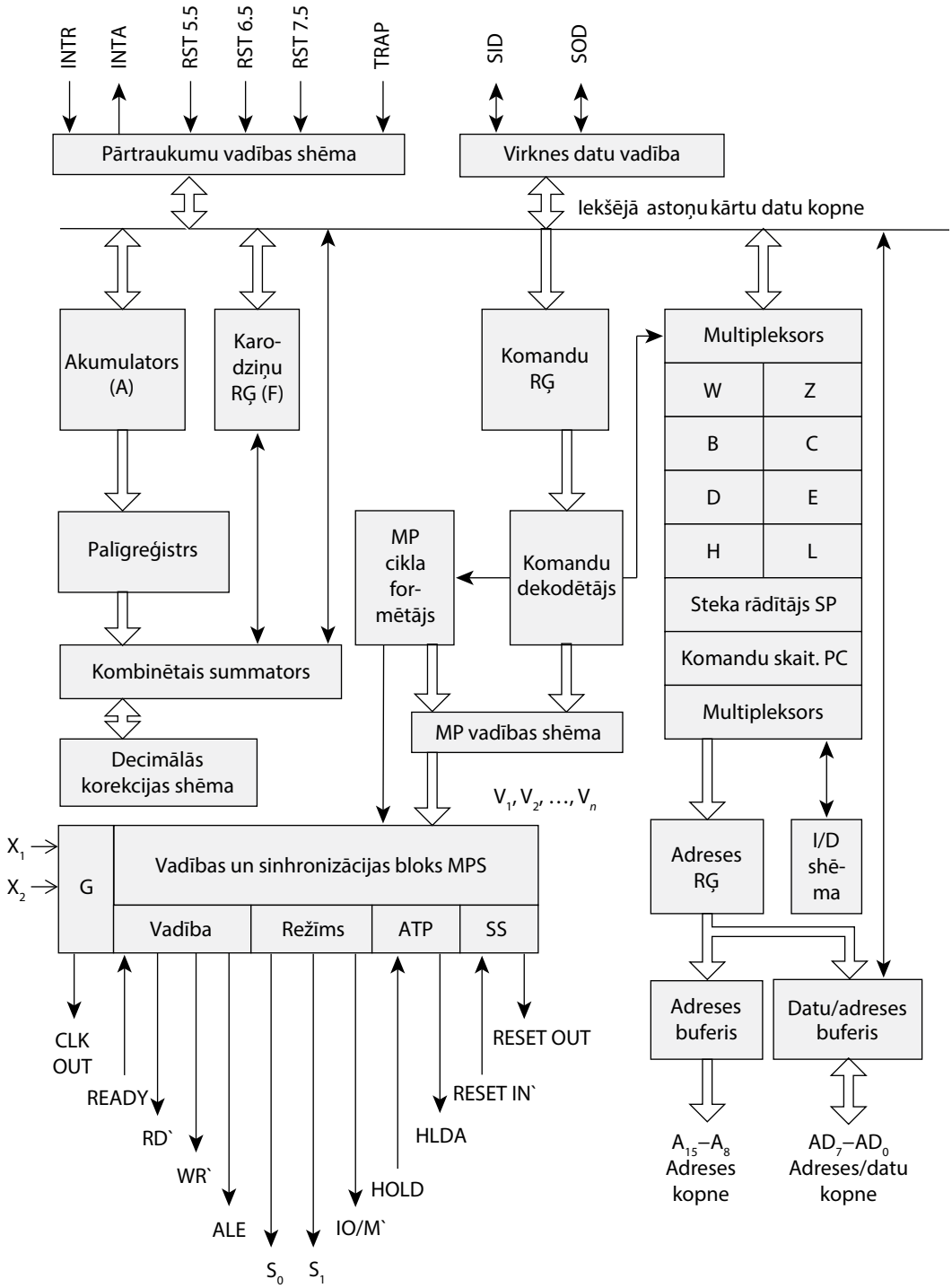
Mikroprocesors *Intel 8085* ir izgatavots pēc KMOP tehnoloģijas, un tam ir tikai viens barošanas spriegums (+5 V) un ieviests arī pazemināta jaudas režīms (*power down mode*). Darba frekvence ir palielināta un tās maksimālais lielums ir 5 MHz. Lai izslēgtu stāvokļa informācijas izdošanu pa datu kopni, nepalielinot mikroshēmas kājiņu skaitu, izmanto kopēju multipleksējamu adreses un datu kopni (AD_7–AD_0). Tas, protams, sarežģī informācijas apmaiņu starp mikroprocesoru un atmiņu, kā arī ievadizvades ierīcēm (III). Adreses jaunāko kārtu (AD_7–AD_0) informācija jāsaglabā speciālā reģistrā, lai nodrošinātu datu apmaiņas procesu.

Mikroprocesorā *Intel 8085* ir ievietoti daudzi mikroprocesora *I8080* sistēmas elementi. Ārējais takts impulsu ģenerators tagad atrodas mikroprocesora mikroshēmā un, lai tas sāktu darboties, jāpievieno kvarca frekvences stabilizators pie izvadiem X_1, X_2 (skat. mikroprocesora *I8085* struktūras shēmu 1.1. attēlā). Turklāt nav nepieciešama arī vadības signālu formēšanas mikroshēma — tie tiek formēti tieši mikroprocesora mikroshēmā (RD`,WR`,IO/M`). Bez tam tiek formēts papildu signāls — adreses ieraksta signāls (ALE — *address latch enable*), kas nodrošina adreses informācijas fiksāciju ārējā reģistrā.

1.1. tabula

Mikroprocesora mašīnas ciklu raksturojuma signāli

Mašīnas cikls	IO/M`	S_1	S_0
Ieraksts atmiņā	0	0	1
Atmiņas nolasīšana	0	1	0
Ieraksts izvadierīcē	1	0	1
Ievadierīces nolasīšana	1	1	0
Operācijas koda nolasīšana	0	1	1
Pārtraukuma apstiprinājums	1	1	1
Apstāšanās	×	0	0



1.1. att. Mikroprocesora *Intel 8085* struktūras shēma.

Mikroprocesora stāvokli mašīnas ciklā raksturo signāli S_0, S_1, un IO/M`, kā tas paskaidrots 1.1. tabulā. Mikroprocesorā *I8085* ir izmainīta pārtraukumu sistēma, salīdzinot ar mikroprocesoru *I8080*, un ieviesti papildu sistēmas pārtraukumi, kā parādīts 1.2. tabulā. Pārtraukuma pieprasījuma signāls INTR (*interrupt request*) atbilst mikroprocesora *I8080* signālam INT. Ārējie pārtraukumi TRAP, RST 7.5, RST 6.5, RST 5.5 nostāda mikroprocesoru sākuma stāvoklī, turklāt TRAP nevar maskēt ar programmas palīdzību.

Mikroprocesorā *Intel 8085* ir ieviesta arī virknes datu pieņemšana (SID — *serial input data line*) un nosūtīšana (SOD — *serial output data line*). Šo datu pārsūtīšanu vada virknes ievadizvades ierīču vadības bloks.

1.2. tabula

Pārtraukumu veidi, to prioritātes un sākumadreses

Pārtraukuma nosaukums	Pārtraukuma prioritāte	Pārtraukuma sākumadrese	Piezīmes
TRAP	1.	24H	Nemaskēts pārtraukuma pieprasījums
RST 7.5	2.	3CH	
RST 6.5	3.	34H	
RST 5.5	4.	2CH	
INTR	5.		Sākumadresi nosaka ievadītā informācija pēc pārtraukuma pieprasījuma atļaujas

Mikroprocesora *I8085* komandu sistēma ir papildināta ar šādām komandām:

- pārtraukuma maskas lasīšana **RIM** (*read interrupt mask*);
- pārtraukuma maskas uzstādīšana **SIM** (*set interrupt mask*);
- sākuma stāvoklis **RST** (*reset*).

SIM un RIM komandas vienlaicīgi ar pārtraukuma masku uzstādīšanu un lasīšanu veic datu ievadi vai nolasīšanu no attiecīgās virknes spailes.

Pazeminātas jaudas režīmu var uzstādīt divos veidos:

1) izpildot komandu **HLT**;
2) pēc HOLD signāla saņemšanas.

Pāreja darba režīmā notiek pēc *reset* signāla vai noņemot kopnes pieejas atļaujas signālu HLDA.

Mikroprocesora *Intel 8085* struktūras shēma parādīta 1.1. attēlā.

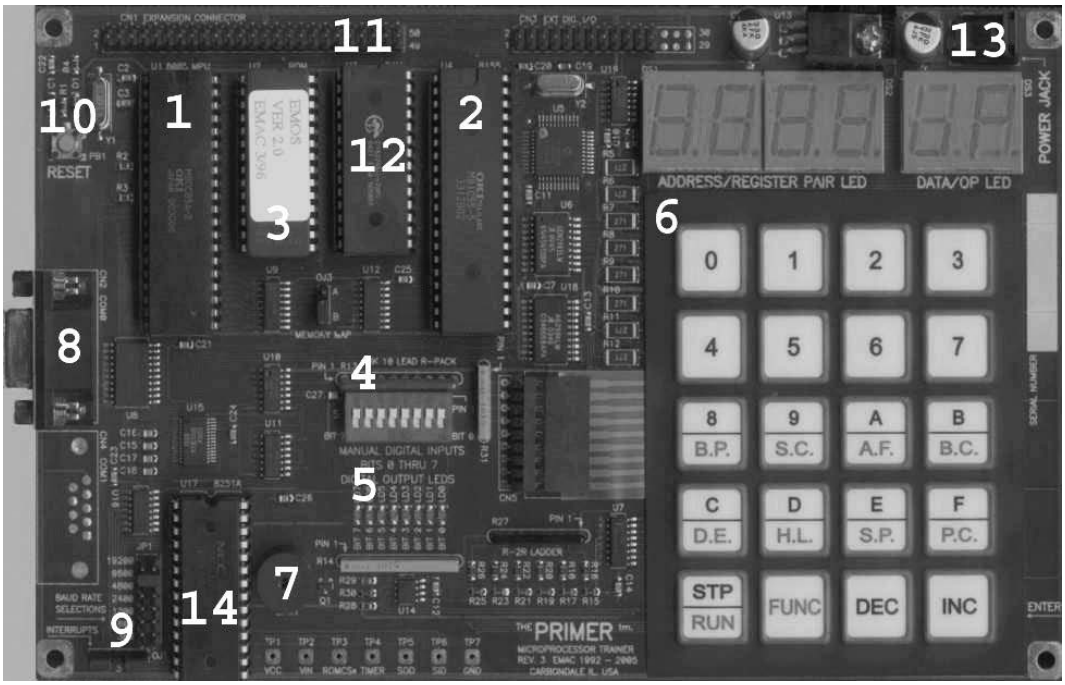
2. Laboratorijas ierīces *Primer* apraksts

Laboratorijas ierīcē *Primer* ietilpst:

- *Intel 8085* sērijas mikroprocesors;
- programmējamais perifērijas interfeiss ar taimeri un 2048 baitu RAM;
- paplašināta MOS operētājsistēma;

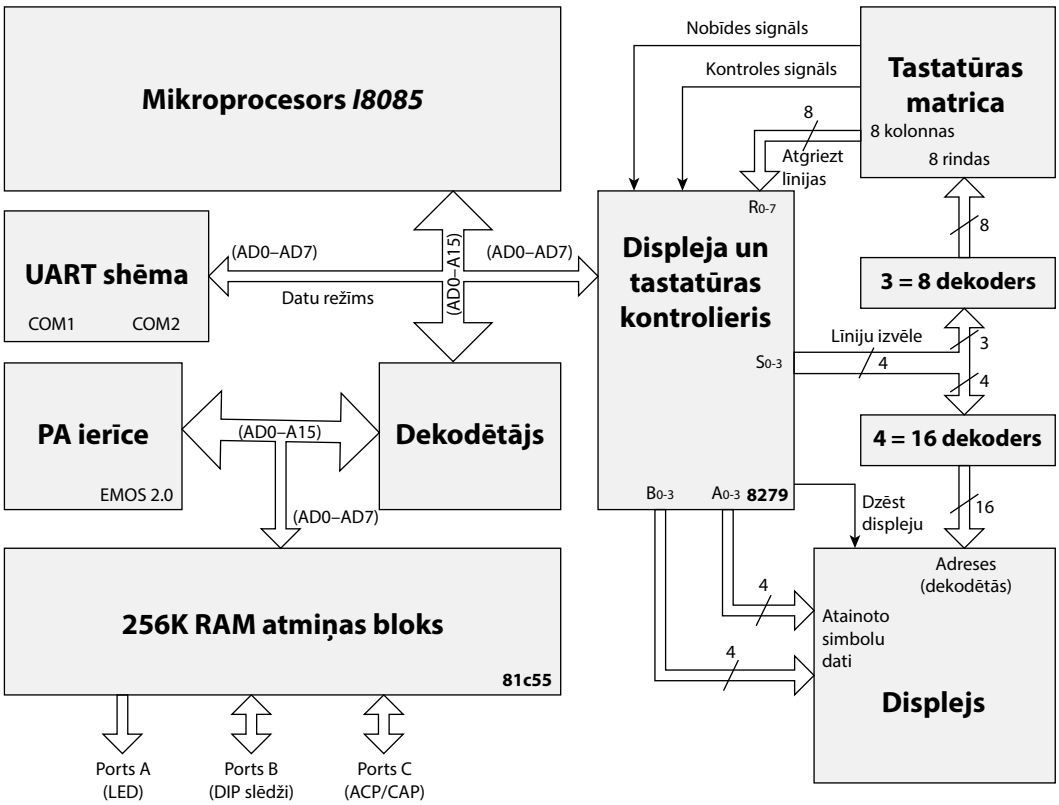
- astoņu bitu slēdžu (DIP) ievadports;
- astoņu bitu diožu izvadports;
- programmējamais tastatūras/displeja interfeiss;
- skaļrunis;
- *Primer* COM ports pieslēgšanai datoram;
- pārtraukuma režīma un **Baud** ātruma izvēles tiltslēga (*jumper*) tipa slēdži;
- plates poga **Reset**;
- paralēlie porti;
- RAM atmiņa;
- barošanas pieslēguma punkts;
- universālais sinhronais uztvērējs/raidītājs.

Laboratorijas ierīces *Primer* montāžas shēma parādīta 1.2. attēlā.



1.2. att. Laboratorijas ierīces *Primer* montāžas shēma: 1 — *Intel 8085* sērijas mikroprocesors; 2 — programmējamais perifērijas interfeiss ar taimeri un 2048 baitu RAM; 3 — paplašināta MOS operētājsistēma; 4 — astoņu bitu slēdžu ievadports; 5 — astoņu bitu diožu izvadports; 6 — programmējamais tastatūras/displeja interfeiss; 7 — skaļrunis; 8 — Primer COM ports pieslēgšanai datoram; 9 — pārtraukuma režīma un datu pārraides ātruma izvēles tiltslēga tipa slēdži; 10 — plates poga Reset; 11 — paralēlie porti; 12 — RAM atmiņa; 13 — barošanas pieslēguma punkts; 14 — universālais sinhronais uztvērējs/raidītājs.

Laboratorijas ierīces *Primer* struktūras shēma parādīta 1.3. attēlā. Tās galvenās sastāvdaļas ir mikroprocesors (MP) *I8085*, atmiņas ierīce, kas sastāv no lasāmatmiņas (ROM) un operatīvās atmiņas (RAM), un ievadizvades ierīcēm (I/O).



1.3. att. Laboratorijas ierīces *Primer* struktūras shēma.

3. Laboratorijas ierīces *Primer* sagatavošana darbam

3.1. Pieslēgšana

Atšķirībā no UMKVEF laboratorijas ierīce *Primer* pieslēdzama datoram un programmas ieraksts operatīvajā atmiņā tiek veikts ar speciālas programmas palīdzību. Lai sāktu darbu, *Primer* COM ports jāpieslēdz datora COM portam. Apmaiņas procesa vadībai jāizmanto programma *Hyperterminal*.

Hyperterminal programma strādā *Windows* vidē un ļauj savienoties ar ārējām ierīcēm. Kad simbols tiks saņemts, lietojot datora virknes portu, kuru izmanto *Hyperterminal*, tas tiks parādīts programmas logā, bet, kad poga ir nospiesta *EMAC* tastatūrā, atbilstošais simbols tiek sūtīts pa virknes portu. Lai uzstādītu apmaiņas ātrumu 9600 bodi laboratorijas ierīcē *Primer*, izmanto šādas slēdžu (tiltslēgu) pozīcijas:

- JP1 = 9600;
- OJ1 = tiltslēgs 1-2 un 3-4 kontaktu savienojums;
- OJ2 = tiltslēgs B;
- OJ3 = tiltslēgs B.

Programma pēc noklusēšanas ir noskaņota, lai izmantotu portu COM1, lietojot virknes kabeli, kas pieslēgts pie laboratorijas ierīces. Lai iesāktu darbu ar apmaiņas programmu, datorā jāpalaiž programma PRIMER_XP-2K-NT.ht, ja datorā ir uzstādīta operētājsistēma *Microsoft Windows XP*, vai PRIMER_95_98.HT, ja ir uzstādīta operētājsistēma *Microsoft Windows 95* vai *Microsoft Windows 98*.

3.2. HEX faila sagatavošana

Atveriet jums pieejamu teksta redaktoru. Uzrakstiet programmu un saglabājiet failu ar paplašinājumu .asm. Lai iegūtu HEX failu, vislabāk izmantot krosasambleri MA85.exe, bet, tā kā mikroprocesora *I8085* komandu sistēma ir praktiski identiska *I8080*, tad var izmantot failus, kas iegūti ar krosasambleri *ASM80*. *MA85* lietošanas skaidrojums:

- ***MA85* izmantošana *FAR* vidē**.
  Atveriet direktoriju, kurā atrodas fails MA85.exe, un ievietojiet tur jūsu ASM failu. Komandu rindā uzrakstiet "MA85.exe -l XXX.asm XXX.hex", kur XXX — jūsu faila nosaukums. Asambleris izveidos HEX un LST failus, kuri gatavi tālākai izmantošanai;
- ***MA85* izmantošana *Windows* vidē**.
  Atveriet direktoriju, kurā atrodas fails MA85.exe, un ievietojiet tur jūsu ASM failu, pārnesiet to (turot kreiso peles taustiņu) uz failu MA85.exe. Programma izveidos OBJ failu ar tādu pašu nosaukumu kā ASM fails. Iegūtais OBJ fails ir tas pats HEX fails, atšķirīgs ir tikai paplašinājums. OBJ failu var tāpat ierakstīt atmiņā. Metodes trūkums — pēc kompilēšanas nevar dabūt LST failu. Lai to novērstu, veiciet *shortcut* failam MA85.exe un tur komandu rindā pierakstiet "–l" parametru, tagad, pārnesot ASM failu, jūs iegūsiet gan OBJ, gan LST failu.

3.3. HEX faila nosūtīšana uz *Primer*

Palaidiet nepieciešamo apmaiņas programmu PRIMER_XP-2K-NT.ht vai PRIMER_95_98.HT. Nospiediet pogu **Reset** (tā atrodas laboratorijas ierīces augšējā kreisajā stūrī), **FUNC** un **0** (uz *Primer* tastatūras), lai iedarbinātu laboratorijas ierīces vadības programmu *EMOS*. Jūs ieraudzīsiet šādu informāciju uz datora, kas strādā ar programmu *Hyperterminal*, displeja:

```
E-FORTH V1.10 SBC+
Copyright 1987-91 EMAC inc.
License no. 00.
```

```
        EMOS V2.00 HELP MENU

B -->   Bring Block from RAMDISK to memory
C -->   Change register contents
D -->   Dump memory contents
E -->   Edit memory contents
F -->   Fill memory with byte
G -->   Go execute program { full speed }
H -->   Hex/dec math { 1st + 2nd, 1st - 2nd }
I -->   Input from I/O port
L -->   List memory contents using mnemonics
M -->   Move section of memory
O -->   Output to I/O port
R -->   display Register contents
S -->   MOS Service call
T -->   Trace program execution
W -->   Write memory to RAMDISK block
Z -->   Zap application EPROM
< -->   hex download from trainer to host
> -->   hex upload to trainer from host
? -->   display this help menu
```

Nospiediet uz datora tastatūras >, lai ierakstītu HEX failu laboratorijas ierīces atmiņā. *Hyperterminal* piedāvās jums ievadīt programmas sākumadresi:

```
- >     STARTING ADDRESS..
```

Ievadiet "FF01" — tā būs šūna, no kuras sāksies jūsu programmas ierakstīšana atmiņā. *Hyperterminal* atbildēs ar:

```
- >     STARTING ADDRESS.. FF01
       READY TO RECEIVE, <ESC> TO ABORT
```

Tas nozīmē, ka *Primer* ierīce ir gatava saņemt informāciju (HEX failu). Izvēlieties **Transfer → Send Text File**. Atvērtā **File Browser** logā samainiet **File of type** opciju uz **All files (*.*)** un izvēlieties jūsu HEX failu. Faila ierakstīšana aizņem ne vairāk par divām sekundēm, kad tā beidzas, atkal parādās "-".

3.4. Ierakstītās programmas palaišana

Pēc programmas ierakstīšanas uz datora tastūras nospiediet taustiņu **G** (no angļu val. *go* — palaist), un apmaiņas programmas logā parādīsies:

```
- G STARTING ADDRESS..
```

Ievadiet "FF01" (vai citu adresi, kas sakrīt ar jūsu programmas sākumadresi), un, kad terminmāls prasīs *breakpoint* adresi, nospiediet **Enter**. Pirms programmas palaišanas *Hyperterminal* logā parādīsies:

```
- G STARTING ADDRESS.. FF01 BREAKPOINT ADDRESS..
RUNNING.. HIT ANY KEY TO STOP
```

4. LABORATORIJAS DARBI

4.1. Laboratorijas darbs. Astoņu bitu skaitļu un 16 bitu skaitļu saskaitīšana

4.1.1. Astoņu bitu skaitļu saskaitīšana

■ **Uzdevums**. Saskaitīt heksadecimālos skaitļus 12h un 1fh.

■ **Darba gaita**. Lai to izdarītu, skaitļi jāievada akumulatorā (A) un vienā no reģistriem, piemēram, B, un pēc tam jāsaskaita. Laboratorijas ierīcē iebūvētā vadības sistēma (MOS) komandas skaitītāju nostāda uz adresi FF01h. Tāpēc visas programmas sāksiet ar šo adresi. Summu ievietosiet atmiņas šūnā FF40h, jo programmas izmantoto adrešu kopums neaizņems lielāku adrešu telpu.

Programmas pieraksts asamblera valodā.

```
org 0ff01h
mvi a,12h       ; ievada pirmo skaitli akumulatorā
mvi b,1fh       ; ievada otro skaitli B reģistrā
add b           ; saskaita A un B (rezultāts) A reģistrā
sta 0ff40h      ; ievieto summu atmiņas šūnā ff40h
rst 7           ; atdod kontroli MOS
end
```

Saskaitīšanas rezultātu atradīsiet atmiņas šūnā ar adresi ff40h. Lai nolasītu šūnas saturu, var ievadīt "E" *Hyperterminal* programmā un attiecīgo adresi vai pēc pogas **Reset** nospiešanas palielināt adreses rādītāju laboratorijas ierīcē, spiežot pogu **INC**, kamēr sasniedzat vajadzīgo adresi.

4.1.2. 16 bitu skaitļu saskaitīšana

■ **Uzdevums**. Saskaitīt 16 bitu skaitļus, kas atrodas reģistros BC un DE (jaunākie baiti — C un E reģistros). Saskaitīšanas rezultātu ievietosiet pirmā skaitļa vietā BC reģistrā. Programma uzdevuma risināšanai:

```
org 0ff01h
mov a,c ; pirmā skaitļa jaunākā baita ievietošana akumulatorā
add e   ; reģistra E satura pieskaitīšana akumulatora saturam
mov c,a ; ievieto jaunāko baitu saskaitīšanas rezultātu C reģistrā
mov a,b ; pirmā skaitļa vecākā baita ievietošana akumulatorā
adc d   ; saskaita vecākos baitus (rezultāts A)
mov b,a ; vecāko baitu saskaitīšanas rezultātu ievieto B reģistrā
rst 7   ; atdot kontroli MOS
end
```

Lai izmainītu reģistru saturu pēc programmas ierakstīšanas atmiņā, var izmantot komandu **Change reģister contents**, nospiežot "C" *Hyperterminal* interfeisā.

Ievadiet reģistra pāros 16 bitu skaitļus, piemēram, 2302h un 10FFh. Pirmo ievadām BC reģistros, otro — DE reģistros. Programmas izpilde dos šādu rezultātu:

```
2302h
+   10FFh
=   3401h
```

Samainot komandu **ADD** uz **SUB** un **ADC** uz **SBC**, programma atņems vienu 16 bitu skaitli no otra. Patstāvīgi uzrakstiet šo programmu un pārbaudiet, izmantojot laboratorijas ierīci *Primer*.

4.2. Laboratorijas darbs. Datu baita sadalīšana divās daļās

■ **Uzdevums**. Izdalīt no datu baita, kas atrodas reģistrā B, četras jaunākās un četras vecākās kārtas, un tās attiecīgi ievietot atmiņas šūnu ff41h un ff40h jaunākajās kārtās.

■ **Darba gaita**. Datu maskēšana notiek, izmantojot loģisko operāciju komandu **ANI <byte>**, ja maskā dotajā kārtā būs 0, tā tiks maskēta, ja 1, tad netiks maskēta. Datu baitam izmantot skaitli 23h.

```
org 0ff01h
mvi b,23h       ; ievada B reģistrā datu baitu 23h
mov a,b         ; ievada akumulatorā B reģistra saturu
ani 00001111b   ; maskē četras vecākās kārtas
sta 0ff41h      ; ievieto akumulatora saturu AŠ ff41h
mov a,b         ; ievada akumulatorā B reģistra saturu
rrc             ; datu četras vecākās kārtas pārvieto uz jaunāko vietām
rrc             ;
rrc             ;
rrc             ;
ani 11110000b   ; maskē četras vecākās kārtas
sta 0ff40h      ; ievieto datu baita vecākās kārtas AŠ ff40h
rst 7           ; atdot kontroli MOS
end
```

Pēc programmas izpildes atmiņas šūnā ff40h būs skaitlis 02, bet atmiņas šūnā ff41h — skaitlis 03.

4.3. Laboratorijas darbs. Skaitļu masīva maksimuma atrašana un divu daudzbaitu skaitļu saskaitīšana

4.3.1. Skaitļu masīva maksimuma atrašana

■ **Uzdevums**. Atrast maksimumu skaitļu masīvā. Skaitļi ievietoti atmiņā, sākot ar atmiņas šūnu ff01, turklāt pirmais skaitlis norāda masīva garumu *n*. Atrast risinājumu, ja $n = 3$, bet skaitļi ir 12h, 43h un 1fh. Maksimālo skaitli ievieto atmiņas šūnā ff40h.

■ **Risinājums**. Programmā izmantosim komandu **CMP M**. Tā salīdzina akumulatorā ievietoto skaitli ar to skaitli, kura adresi atmiņā norāda HL reģistra pārī ievietotā adrese. Komandas izpildes rezultāts ir karodziņa uzstādīšana. Ja atmiņas šūnā ievietotais skaitlis ir mazāks par skaitli akumulatorā ($M < A$), tad karodziņš $C = 0$, ja otrādi, tad $C = 1$. Pēc C karodziņa uzstādīšanas, komanda pāriet uz vajadzīgo izpildes posmu ar zarošanās komandu **JNC** (*jump not C*), bet cikla organizēšanai tiek izmantota komanda **JNZ** (*jump not zero*).

```
        org 0ff01h
        lxi h,sk          ; HL reģistrā pirmā masīva elementa adrese
        mov b,m           ; B reģistrā ievieto masīva garumu n
        dcr b             ; samazina B reģistra saturu par vienu, jo A ir vienāds
                          ; ar pirmo skaitli
        inx h             ; palielina HL reģistra saturu par vienu
        mov a,m           ; ievada pirmo skaitli akumulatorā
max:    inx h             ; palielina HL reģistra saturu par vienu
        cmp m             ; salīdzina akumulatora saturu ar nākamo skaitli
        jnc m1            ; ja skaitlis mazāks par A saturu uz m1
        mov a,m           ; ievieto akumulatorā lielāko skaitli
m1:     dcr b             ; samazina skaitītāju par vienu
        jnz max           ; ja B nav nulle, turpināt salīdzināšanu
        sta 0ff40h        ; ievieto maksimālo skaitli atmiņas šūnā ff40h
        rst 7             ; atdot kontroli MOS
sk:     db 3h,12h,43h,11h ; skaitļu masīvs
        end
```

Skaitļu masīvs atrodas pēc norādes "sk:". Pēc komandas **db** (*data byte*) ir ievietoti, mūsu piemērā, četri skaitļi, tie glabāsies atmiņā uzreiz aiz programmas. Pirmais skaitlis norāda masīva garumu *n*. Programmas sākumā tas iekopējas B reģistrā un samazinās par 1 ar katru apstrādāto skaitli. Kad tas ir vienāds ar 0, programma ir salīdzinājusi visus skaitļus. Programmas izpildes rezultātā atmiņas šūnā ff40h būs skaitlis 43.

4.3.2. Divu *n* baitu skaitļu saskaitīšana, izmantojot netiešo adresāciju

■ **Uzdevums**. Saskaitīt divus skaitļus, kuru garums baitos *n* dots atmiņas šūnā (AŠ) ff40h, bet paši skaitļi novietoti, sākot ar AŠ ff41h un ff51 attiecīgi. Sākumā novietotas jaunākās kārtas, bet pēc tam vecākās. Rezultāts jāievieto AŠ, kurās glabājās pirmais skaitlis. Piemērs, nepieciešams saskaitīt divus 24 kārtu skaitļus 50A429h un 2837FBh. Šajā piemērā $n = 24/8 = 3$. Tabulā parādīts, kā ievietota informācija atmiņas šūnās:

AŠ adrese	ff40h	ff41h	ff42h	ff43h	ff51h	ff52h	ff53h
Datu baits	03	29	A4	50	FB	37	28

```
        org 0ff01h
        sub a           ; nullēt pārneses bitu
        lxi h,0ff40h    ; ievietot skaitītājā vārda garumu baitos
        mov b,m         ;
        lxi d,0ff50h    ; otrā skaitļa rādītāja uzstādīšana
sum:    inx d           ;
        inx h           ;
        ldax d          ; paņemt otrā skaitļa astoņas kārtas un ievietot akumulatorā
        adc m           ; pieskaitīt pirmā skaitļa astoņas kārtas
        mov m,a         ; ievietot summu pirmā skaitļa vietā
        dcr b           ; samazināt skaitītāja saturu par vienu
        jnz sum         ; apstrādāt nākamos astoņus bitus
        rst 7           ; atdot kontroli MOS`am
        end
```

Lai ievadītu un nolasītu atmiņu, izmantojiet taustiņu **E** *Hyperterminal* programmā un attiecīgajā adresē ievadiet vai nolasiet informāciju.

4.4. Laboratorijas darbs. Iepazīšanās ar MOS servisiem LEDHEX un PITCH

Šajā laboratorijas darbā iepazīsimies ar ļoti svarīgu *Primer* funkciju — darbs ar MOS servisiem vai apakšprogrammām (MOS — *services or subroutines*). Darbs ar tiem notiek, ievietojot C reģistrā servisa numuru un izpildot komandu **CALL 1000h**.

4.4.1. MOS serviss: LEDHEX (servisa numurs 12)

Ar šī servisa palīdzību tiek parādīts DE reģistra pārī ievietoto četru skaitļu kods uz četriem kreisajiem indikatoriem. Uzdevumā parādīsim skaitļu 1, 2, 3 un 4 atainojumu.

```
mos     equ     1000h     ; mainīgam mos tiek piešķira vērtība 1000
ledhex  equ     12h       ; mainīgam ledhex tiek piešķira vērtība 12
        org     0ff01h
        mvi     c,ledhex  ; ievada C reģistrā servisa numuru
        lxi     d,1234h   ; ievieto DE reģistru pārī uzdevumā dotos skaitļus
        call    mos       ; izsauc servisa programmu
wait:   jmp     wait      ; pāreja gaidīšanas režīmā
        end
```

4.4.2. MOS serviss PITCH (servisa numurs 10 — skaņas augstums)

Ar šī servisa palīdzību DE reģistra pāra saturs tiek nosūtīts uz skaļruņa taimeri (tikai 14 jaunākie biti). Jo lielāks reģistra saturs, jo zemāks skaņas augstums. Ja saturs ir nulle, skaņa tiek izslēgta. Apskatīsim uzdevumu, kuru izpildot, tiek izdalīti 14 jaunākie biti no 16 ievadītiem ar DIP slēdžiem (12. ports), izdalītā informācija tiek ierakstīta DE reģistra pārī un ierīces *Primer* skaļrunis atskaņo signālu, kura frekvence ir apgriezti proporcionāla DE reģistrā ierakstītam lielumam.

```
dips    equ     12h             ; mainīgam dips tiek piešķirta vērtība 12h
mos     equ     1000h           ; mainīgam mos tiek piešķirta vērtība 1000h
        org     0ff01h
loop:   in      dips            ; ievads no 12. porta, kuram pieslēgti slēdži
        rrc                     ; nobīde pa labi
        rrc                     ; nobīde pa labi
        mov     e,a             ; ievietot akumulatora saturu E reģistrā
        ani     00111111b       ; maskēt divas vecākās kārtas
        mov     d,a             ; A saturu ievieto D reģistrā
        mov     a,e             ; divas jaunākās kārtas E reģistrā
        ani     11000000b       ;
        mov     e,a             ;
        mvi     c,10h           ; izsaukt 10. servisu
        call    mos             ;
        jmp     loop            ; atgriezties uz slēdžu aptauju
        end
```

Ar komandu **in dips** (dips = 12h) programma veic DIP slēdžu aptauju un saglabā rezultātu akumulatorā.

Aprakstītā programma izmanto servisu **PITCH**. Tā kods ir 10h, tas nozīmē, ka, ja reģistrā C glabāsies "10h" un programma izsauks MOS (**call mos / call 1000h**), shēmas skaļrunis **PITCH** sāks izdot skaņu.

Skaļrunis **PITCH** servisa gadījumā reaģē tikai uz 14 jaunākajiem DE reģistrā saglabātiem bitiem. Tāpēc, lai slēdžu pārslēgšana maksimāli ietekmētu signāla frekvenci, ir nepieciešams, lai saņemtā no astoņiem slēdžiem informācija būtu ar maksimālo "svaru":

```
DE reģistrs     xx00000000000000   (14 kārtas)
DIP slēdži        00000000         (astoņas kārtas)
```

Tās nozīmē, ka sešām vecākajām kārtām jābūt D reģistrā un divām jaunākajām — E. To var sasniegt ar jau pazīstamo maskēšanu un rotēšanu, kā parādīts programmā.

4.5. Laboratorijas darbs. Apakšprogrammas izmantošana aiztures veidošanā

Apakšprogrammas izpildīšanu nodrošina komandas **CALL** un **RET**. Apakšprogramma ir programmas daļa, kas izpilda kaut kādu noteiktu, galvenajai programmai vajadzīgu funkciju. Šajā gadījumā tā ir aizture, ar kuru tiks izgaismotas trīs diožu kombinācijas. Ar komandu **LXI 8000h** ievietosim HL reģistrā skaitli 8000, kuru samazināsim (dekrementēsim ar komandu **DCX**) apakšprogrammas gaitā.

```
leds    equ     11h             ; mainīgam leds tiek piešķirta vērtība 11h
        org     0ff01h          ; programmas sākumadrese
        lxi     h,8000h         ; ievada konstanti HL reģistru pārī
loop:   mvi     a,00001111b     ; pirmā diožu kombinācija (kreisā)
        call    delay           ; aiztures apakšprogrammas izsaukšana
        cma                     ; otrā diožu kombinācija (labā)
        call    delay           ; aiztures apakšprogrammas izsaukšana
        mvi     a,01010101b     ; trešā diožu kombinācija (pāra)
        call    delay           ; aiztures apakšprogrammas izsaukšana
        dcr     h               ; konstantes samazināšana
        jmp     loop            ; atgriezties cikla sākumā
delay:  push    h               ; saglabāt datus stekā
        push    psw             ;
        out     leds            ; iedegt diodes
delay1: dcx     h               ; laika aiztures formēšana
        mov     a,h             ;
        ora     L               ;
        jnz     delay1          ;
        pop     psw             ; izņemt datus no steka
        pop     h               ;
        ret                     ; atgriezties no apakšprogrammas
        end
```

Pēc komandas **CALL** izpildīšanas programma pāriet uz komandā norādīto adresi, saglabājot esošo adresi stekā. Apakšprogrammas pēdējā komanda būs **RET**, ar kuru komandu skaitītājs pāries uz adresi, kura glabājas stekā.

Ar komandu **out leds** vai **out 11h** tiks izgaismots uz diodēm binārais skaitlis, kas ir ievietots akumulatorā. Pēc tam komanda pāries uz apakšprogrammu, tiks cikliski samazināts no 8000h līdz 0h ar komandu **DCX**, atņemot no HL satura 1h katrā ciklā (apakšprogramma *delay1*).

Lai novērtētu izveidoto laika aizturi skaitliski, jānoskaidro, cik taktīs tiek izpildīta izsauktā apakšprogramma. Izmantojot datus no komandu sistēmas tabulas, aprēķiniet kopējo takts skaitu.

```
delay:  push    h       ; 12 taktis           x       1
        push    psw     ; 12 taktis           x       1
        out     leds    ; 10 taktis           x       1
delay1: dcx     h       ; sešas taktis        x       32768 (8000h)
        mov     a,h     ; četras taktis       x       32768
        ora     l       ; četras taktis       x       32768
        jnz     delay1  ; 10 taktis           x       32767 (ja Z = 0)
                        ; septiņas taktis     x       1 (ja Z = 1)
        pop     psw     ; 10 taktis           x       1
        pop     h       ; 10 taktis           x       1
        ret             ; 10 taktis           x       1
```

Tādā veidā apakšprogramma izpildās procesora *I8085 N* taktīs:

$$N = 12 + 12 + 10 + 32\,768 \cdot (6 + 4 + 4) + 32\,767 \cdot 10 + 7 + 10 + 10 + 10 = 786\,493$$

Zinot viena procesora takts periodu, var atrast kopējo laika aizturi, sareizinot šo periodu ar takts skaitu *N*. *EMAC* komplektā tiek izmantots ārējais kvarca ģenerators ar frekvenci $F = 6{,}144$ MHz. Tādā gadījumā *I8085* procesora iekšējā takts frekvence būs divreiz mazāka, t. i., 3,072 MHz.

Ja ir zināma procesora takts frekvence, var atrast vienas takts periodu, un tas veido:

$$T = 1 / 3{,}072 \text{ MHz} = 0{,}3255 \cdot 10^{-6} \text{ s}.$$

Tādā gadījumā kopējais aiztures lielums (pie pirmā izsaukšanas) būs vienāds ar:

$$t = TN = 0{,}256 \text{ s}.$$

Turpmākā programmas izpildīšanas laikā laika konstantes (8000h) lielums tiek samazināts par vienu, un laika aizture katrā ciklā arī samazinās par lielumu:

$$\Delta t = (6 + 4 + 4 + 10)T = 7{,}8 \text{ μs}.$$

4.6. Laboratorijas darbs. Kalkulators ar MOS servisiem KEYIN un LEDHEX

Programma saskaita divus heksadecimālos skaitļus, kurus lietotājs ievada ar tastatūru. Pēc pogas nospiešanas atbilstošs HEX cipars parādīsies uz displeja. Pēc otra skaitļa ievadīšanas un pogas **INC** (**Enter**) nospiešanas summēšanas rezultāts parādīsies uz displeja.

```
        org     0ff01h          ; programmas sākumadrese
mos     equ     1000h           ; mainīgam mos tiek piešķirta vērtība 1000h
keyin   equ     0bh             ; mainīgam keyin tiek piešķirta vērtība 0bh
ledhex  equ     12h             ; mainīgam ledhex tiek piešķirta vērtība 12h
start:  mvi     a,0             ; akumulatora un reģistru saturs vienāds ar nulli
        mvi     b,0             ;
        lxi     d,0             ;
        lxi     h,0             ;
        mvi     c,ledhex        ; parādīt DE reģistra saturu (nulli)
        call    mos             ;
rdkey1: mvi     c,keyin         ; nolasīt pogu stāvokli
        call    mos             ;
        mvi     a,0fh           ; pārbaudīt, vai ievadīts heksadecimāls skaitlis
        cmp     L               ;
        jc      rdkey1          ; skaitlis lielāks par 15 - atkārtota ievade
        xchg                    ; atainojam pirmo skaitli
        mvi     c,ledhex        ;
        call    mos             ;
rdkey2: mvi     c,keyin         ; ievada otro skaitli
        call    mos             ;
        mvi     a,0fh           ; pārbauda, vai ievadīts heksadecimālais skaitlis
        cmp     L               ;
        jc      rdkey2          ; skaitlis lielāks par 15 - atkārtota ievade
        xchg                    ;
        mvi     c,ledhex        ; parāda otro skaitli
        call    mos             ;
        mov     b,L             ;
        mvi     c,keyin         ; veic saskaitīšanu
check:  call    mos             ;
        mvi     a,16h           ; pārbauda, vai nospiests taustiņš Enter
        cmp     L               ;
        jnc     check           ; veic atkārtotu ievadi
        mov     L,b             ;
        mvi     h,0             ;
        dad     d               ;
        xchg                    ; parādīt rezultātu
        mvi     c,ledhex        ;
        call    mos             ;
        rst     7               ; atdot kontroli MOS`am
        end
```

Tastatūras aptauja tiek veikta ar servisa **KEYIN** palīdzību. Pēc tā izsaukšanas ar komandu **call mos** programma apstājas un gaida, līdz būs nospiesta tastatūras poga. Pogas vērtība saglabājas L reģistrā, tiek kopēta E reģistrā un ar servisu **LEDHEX** parādās uz displeja. Kad abi skaitļi ir ievadīti, tos saskaita ar komandu **dad d** (HL + DE) un pārvieto rezultātu no HL reģistra pāra uz DE reģistra pāri, lai izgaismotu ar servisu **LEDHEX**.

4.7. Laboratorijas darbs. Muzikālais instruments ar MOS servisiem KEYIN, PITCH un LEDHEX

Programma izdod uz skaļruni signālu ar frekvenci, kas atbilst nospiestiem taustiņiem. Jo lielāks ir skaitlis (no 00H līdz 0DH), jo zemāks ir skaņas tonis. Uz astoņu segmentu indikatoru pāra H7, H6 parādās nospiestās pogas HEX kods. No 00H līdz 0FH, kad nospiestas pogas **0**–**F**, un 14H–17H, kad tiek nospiestas vadības pogas **STP**, **FUNC**, **DEC**, un **INC**. Ar astoņiem slēdžiem var precīzāk izmainīt toni. Jo lielāks ir binārais skaitlis, kuru ievada no DIP slēdžiem, jo zemāks ir skaņas tonis. Skaitlis tiek parādīts indikatoros H5 un H4 HEX kodā, un tiek parādīts ar astoņām diodēm zem slēdžiem binārā kodā (1 — lampiņa deg, 0 — nedeg).

```
mos     equ     1000h    ; mainīgam mos piešķir vērtību 1000h
keyin   equ     0bh      ; mainīgam keyin piešķir vērtību 0bh
dsi     equ     12h      ; mainīgam dsi piešķir vērtību 12h
pitch   equ     10h      ; mainīgam pich piešķir vērtību 10h
ledhex  equ     12h      ; mainīgam ledhex piešķir vērtību 12h
dso     equ     11h      ; mainīgam dso piešķir vērtību 11h
        org     0ff01h   ; programmas sākumadrese
start:  mvi     c,keyin  ; taustiņu aptauja
        call    mos      ;
        mov     d,L      ; ievada taustiņam atbilstošo kodu D reģistrā
        in      dsi      ; ievada slēdžu stāvoklim atbilstošo kodu E reģistrā
        mov     e,a      ;
        mvi     c,pitch  ; izsauc skaļruņa vadības programmu
        call    mos      ;
        mvi     c,ledhex ; parāda vadības signālu uz septiņu segmentu indikatora
        call    mos      ;
        mov     a,e      ; parāda slēdžu informāciju ar diodēm
        out     dso      ;
        call    mos      ;
        jmp     start    ; sākt jaunu taustiņu aptauju
        end
```

5. Studiju darbs

5.1. Studiju darba saturs

Studiju darbā nepieciešams izstrādāt struktūras shēmu mikroprocesora komplektam, kas nodrošina uzdevuma risinājumu, izmantojot mikroprocesoru *I8085* un tā komplektā ietvertās mikroshēmas. Uzdevumu risināšanai jāizmanto apmācības ierīces *Primer*, kas izgatavota firmā *EMAC*, elementi. Pēc attiecīgās struktūras shēmas izveides un apraksta studiju darbā jāveic šādi uzdevumi:

- jāsastāda uzdevuma risināšanas algoritma struktūras shēma;
- jāsastāda programma mikroprocesora *I8085* asamblera valodā;
- jāpārvērš programmu mašīnkodos, izmantojot instrukciju tabulu [1, 2] vai izmantojot speciālu datora krosasamblera programmu (translatoru);
- jāpārbauda sastādītā programma, izmantojot emulatoru *i80*, kā arī apmācības ierīces mikroprocesora sistēmu *Primer*. Mājas darba secinājumos parādīt iegūtos pārbaudes rezultātus.

5.2. Studiju darba uzdevumi

1. Izgaismot uz diviem pirmajiem no labās puses astoņu segmentu indikatoriem skaitli sešpadsmitnieku sistēmā, kas uzdots ar astoņu kārtu bināro kombināciju, izmantojot DIP slēdžus.
2. Izgaismot uz pirmā astoņu segmentu indikatora (H1) skaitli 1, ja ieslēgts DIP pirmais slēdzis, skaitli 2, ja ieslēgts DIP otrais slēdzis, utt. — līdz skaitlim 8, ja ieslēgts DIP astotais slēdzis.
3. Izgaismot uz pirmā astoņu segmentu indikatora (H1) burtu A, ja ieslēgts DIP pirmais slēdzis, burtu B, ja ieslēgts DIP otrais slēdzis un burtu C, ja ieslēgts DIP trešais slēdzis.
4. Uz pirmā astoņu segmentu indikatora (H1), ieslēdzot DIP pirmo slēdzi, jāmirgo burtam S ar frekvenci 1 Hz, bet, ieslēdzot DIP otro slēdzi, jāmirgo burtam L ar frekvenci 0,5 Hz.
5. Ieslēdzot DIP ceturto slēdzi, periodiski jāiedegas pirmā astoņu segmentu indikatora (H1) segmentiem D0, D1, D2, D3, D4, D5, D6, D7, D0 utt. Segmenta degšanas laiks — 1 s.
6. Ieslēdzot DIP astoto slēdzi, periodiski jāiedegas gaismas diodēm LD0, LD1, LD2, LD3, LD4, LD5, LD6, LD7, LD0 utt. Gaismas diodes degšanas laiks — 2 s.
7. Ieslēdzot DIP pirmo slēdzi, jādeg gaismas diodei LD0, ieslēdzot DIP otro slēdzi — gaismas diodei LD1, utt. — līdz diodei LD7, ja ieslēgts DIP astotais slēdzis. Ja ieslēgti vairāki slēdži, nedeg neviena gaismas diode.
8. Ieslēdzot DIP pirmo slēdzi, uz pirmā astoņu segmentu indikatora (H1) jāizgaismo cipars 0, bet, ieslēdzot DIP otro slēdzi, jāmirgo gaismas diodei LD0 ar frekvenci 2 Hz.

9. Nodrošināt skaitļu no 0 līdz F indikāciju uz pirmā astoņu segmentu indikatora (H1). Katra skaitļa indikācijas sākums tiek uzdots, ieslēdzot DIP pirmo slēdzi, bet beigas — izslēdzot to.
10. Nodrošināt skaitļu no 0 līdz 9 periodisku indikāciju uz pirmā astoņu segmentu indikatora (H1). Katra skaitļa indikācijas laiks ir vienāds ar 1 s.
11. Ja nav ieslēgts neviens no DIP slēdžiem, gaismas diodes nedeg, ja ir ieslēgts kāds no DIP slēdžiem, jādeg gaismas diodei LD0, ja divi — gaismas diodei LD1, utt. — līdz gaismas diodei LD7, ja ir ieslēgti visi astoņi DIP slēdži.
12. Ieslēdzot DIP pirmo slēdzi, periodiski uz vienu sekundi jāiedegas gaismas diodēm LD0, LD1, LD0 utt., bet, ja ieslēgts DIP astotais slēdzis — gaismas diodēm LD7, LD6, LD7 utt.
13. Nodrošināt astoņu kārtu binārās kombinācijas, kas uzdota ar astoņiem DIP slēdžiem, vienīnieka skaita indikāciju uz pirmā astoņu segmentu indikatora (H1).
14. Noteikt, cik reizes ieslēgts un pēc tam izslēgts DIP pirmais slēdzis (līdz septiņām reizēm), rezultātu attēlot uz pirmā astoņu segmentu indikatora (H1) pēc DIP otrā slēdža ieslēgšanas.
15. Nodrošināt segmenta izgaismošanu, kura numuru veido binārais kods, kas uzdots ar DIP slēdžiem 1, 2, 3 uz pirmā astoņu segmentu indikatora (H1).
16. Noteikt laika intervālu sekundēs intervālā no 0 līdz 15 sekundēm. Intervāla sākums — taustiņa **STP** nospiešana, intervāla beigas — taustiņa **FUNC** nospiešana. Pēc mērījumu beigām padot skaņas signālu. Mērījumu rezultātu parādīt uz pirmā astoņu segmentu indikatora (H1) heksadecimālā kodā.
17. Noteikt laika intervālu sekundēs (līdz 99 s). Intervāla sākums — taustiņa **STP** nospiešana, intervāla beigas — taustiņa **FUNC** nospiešana. Pēc mērījumu beigām padot skaņas signālu. Mērījumu rezultātu BCD kodā ierakstīt atmiņas šūnā FF80H.
18. Noteikt, cik reizes nospiests taustiņš **STP** (līdz 15 reizēm). Rezultātu parādīt uz sestā astoņu segmentu indikatora (H6) pēc taustiņa **FUNC** nospiešanas.
19. Nospiežot taustiņu **STP**, uz pirmā un otrā astoņu segmentu indikatoriem (H1 un H2) jāparādās cipariem AAh. Nospiežot taustiņu **FUNC**, jāparādās cipariem 00h. Nospiežot jebkuru citu taustiņu, indikatori tiek dzēsti un tiek padots skaņas signāls.
20. Ja nospiests taustiņš **STP**, uz pirmā astoņu segmentu indikatora (H1) jāmirgo skaitlim 5 ar frekvenci 1 Hz. Ja nospiests taustiņš **FUNC**, uz tā paša indikatora jāmirgo skaitlim 0 ar frekvenci 5 Hz. Nospiežot jebkuru citu taustiņu, indikators tiek dzēsts un tiek padots skaņas signāls.
21. Nodrošināt periodisku burta A mirgošanu uz pirmā astoņu segmentu indikatora (H1), burta P mirgošanu uz trešā indikatora (H3) un burta E mirgošanu uz sestā indikatora (H6) ar periodu 0,5 s.
22. Nodrošināt skaitļa 5 periodisko izgaismošanu uz katra no astoņu segmentu indikatoriem (no H1 līdz H6) pēc kārtas. Skaitlis 5 tiek izgaismots vienu sekundi, pēc tam vienu sekundi netiek utt.

23. Doto bināro skaitli (robežās no 00H līdz 63H), kas ievietots atmiņas šūnā FF80H, pārvērst bināri decimālā kodā (BCD). Rezultātu parādīt uz pirmā un otrā astoņu segmentu indikatoriem (H1 un H2).
24. Atmiņas šūnā FF80H doto bināro skaitli parādīt uz pirmā un otrā astoņu segmentu indikatoriem (H1 un H2) heksadecimālā skaitīšanas sistēmā.
25. Pārvērst bināri decimālā kodā uzdotu datu baitu, kas ievietots atmiņas šūnā FF80H, binārā kodā un ievietot atmiņas šūnā FF81H.
26. Uz pirmā un otrā astoņu segmentu indikatoriem (H1 un H2) nodrošināt periodisku sekunžu indikāciju no 00 līdz 99. Sasniedzot skaitli 99, jāpadod skaņas signāls un jāsāk indikācija no jauna (no stāvokļa 00).
27. Nospiežot taustiņu **STP**, periodiski jāiedegas segmentiem D0, D1, D2, D3, D4, D5, D6, D7, D0 utt. uz pirmā astoņu segmentu indikatora (H1). Segmenta degšanas laiks — 1 s.
28. Nodrošināt skaitļu no 9 līdz 0 periodisku indikāciju uz sestā astoņu segmentu indikatora (H6). Katra skaitļa indikācijas laiks vienāds ar 1 s. Sasniedzot 0, tiek padots skaņas signāls, un process tiek atkārtots.
29. Doto bināro skaitli (robežās no 0 līdz 255), kas ievietots atmiņas šūnā FF80H, pārvērst bināri decimālā kodā (BCD). Rezultātu ievietot atmiņas šūnās FF80H un FF81H (vecākais baits tiek ievietots atmiņas šūnā FF81H, jaunākais baits — FF80).

5.3. Laboratorijas ierīces *Primer* servisi

Laboratorijas ierīcē *Primer* tiek izmantotas vadības programma (MOS vai monitors), kas nodrošina perifērijas ierīču vadību, laika aizturi un citu bieži lietotu procedūru izpildi. Šajā gadījumā nav jāveic interfeisa programmēšana, bet var izmantot vadības programmu, ievadot servisa numuru C reģistrā (dažos gadījumos citos reģistros jāievada nepieciešamā informācija).

SERVICE A DIPSWIN

Visu astoņu DIP slēdžu pozīciju stāvokļu nolasīšana.

JĀIEVADA: C reģistrā — 0AH vērtība.

REZULTĀTS: nolasītais kods no DIPS slēdžiem L reģistrā.

SERVICE B KEYIN

Nospiestā tastatūras taustiņa koda nolasīšana. Notiek gaidīšana, līdz tiks nospiests kāds no taustiņiem, un nospiestā taustiņa koda ierakstīšana L reģistrā.

JĀIEVADA: C reģistrā — 0BH vērtība.

REZULTĀTS: nospiestā taustiņa kods L reģistrā. Taustiņiem **0**–**F** tiek piešķirts kods 00H–0FH attiecīgi, bet taustiņiem **STP**, **FUNC**, **DEC** un **INC** attiecīgi kodi 14H–17H.

SERVICE C PTAOUT

Porta A izvade. Notiek datu padošana uz ciparu izvades portu A (gaismas diodes un CN3 savienotājs).

JĀIEVADA: C reģistrā — 0CH vērtība;
E reģistrā — astoņu bitu kods, kuru padod uz portu A.

REZULTĀTS: attiecīgo pozīciju gaismas diožu ieslēgšana.

SERVICE 10 PITCH

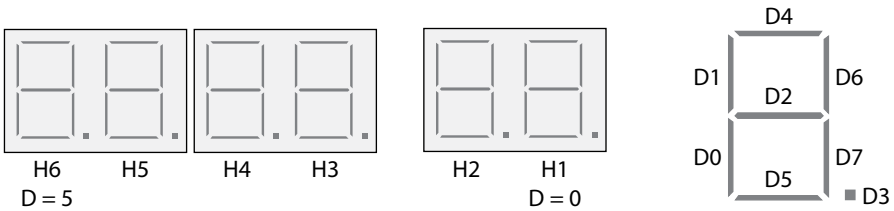
Skaņas izvade. Notiek DE reģistru pāra satura 14 jaunāko bitu padošana (divi vecākie biti tiek ignorēti) uz skaļruņa taimeri. Jo lielākais ir šis skaitlis, jo zemāks ir tonis. Ja DE = 0, skaļrunis tiek atslēgts.

JĀIEVADA: C reģistrā — 10H vērtība;
DE reģistru pārī — 14 bitu toņa vērtība.

REZULTĀTS: skaņa ar uzdotu toni.

SERVICE 11 LEDOUT

Astoņu segmentu indikatora izgaismošana. Notiek E reģistrā uzdotās LED segmentu secības izgaismošana uz astoņu segmentu indikatora, kura numurs uzdots D reģistrā. Indikatori tiek numurēti no 0 līdz 5 (ejot no labās puses uz kreiso).



Segmentu D0, D1, D2, D3, D4, D5, D6, D7 iedegšana notiek, uzstādot bitu attiecīgajā E reģistra pozīcijā.

JĀIEVADA: C reģistrā — 11H vērtība;
E reģistrā — ieslēgto segmentu secība;
D reģistrā — astoņu segmentu indikatora numurs.

REZULTĀTS: iedegās uzdotie segmenti uz norādītā astoņu segmentu indikatora.

SERVICE 12 LEDHEX

DE reģistru pāra satura izvade uz četriem kreisajiem (no H6 līdz H3) astoņu segmentu indikatoriem heksadecimālā formātā.

JĀIEVADA: C reģistrā — 12H vērtība;
DE reģistru pārī — 16 bitu skaitlis heksadecimālā skaitļošanas sistēmā.

REZULTĀTS: norādītais skaitlis tiek attēlots uz četriem kreisajiem indikatoriem.

SERVICE 13 LEDDEC

DE reģistru pāra satura izvade uz četriem kreisajiem (no H6 līdz H3) astoņu segmentu indikatoriem decimālā formātā. Maksimālais skaitlis, kas tiek parādīts uz indikatoriem, ir 9999.

JĀIEVADA: C reģistrā — 13H vērtība;
DE reģistru pārī — rādāmais skaitlis decimālā skaitļošanas sistēmā.
REZULTĀTS: norādītais skaitlis tiek attēlots uz četriem kreisajiem indikatoriem.

SERVICE 14 DELAY

Laika aizture, kas atbilst HL reģistru pāra ierakstītajai vērtībai. Jo lielāka ir šī vērtība, jo ilgāka ir aizture.

JĀIEVADA: C reģistrā — 14H vērtība;
HL reģistru pārī — laika aiztures konstante.
REZULTĀTS: tiek veidota laika aizture, kas atbilst norādītajai konstantei.

SERVICE 15 PTBIN

Nolasīt inversās porta B vērtības. Līdzīgi servisam **DIPSWIN** — tikai ar slēdžu bitu inversiju.

JĀIEVADA: C reģistrā — 15H vērtība.
REZULTĀTS: L reģistrā — invertēti dati, kas padoti uz portu B (DIP slēdži).

SERVICE 16 KEYSTAT

Nolasīt tastatūras stāvokli HL reģistru pārī. Ja ir nospiests kāds no taustiņiem, H reģistrā būs 01H vērtība, bet L reģistrā — nospiestā taustiņa kods. Pretējā gadījumā HL ir vienāds ar 0. Taustiņiem no **0** līdz **F** tiek piešķirts attiecīgais kods 00H–0FH, bet taustiņiem **STP**, **FUNC**, **DEC** un **INC** attiecīgi 14H–17H.

JĀIEVADA: C reģistrā — 16H vērtība.
REZULTĀTS: HL reģistru pārī — tastatūras stāvokļa kods.

SERVICE 17 DIGOUT

Parādīt heksadecimālo ciparu, kas glabājas E reģistrā uz astoņu segmentu indikatora, kura numurs norādīts D reģistrā. E reģistra saturam jābūt mazākam par 10H. D reģistra vērtībai ir jābūt robežās no 0 līdz 5, kur 0 atbilst pirmais indikators no labās puses (H1), bet 5 — pirmais indikators no kreisās puses (H6).

JĀIEVADA: C reģistrā — 17H vērtība;
D reģistrā — astoņu segmentu indikatora numurs;
E reģistrā — cipars heksadecimālā skaitļošanas sistēmā.
REZULTĀTS: uzdotais cipars iedegas uz norādītā astoņu segmentu indikatora.

5.4. Mikroprocesora *I8085* komandu sistēma

Mneimonika	Taktis	Atšifrējums	Piezīmes
ACI n	7	Add with Carry Immediate	A = A + n + CY
ADC r	4	Add with Carry	A = A + r + CY
ADC M	7	Add with Carry to Memory	A = A + [HL] + CY
ADD r	4	Add	A = A + r
ADD M	7	Add to Memory	A = A + [HL]
ADI n	7	Add Immediate	A = A + n
ANA r	4	AND Accumulator	A = A & r
ANA M	7	AND Accumulator and Memory	A = A & [HL]
ANI n	7	AND Immediate	A = A & n
CALL a	18	Call unconditional	[SP] = [SP] – 1
[SP] = PC, PC = a			
CC a	9/18	Call on Carry	Ja CY = 1, 18 taktis
CM a	9/18	Call on Minus	Ja S = 1, 18 taktis
CMA	4	Complement Accumulator	A = ~A
CMC	4	Complement Carry	CY = ~CY
CMP r	4	Compare	A – r
CMP M	7	Compare with Memory	A – [HL]
CNC a	9/18	Call on No Carry	Ja CY = 0, 18 taktis
CNZ	9/18	Call on No Zero	Ja Z = 0, 18 taktis
CP a	9/18	Call on Plus	Ja S = 0, 18 taktis
CPE a	9/18	Call on Parity Even	Ja P = 1, 18 taktis
CPI n	7	Compare Immediate	A – n
CPO a	9/18	Call on Parity Odd	Ja P = 0, 18 taktis
CZ a	9/18	Call on Zero	Ja Z = 1, 18 taktis
DAA	4	Decimal Adjust Accumulator	A = BCD formāts
DAD B	10	Double Add BC to HL	HL = HL + BC
DAD D	10	Double Add DE to HL	HL = HL + DE
DAD H	10	Double Add HL to HL	HL = HL + HL
DAD SP	10	Double Add SP to HL	HL = HL + SP
DCR r	4	Decrement	r = r – 1
DCR M	10	Decrement Memory	[HL] = [HL] – 1
DCX B	6	Decrement BC	BC = BC – 1
DCX D	6	Decrement DE	DE = DE – 1
DCX H	6	Decrement HL	HL = HL – 1
DCX SP	6	Decrement Stack Pointer	SP = SP – 1
DI	4	Disable Interrupts	
EI	4	Enable Interrupts	
HLT	5	Halt	
IN p	10	Input	A = [p]
INR r	4	Increment	r = r + 1
INR M	10	Increment Memory	[HL] = [HL] + 1
INX B	6	Increment BC	BC = BC + 1
INX D	6	Increment DE	DE = DE + 1

INX H	6	Increment HL	HL = HL + 1
INX SP	6	Increment Stack Pointer	SP = SP + 1
JMP a	7	Jump unconditional	PC = a
JC a	7/10	Jump on Carry	Ja CY = 1, 10 taktis
JM a	7/10	Jump on Minus	Ja S = 1, 10 taktis
JNC a	7/10	Jump on No Carry	Ja CY = 0, 10 taktis
JNZ a	7/10	Jump on No Zero	Ja Z = 0, 10 taktis
JP a	7/10	Jump on Plus	Ja S = 0, 10 taktis
JPE a	7/10	Jump on Parity Even	Ja P = 1, 10 taktis
JPO a	7/10	Jump on Parity Odd	Ja P = 0, 10 taktis
JZ a	7/10	Jump on Zero	Ja Z = 1, 10 taktis
LDA a	13	Load Accumulator direct	A = [a]
LDAX B	7	Load Accumulator indirect	A = [BC]
LDAX D	7	Load Accumulator indirect	A = [DE]
LHLD a	16	Load HL Direct	HL = [a]
LXI B,nn	10	Load Immediate BC	BC = nn
LXI D,nn	10	Load Immediate DE	DE = nn
LXI H,nn	10	Load Immediate HL	HL = nn
LXI SP,nn	10	Load Immediate Stack Pointer	SP = nn
MOV r1,r2	4	Move register to register	r1 = r2
MOV M,r	7	Move register to Memory	[HL] = r
MOV r,M	7	Move Memory to register	r = [HL]
MVI r,n	7	Move Immediate	r = n
MVI M,n	10	Move Immediate to Memory	[HL] = n
NOP	4	No Operation	
ORA r	4	Inclusive OR Accumulator	A = A v R
ORA M	7	Inclusive OR Accumulator	A = A v [HL]
ORI n	7	Inclusive OR Immediate	A = A v n
OUT p	10	Output	[p] = A
PCHL	6	Jump HL indirect	PC = [HL]
POP B	10	Pop BC	BC = [SP] + 1
POP D	10	Pop DE	DE = [SP] + 1
POP H	10	Pop HL	HL = [SP] + 1
POP PSW	10	Pop Processor Status Word	{PSW,A} = [SP] + 1
PUSH B	12	Push BC	[SP] – 1 = BC
PUSH D	12	Push DE	[SP] – 1 = DE
PUSH H	12	Push HL	[SP] – 1 = HL
PUSH PSW	12	Push Processor Status Word	[SP] – 1 = {PSW,A}
RAL	4	Rotate Accumulator Left	A = {CY,A} ←
RAR	4	Rotate Accumulator Right	A = → {CY,A}
RET	10	Return	[SP] = [SP] + 1
PC = [SP]			
RC	6/12	Return on Carry	Ja CY = 1, 12 taktis
RIM	4	Read Interrupt Mask	A = maska
RM	6/12	Return on Minus	Ja S = 1, 12 taktis
RNC	6/12	Return on No Carry	Ja CY = 0, 12 taktis

RNZ	6/12	Return on No Zero	Ja Z = 0, 12 taktis
RP	6/12	Return on Plus	Ja S = 0, 12 taktis
RPE	6/12	Return on Parity Even	Ja P = 1, 12 taktis
RPO	6/12	Return on Parity Odd	Ja P = 0, 12 taktis
RZ	6/12	Return on Zero	Ja Z = 1, 12 taktis
RLC	4	Rotate Left Circular	A = A ←
RRC	4	Rotate Right Circular	A = → A
RST z	12	Restart	[SP] = [SP] – 1;
[SP] = PC, PC = z			
SBB r	4	Subtract with Borrow	A = A – r – CY
SBB M	7	Subtract with Borrow	A = A – [HL] – CY
SBI n	7	Subtract with Borrow Immediate	A = A – n – CY
SHLD a	16	Store HL Direct	[a] = HL
SIM	4	Set Interrupt Mask	maska = A
SPHL	6	Move HL to SP	SP = HL
STA a	13	Store Accumulator	[a] = A
STAX B	7	Store Accumulator indirect	[BC] = A
STAX D	7	Store Accumulator indirect	[DE] = A
STC	4	Set Carry	CY = 1
SUB r	4	Subtract	A = A – r
SUB M	7	Subtract Memory	A = A – [HL]
SUI n	7	Subtract Immediate	A = A – n
XCHG	4	Exchange HL with DE	HL ↔ DE
XRA r	4	Exclusive OR Accumulator	A = A ⊕ r
XRA M	7	Exclusive OR Memory	A = A ⊕ [HL]
XRI n	7	Exclusive OR Immediate	A = A ⊕ n
XTHL	16	Exchange stack Top with HL	[SP] ↔ HL

Izmantotā literatūra

1. Klūga, A. *Mikroprocesori un mikroprocesoru sistēmas*. Mācību līdzeklis. Rīga: RTU Izdevniecība, 2007. 152 lpp.
2. Klūga, A. *Metodiskie norādījumi studiju darba izpildei mācību priekšmetā "Transporta mikroprocesoru sistēmas"*. Rīga: RTU Izdevniecība, 2005. 16 lpp.

II. Mikrokontrollera *MCS-51* lietošana

1. Vispārīgā informācija par *MTS-51* sistēmu

MTS-51 ir mikrokontrollera sistēmas stends (*microcomputer trainer system*), kas veidots vienkristāla mikrokontrollera *MCS-51* apgūšanai. Dažādu uzdevumu risināšanai un daudzpusīgai *MCS-51* apgūšanai stendā ir uzstādītas daudzas ievadizvades ierīces.

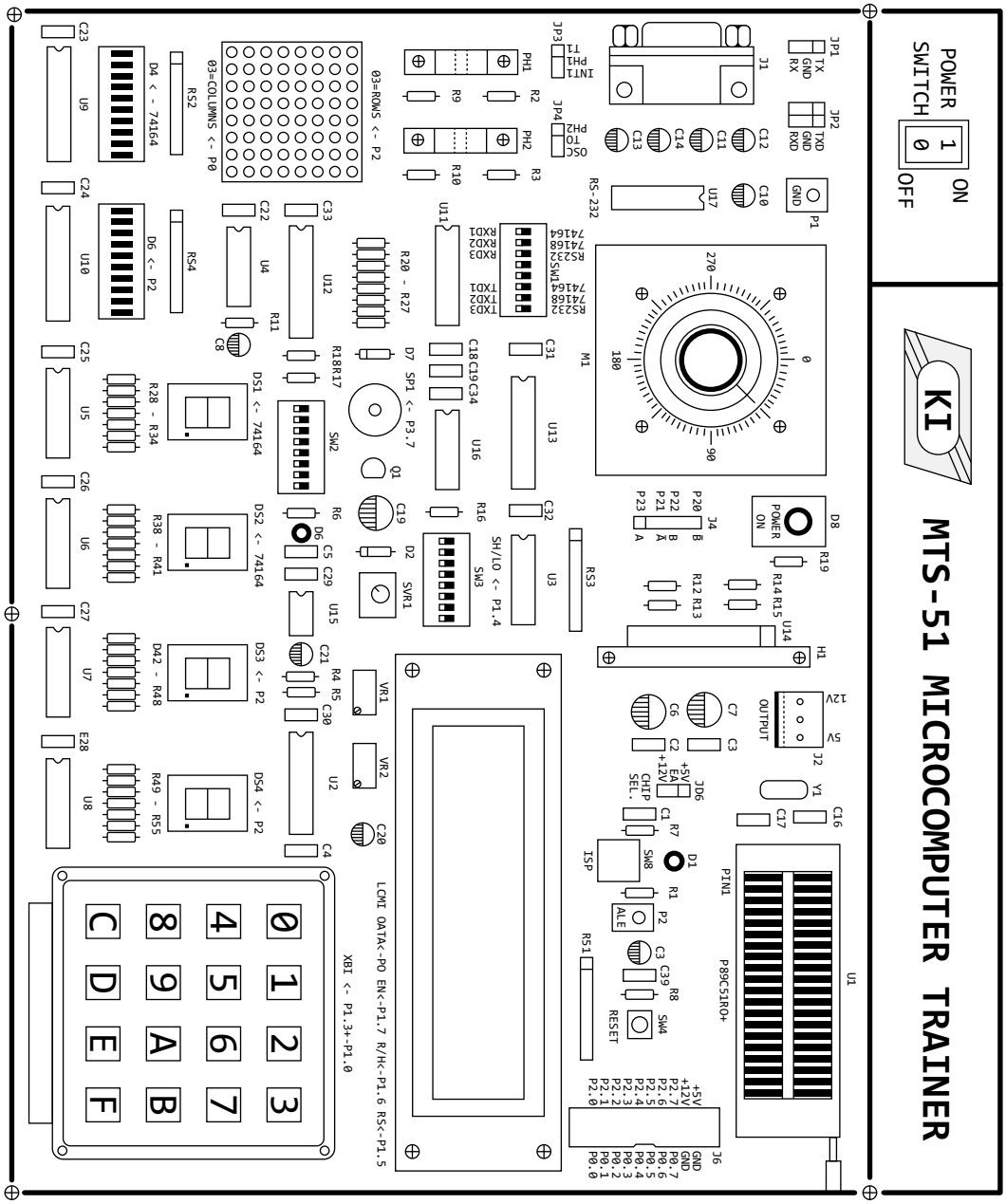
MTS-51 raksturojums:

1. Tiek izmantots firmas *Philips* mikrokontrolleris *P89C51RD+/P89C51RD2*, kas paredzēts programmēšanai, neatvienojot to no visas sistēmas (ISP — *in system programming*). Ierīcē iebūvēts barošanas bloks.
2. Salīdzinājumā ar parastajām apmācības metodēm ISP mikrokontrolleris dod iespēju labot programmu reāllaikā atšķirībā no mikroshēmu simulācijas vai programmatora izmantošanas.
3. Daudzpusīgs ievadizvades ierīču sastāvs, kas paredzēts to izmantošanai apmācībā, lietojot vadībai *MCS-51* sērijas mikrokontrolleri.
4. Virknes interfeisa savienotājs un *RS-232* interfeiss, kas dod iespēju komunicēt ar *IBM* savietojamu personālo datoru vai arī ar citām mikroprocesoru iekārtām daudzprocesoru sistēmu izveidošanai.
5. Ir lietoti ievadizvades portu savienotāji ierīces iespēju paplašināšanai.
6. Ierīcei ir moduļa struktūra, kas dod iespēju ar slēdžiem izvēlēties tikai tās ievades/izvades ierīces, kas tiek izmantotas programmā.
7. Aprīkots ar astoņu segmentu indikatoriem (ar decimālo dekodētāju) un gaismas diodēm programmas izpildes novērošanai.

Lietotājs var apgūt šādas *MCS-51* programmēšanas iemaņas:

- bināro skaitļa atainošanu;
- gaismas diožu ieslēgšanu;
- soļa dzinēja vadību;
- šķidro kristālu displeja/moduļa (LCM) vadību;
- diožu matricas (8 × 8) vadību;
- tastatūras vadību ievades operāciju veikšanai;
- skaļruņa vadību;
- *MCS-51* virknes ievadizvades porta paplašināšanu;
- pārtraukumu vadību, izmantojot ārējos notikumus;
- impulsu skaitīšanu;

- notikumu skaitīšanu;
- ievadi par DIP slēdžu stāvokli,
- *MCS-51* komunikāciju ar personālo datoru;
- *MCS-51* komunikāciju ar citu *MCS-51* ierīci;
- daudzprocesoru komunikāciju.



2.1. att. *MTS-51* priekšējais panelis.

MTS-51 paredzēts arī mikroshēmu sērijām *Intel 8751/52*, *Atmel AT89C51/52*, *Philips P89C51RX+/P89C51RX2*, *Atmel T89C51RC2/T89C51RD2*. Gadījumā, ja tiek izmantotas *Intel 8751/52* un *Atmel AT89C51/52* sēriju mikroshēmas, nepieciešams attiecīgā mikrokontrollera programmators šo mikroshēmu programmēšanai.

2.1. attēlā parādīts *MTS-51* ierīces priekšējais panelis.

2. Ievadizvades porti un ievadizvades ierīces

P0 tiek izmantots datu šķidro kristālu displeja (LCM) datu kopnei un diožu matricas kolonnas signālam;

P1 P1.0–P1.3 tiek izmantoti ievadei ar tastatūru,
P1.4 tiek izmantots nobīdei/ielādei (SH/LD) ar mikroshēmu *74165*,
P1.5, P1.6, P1.7 tiek izmantoti vadības signālu RS, R/W un EN formēšanai LCM indikatoram;

P2 tiek izmantots informācijas parādīšanai uz gaismas diodēm un astoņu segmentu indikatoriem, kā arī soļa dzinēja vadībai un diožu matricas rindas signāliem;

P3 tiek izmantots virknes porta vadībai, *74164*, *74165* un *RS-232* interfeisiem;

INT0 pārtraukumu signāls tiek izmantots, ievadot informāciju ar tastatūru;

INT1 fotopārtraucēja pārtraukumu signāls;

T0 pirmā fotopārtraucēja vai impulsu izeja;

T1 otrā fotopārtraucēja izeja;

P3.7 skaļruņa vadības signāls.

2.1. attēlā ir parādīts *MTS-51* sistēmas priekšējais panelis, kur atrodas šādas vadības ierīces un slēdži:

JP1 *RS-232* raidītāja un uztvērēja (RX un TX) izvadi, tas pats, kas J1;

JP2 *8051* raidītāja un uztvērēja (RXD un TXD) izvadi, paredzēti komunikācijai ar citu *8051* ierīci;

JP3 PH1 izejas signāla izvēle, pie tās var pieslēgt INT1 vai T1;

JP4 T0 ieejas komutators, pie tās var pieslēgt OSC vai PH2;

JP5 ISP sprieguma izvēle 12 V vai 5 V:
P89C51RD+ ISP spriegums ir +12 V,
P89C51RD2 un *T89C51RD2* ISP spriegums ir +5 V;

SW1 savieno *8051* virknes portu ar *74164*, *74165* vai *RS-232*;

SW2 komponenšu barošanas slēdžu bloks;

SVR1 impulsu frekvences pieskaņošana;

VR1 LCM kontrasta pieskaņošana;

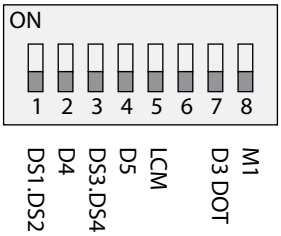
VR2 LCM spilgtuma pieskaņošana;

SW5 ISP darbības režīms. Programmēšanas režīmā šim slēdzim jābūt nospiestam (**ON**). Programmas izpildīšanas režīmā šīs slēdzis ir atlaists (**OFF**);

SW4 poga **Reset** — pārstartē iekārtu;

J6 tiek savienots ar P0 un P2 visām astoņām līnijām, kā arī iekļauj sevī barošanas avota terminālus. Paredzēts ierīces paplašināšanai.

Visas ievadizvades ierīces, kas atrodas uz priekšēja paneļa, pirms to izmantošanas jāsavieno ar barošanas avotu. Šim nolūkam ir paredzēti astoņi DIP slēdži (SW2), kas parādīti 2.2. attēlā. Ar to palīdzību izvēles kārtībā var atļaut/izslēgt katru no ierīcēm. Uz *MTS-51* iekārtas zem katra slēdža ir uzrakstīts, par ko tas atbild.



2.2. att. Ievadizvades ierīču barošanas slēdži SW2.

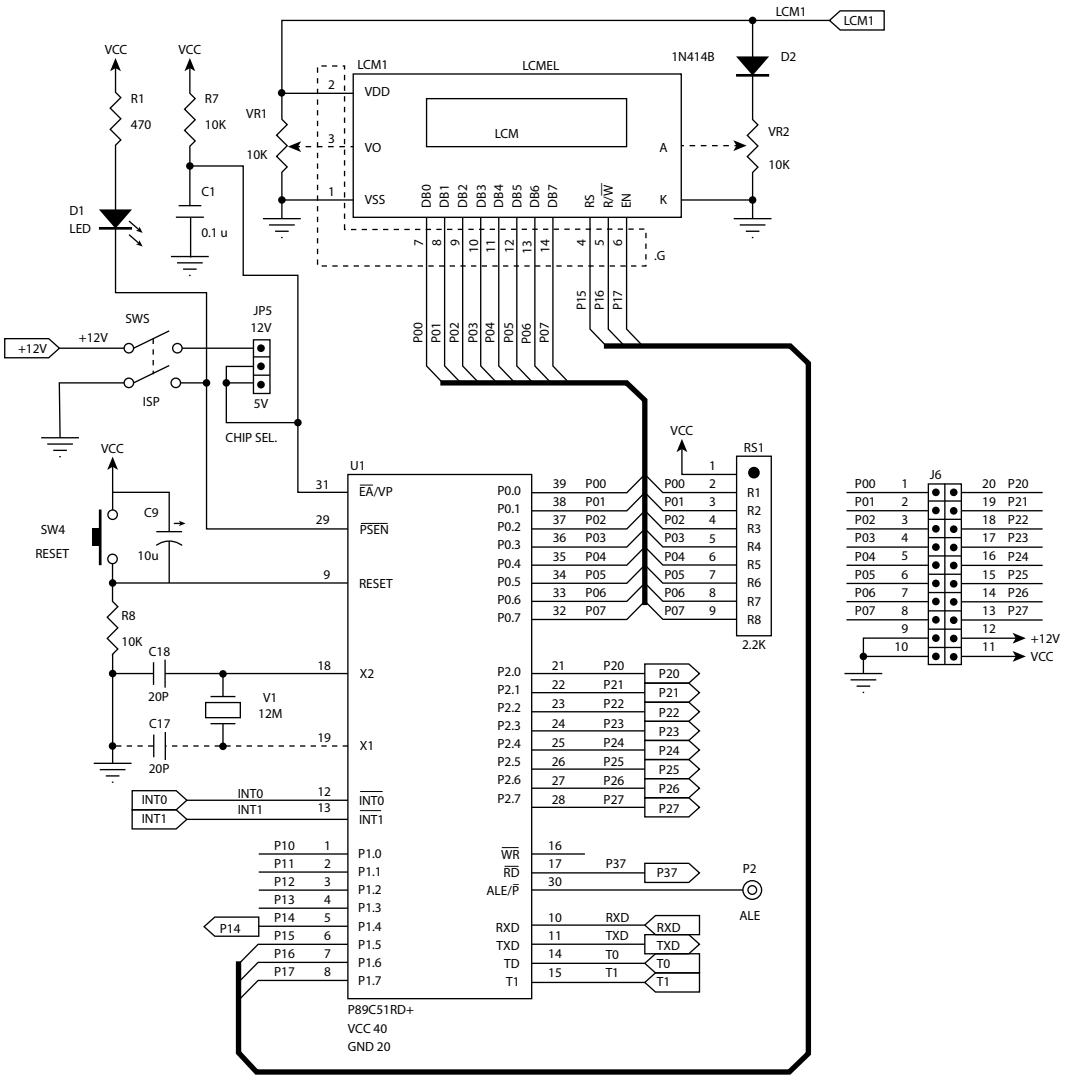
Slēdža SW2 funkcijas un paskaidrojums (skat. 2.2. attēlu):

1. ***DS1.DS2*** — astoņu segmentu indikatori, pie mikrokontrollera tiek pieslēgti, izmantojot *74164* virknes interfeisu.
2. ***D4*** — diožu pakete, pie mikrokontrollera tiek pieslēgta, izmantojot *74164* virknes interfeisu.
3. ***DS3.DS4*** — astoņu segmentu indikatori, tiek savienoti ar mikrokontrollera otrā porta P2 līnijām (paralēli ir BCD dekodētāji kopā ar indikatoriem).
4. ***D5*** — diožu pakete, tiek savienota ar mikrokontrolleris otrā porta P2 līnijām (paralēli).
5. ***LCM*** — šķidro kristālu moduļa barošana.
6. Netiek izmantots
7. ***D3 DOT*** — diožu matricas barošanas avota pieslēgšana.
8. ***M1*** — soļa dzinēja barošanas pieslēgšana.

3. *MTS-51* shēmu apraksts

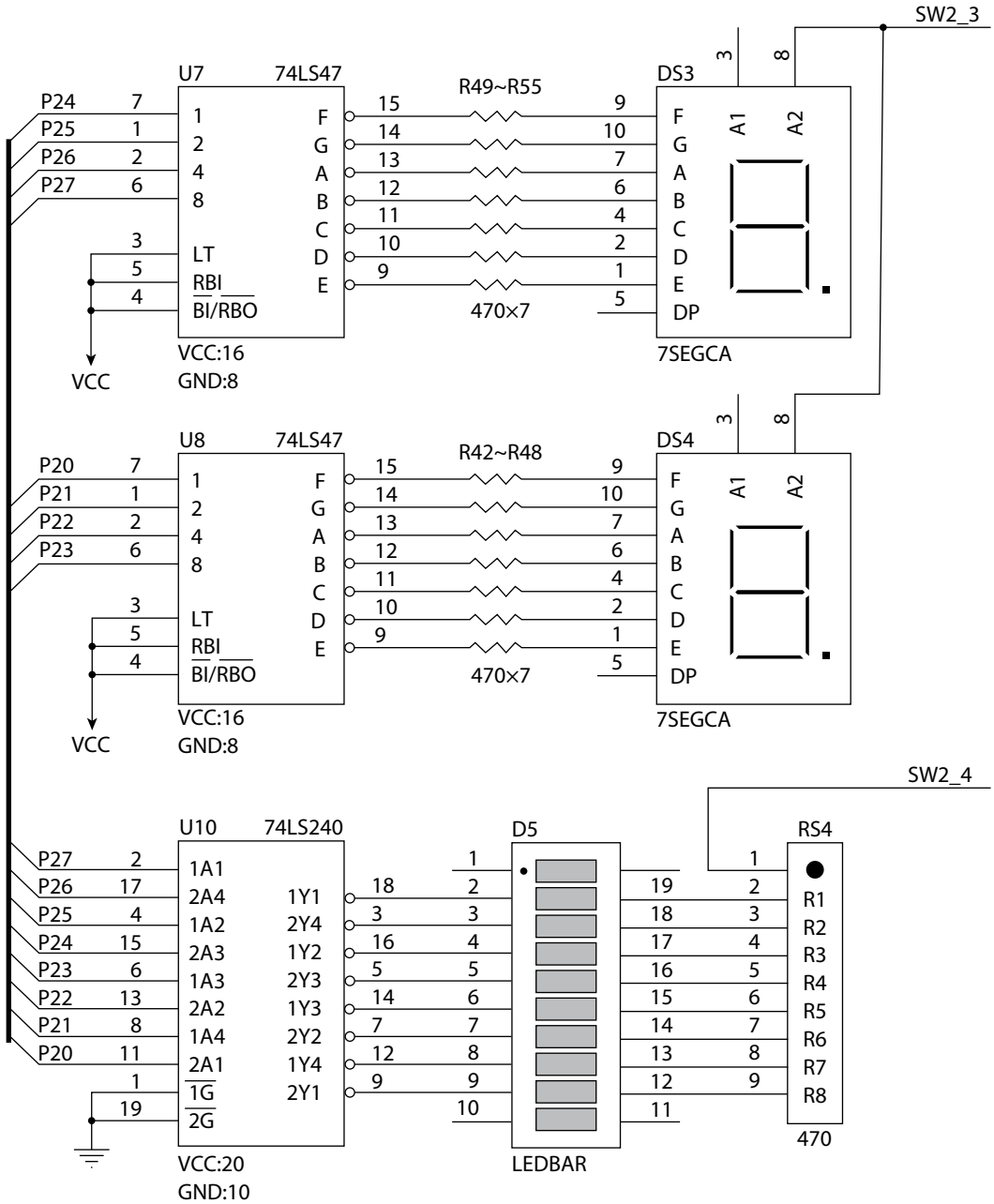
Apmācības ierīce *MTS-51* sastāv no firmas *Philips* mikrokontrollera *P89C51RD+*, kas ir identisks *MCS-51* mikrokontrollerim, ko ražo kompānija *Intel*. Kontrollera ar vadības ķēdēm un šķidro kristālu indikatora LCM shēma parādīta 2.3. attēlā. Nospiežot slēdzi SW4, mikrokontrolleris atrodas U1 sākuma stāvoklī, bet SW5 sagatavo mikrokontrolleri

programmēšanas režīmam (padod "0" spriegumu uz PSEN ieeju), diode D1 signalizē par programmas ierakstu.



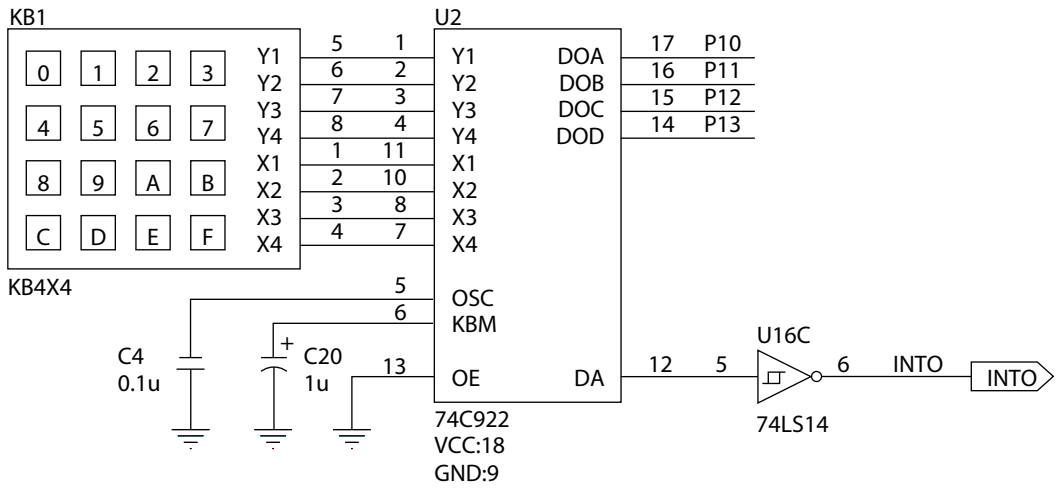
2.3. att. Kontrollera ar vadības ķēdēm un šķidro kristālu indikatora LCM shēma.

Pie mikrokontrollera otrā porta izvadiem pieslēgti divi septiņu segmentu indikatori DS3 un DS4 ar dekoderiem U7 un U8 un astoņas gaismas diodes D5 ar invertējošu buferi U10 (skat. 2.4. attēlu). Visi rezistori R1–R8 blokā RS4 satur kopēju pievadu, uz kuru tiek padots +5 V spriegums no slēdža SW2 ceturtā kontakta.



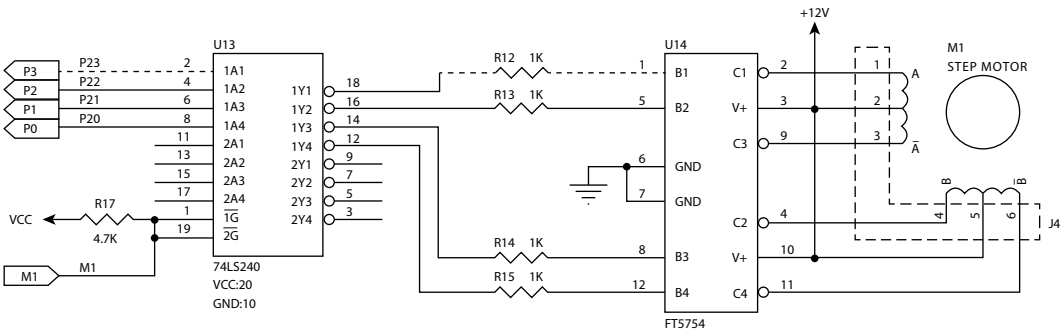
2.4. att. Pie kontrollera otrā porta pieslēgtās ierīces.

Informācijas ievadei var tikt izmantoti 16 taustiņi no **0** līdz **F** (skat. 2.5. attēlu, ierīce KB1). Ar 4 × 4 kodētāja U2 palīdzību tiek veidots taustiņu binārais kods, kas tiek padots uz mikrokontrollera pirmo portu. Nospiežot taustiņu, veidojas arī pārtraukuma pieprasījuma signāls INT0.



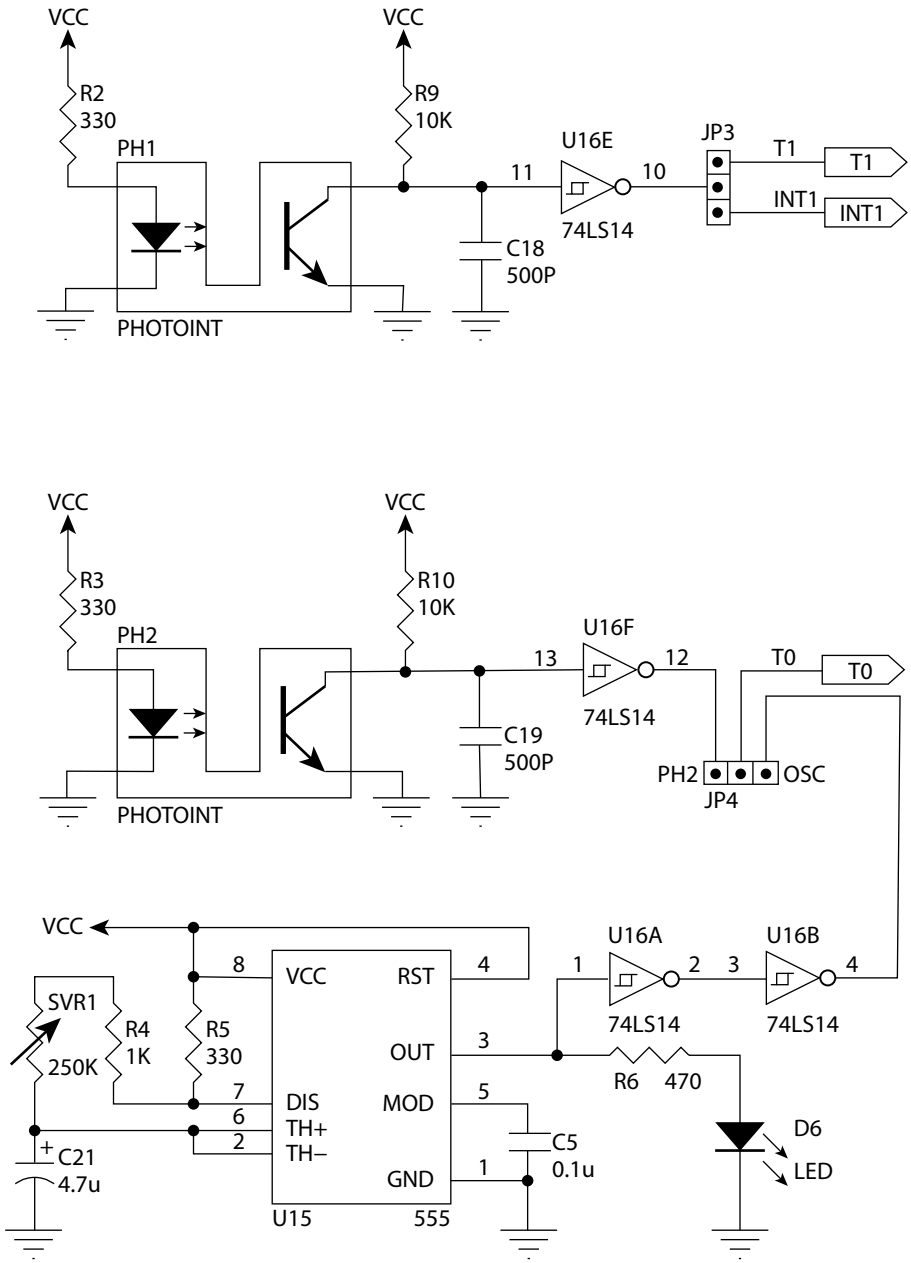
2.5. att. Taustiņu bloka un kodētāja shēma.

Apmācības ierīcē *MTS-51* ir iespēja izmantot soļu dzinēju (*step motor*) M1 un vadīt tā griešanās ātrumu un virzienu (skat. 2.6. attēlu). Dzinējs pieslēgts pie jaudas pastiprinātāja U14, kas savukārt izmanto invertējošā bufera U13 izejas signālus. Savukārt buferis pieslēgts pie otrā porta četriem izvadiem (P20–P23).



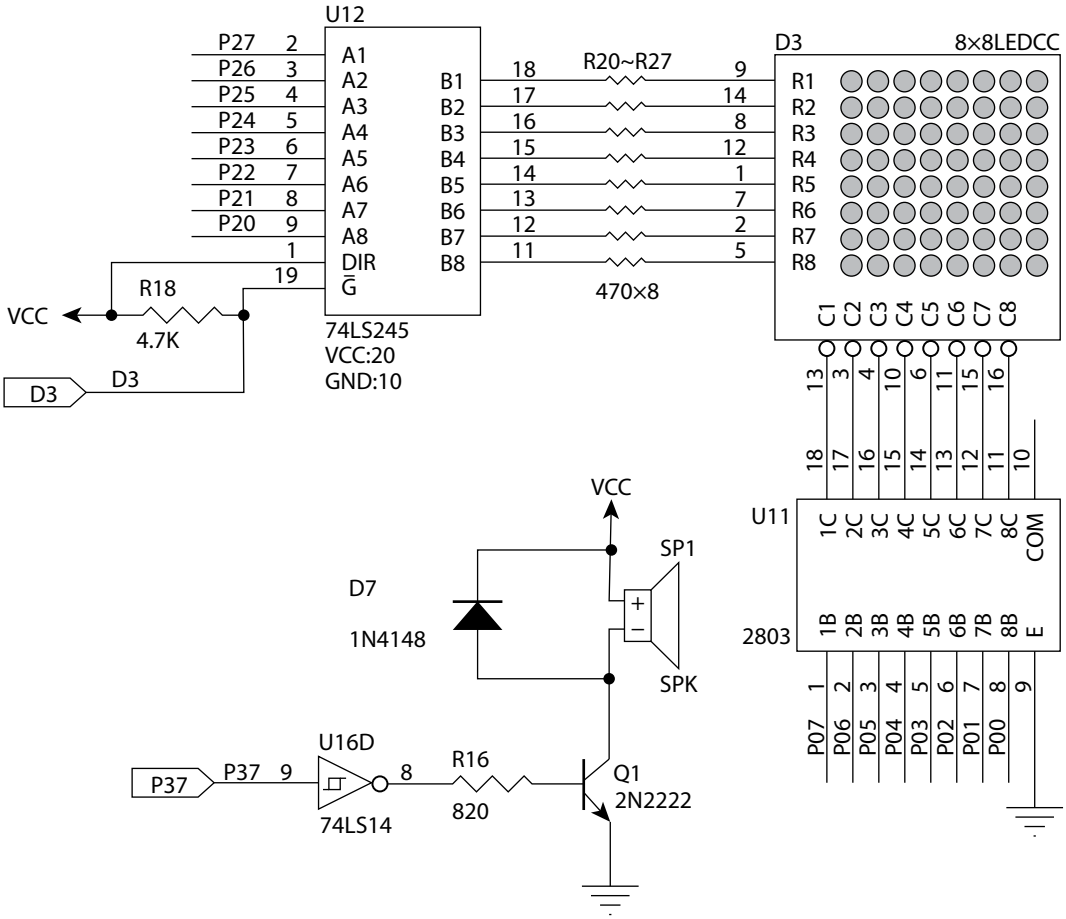
2.6. att. Soļu dzinējs un tā vadības shēma.

Vēl viens plaši lietots ārējās ierīces veids — fotopārtraucējs — ietilpst *MTS-51* ierīcē (skat. 2.7. attēlu). Ievietojot starp gaismas diodi un uztvērēju gaismas necaurlaidīgu lapu, veidojas pārtraukuma signāls. Apmācības ierīcē ir divi fotopārtraucēji PH1 un PH2 (pirmais izdos pārtraukuma signālu INT1 vai paceļ karodziņu T1, otrais — paceļ karodziņu T0). Bez tam pārtraukuma signālu (T0 pacelšanu) var veidot arī no impulsu ģeneratora U15, mainot pārslēdzēja JP4 stāvokli. Ģenerators U15 veidots, izmantojot taimera mikroshēmu (*555*). Šmita trigeri U16A un U16B veido taisnstūra impulsus no ģeneratora sinusa veida signāla.

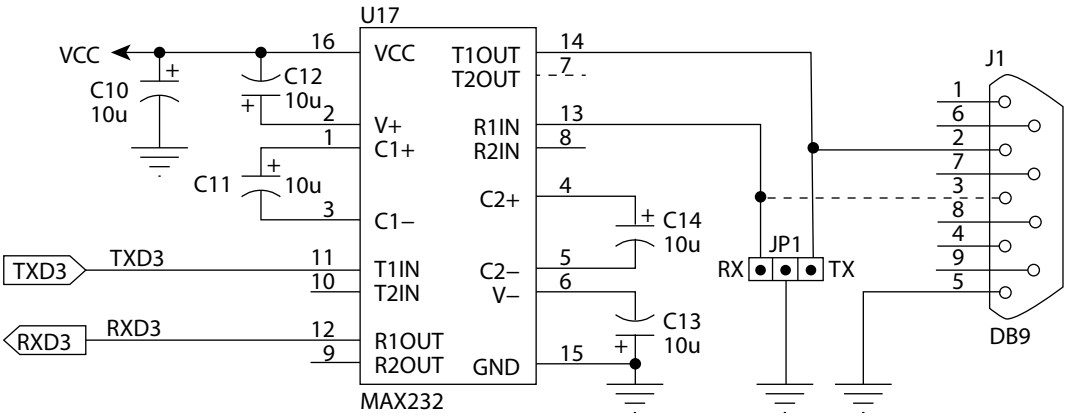


2.7. att. Fotopārtraucēju un ģeneratora shēma.

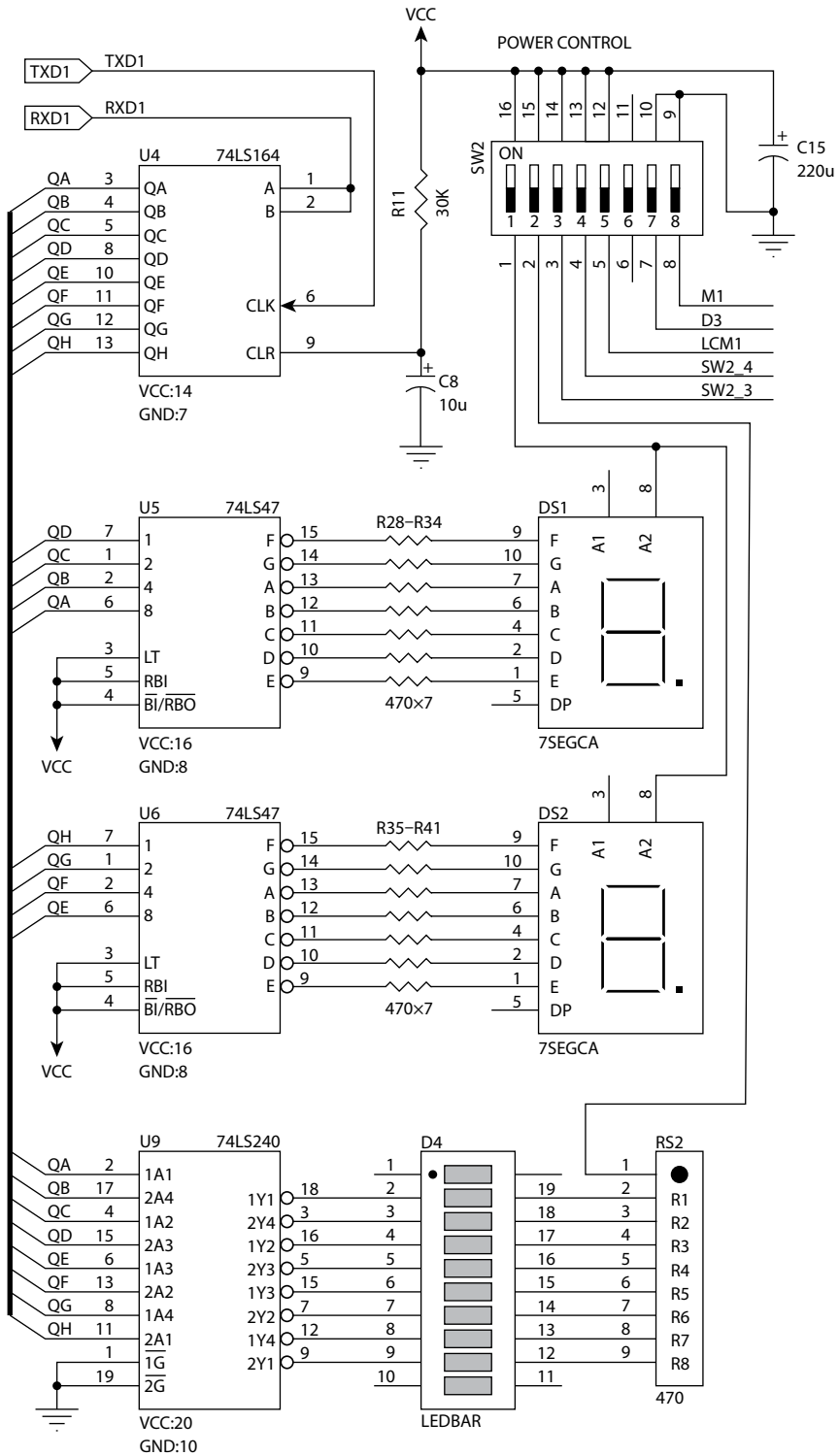
Kā perifērijas ierīces apmācības ierīcē *MTS-51* ir iespējams izmantot gaismas diožu matricu un skaļruni (skat. 2.8. attēlu). Diožu matrica (D3) pieslēgta pie portu P0 un P2 izvadiem, izmantojot astoņu kanālu buferus U11 un U12. Skaļruņa vadība notiek, mainot trešā porta septītās līnijas signālu.



2.8. att. Diožu matricas un skaļruņa shēma.



2.9. att. *RS-232* signālu formētāja shēma.

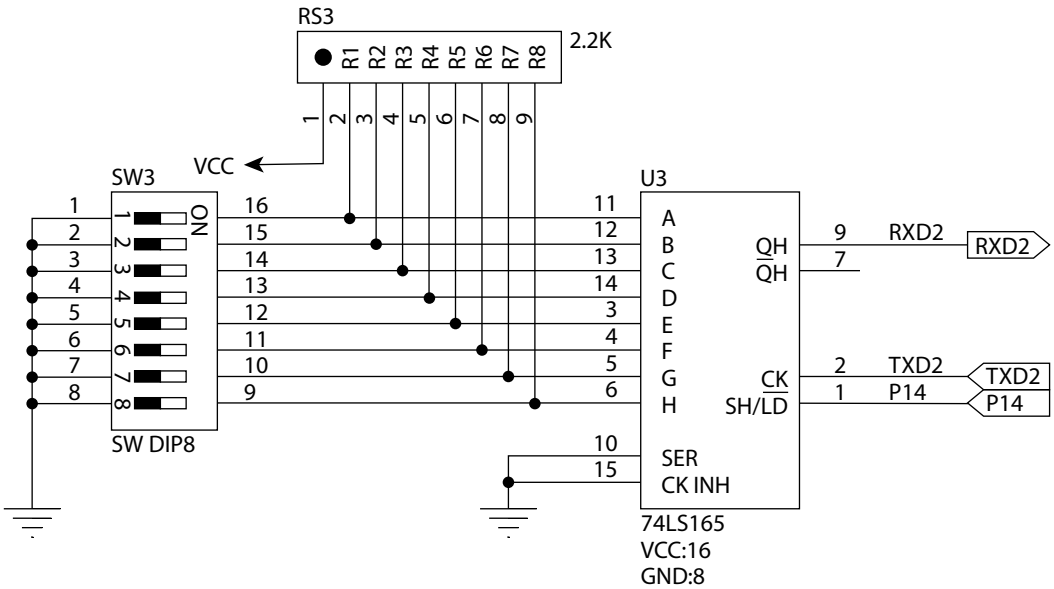


2.10. att. Ar *RS-232* interfeisu saistītās ierīces.

Mikrokontrolleris *MTC-51* nodrošina datu pārraidi un pieņemšanu virknes formā, tomēr, lai pieslēgtu *RS-232* savienotāju, nepieciešams signālus pastiprināt un slāpēt (uztvertos). Šim nolūkam apmācības ierīcē izmantots raidītājs — uztvērējs U17 — *MAX232* (skat. 2.9. attēlu).

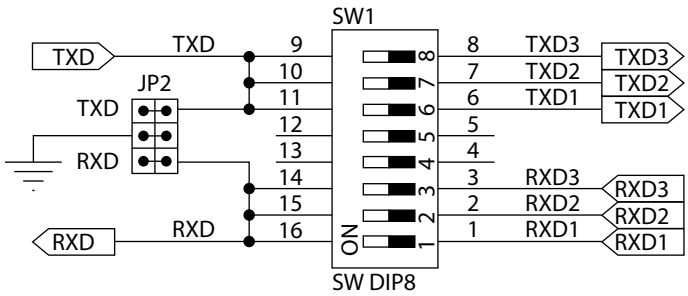
Apmācības ierīcē *MTS-51* izveidota perifērijas ierīču grupa, kura informācijas apmaiņai izmanto virknes formu, izmantojot *MCS-51* virknes portu. Šādā veidā notiek saikne ar diviem septiņu segmentu indikatoriem DS1 un DS2 un astoņām diodēm D4 (skat. 2.10. attēlu). Virknes informācijas pārveidošanā paralēlā formā tiek izmantots pārveidotājs *74LS164* (elements U4). Shēmā vēl parādīta slēdžu grupa, ar kuru padod barošanas spriegumu dažādām perifērijas ierīcēm apmācības sistēmā (SW2).

Arī informācijas ievade mikrokontrollerī var tikt veikta virknes veidā. Šādam nolūkam tiek izmantota slēdžu grupa SW3 un paralēlās informācijas pārveidotājs virknes formā U3 (skat. 2.11. attēlu).



2.11. att. Slēdžu informācijas ievade virknes formātā.

Tā kā apmācības sistēmā pie virknes porta izvadiem var pieslēgt dažādas ierīces, tad nepieciešama to komutācija. 2.12. attēlā parādīta raidītāja TXD un uztvērēja RXD izvadu komutācijas shēma.



RXD1/TXD1:74LS164

RXD2/TXD2:74LS165

RXD3/TXD3:RS232

2.12. att. Virknes portu komutācija.

4. Programmas asamblēšana un sasaiste

Programmas tekstu var rakstīt jebkurā teksta redaktorā, piemēram, *Notepad*. Uzrakstīto programmu ir jāsaglabā ar ASM paplašinājumu, piemēram, EXAMPLE.ASM. Tālāk kā piemērs dots programmas kods, kā iededzināt diodes D5 paketē pēc kārtas (nobīda vienīnieku).

```
        ORG     000H            ; programmas sākumadrese
        MOV     A,#10000000B    ; pirmās diodes kods akumulatorā
NEXT:   MOV     P2,A            ; izgaismot diodi (numurs akumulatorā)
        CALL    DELAY           ; izsaukt aiztures apakšprogrammu DELAY
        RR      A               ; cikliski rotēt akumulatora saturu pa labi
        AJMP    NEXT            ; pāriet uz cikla sākumu NEXT
                                ; aiztures apakšprogramma
DELAY:  MOV     R6,#200         ; R6 = 200 (decimālajā sistēmā)
DL1:    MOV     R7,#249         ; R7 = 249 (decimālajā sistēmā)
        DJNZ    R7,$            ; R7 = R7 – 1, ja R7 > 0, pāriet uz šo pašu rindiņu
        DJNZ    R6,DL1          ; R6 = R6 – 1, ja R6 > 0, pāriet uz DL1
        RET                     ; iziet no apakšprogrammas
        END                     ; programmas beigas
```

Iepriekš minētās programmas teksts uzrakstīts šādā formātā (atdalīšana notiek ar tabulācijas simbolu, nospiežot **TAB** taustiņu):

```
Iezīme:    Operācija      Operandi       ; komentārs
```

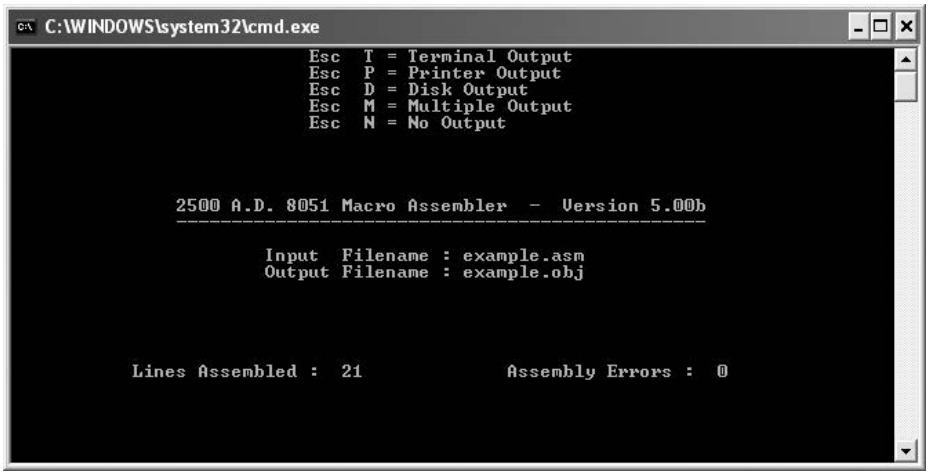
Tādējādi asamblera instrukcija sastāv no četrām daļām, kas atdalītas viena no otras ar tabulācijas simbolu. Šo daļu funkcijas ir šādas:

- **iezīme** — tiek izmantota kā simboliskais atmiņas šūnas apzīmējums. Nepieciešama, lai veiktu zarošanas instrukcijas. Iezīmes garums ir ierobežots ar 32 simboliem, un pati iezīme nedrīkst sakrist ar kādu no komandām. Pēc iezīmes tiek likts kola simbols (:);
- **operācija** — šīs lauks satur izpildāmās komandas operācijas koda simbolisko apzīmējumu vai arī pseidokomandu, kas tiek izmantotas asamblēšanas procesā;
- **operandi** — šajā laukā tiek norādīti reģistri/adreses (netiešās adresācijas jāizmanto "@" simbolu priekšā) vai arī konstantes (jāizmanto "#" simbolu priekšā).
- **komentārs** — sniedz informāciju par izpildāmo darbību programmētājam vai arī ikvienam, kas lasa programmu. Tas padara programmas uztveri vieglāku un palīdz atklāt loģiskās kļūdas. Pirms komentāra nepieciešams likt semikola zīmi (;). Komentāri nav obligāti programmas darbībai un netiek ievēroti asamblēšanas procesā.

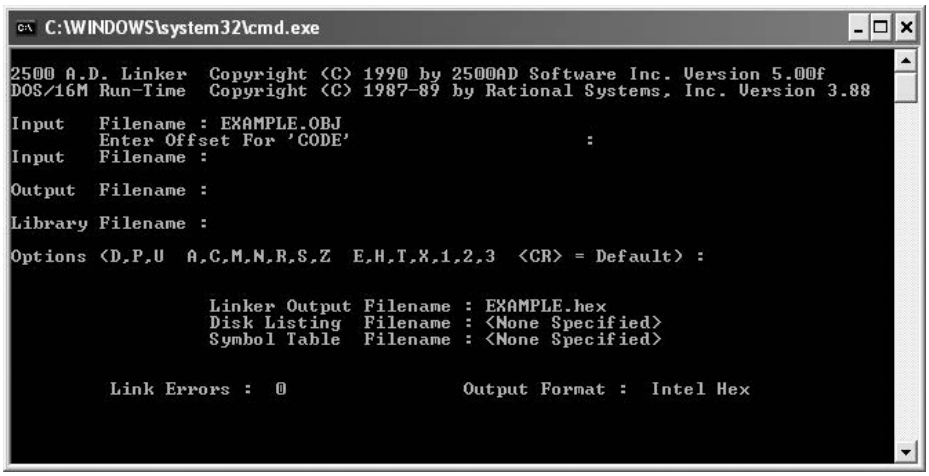
Kad programma ir uzrakstīta un saglabāta EXAMPLE.ASM faila veidā, tā jāpārvērš mašīnkodos. Šis process sastāv no diviem posmiem — no sākuma nepieciešams veikt asamblēšanu, lai iegūtu objekta failu (kopā ar uzskaites (listinga) failu), pēc asamblēšanas veic sasaisti (linkēšanu), kuras procesā tiek iegūts mašīnas kods.

Lai pārveidotu *.asm failu *.hex failā, ir jāizpilda šādas operācijas:

1. Ievietot uzrakstītās programmas koda failu kopā ar asamblera/linkera failiem vienā direktorijā. Asamblera/linkera failus var nokopēt no kompaktdiska, kas nāk līdzi *MTS-51* sistēmai.
2. Palaist DOS komandu rindu, pāriet direktorijā ar failiem, izmantojot **cd** komandu (**cd** pāriet uz diska saknes mapi).
3. Ievadīt komandu asamblēšanai. Vienkāršākajā gadījumā jāizpilda šāda komanda: "X8051.EXE EXAMPLE.ASM". Ja nepieciešams, lai objekta failam būtu cits nosaukums, jānorāda arī tas: "X8051.EXE EXAMPLE.ASM DIODES.OBJ". Ja nepieciešams veidot sasaistes (listinga) failu, jāpievieno parametrs *D*: "X8051.EXE–D EXAMPLE.ASM". Parādīsies logs ar rezultātiem:



4. Ja ir radušās kļūdas, tās jāatrod failā ar paplašinājumu .lst un jāizlabo. Pēc labošanas asamblēšana ir jāatkārto.
5. Pēc asamblēšanas jāveic sasaistes faila veidošana vai linkēšana. Pēc noklusēšanas tiek veidots *Intel* HEX fails, kuru arī ir nepieciešams iegūt. Lai veiktu linkēšanu, jāpalaiž Link51.exe fails un jāatbild uz programmas jautājumiem, kā redzams piemērā:



Input Filename — jāievada objekta faila nosaukums, piemēram, EXAMPLE.OBJ.

Enter Offset For 'CODE' — adreses nobīde mašīnas kodam; var atstāt tukšu (nospiežot taustiņu **Enter**).

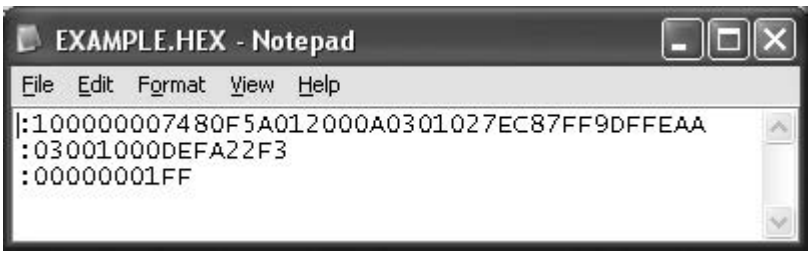
Input Filename — otrā sasaistāmā objekta faila nosaukums. Ja tāda nav, nospiest taustiņu **Enter**.

Output Filename — izejas faila (HEX) nosaukums, ja tas tāds pats kā objekta failam, nospiest taustiņu **Enter**.

Library Filename — sasaistāmās bibliotēkas nosaukums, jā tādas nav, nospiest taustiņu **Enter**.

Options — HEX faila formāta izvēle. Pēc noklusēšanas tiek veidots *Intel* HEX fails; var neko nerakstīt un nospiest taustiņu **Enter**. Ja nepieciešams kāds cits formāts, tas ir jānorāda un jānospiež taustiņš **Enter**.

Rezultātā tiks izveidots *Intel* HEX formāta fails, kuru var atvērt ar jebkuru teksta redaktoru, piemēram, ar *Notepad*.



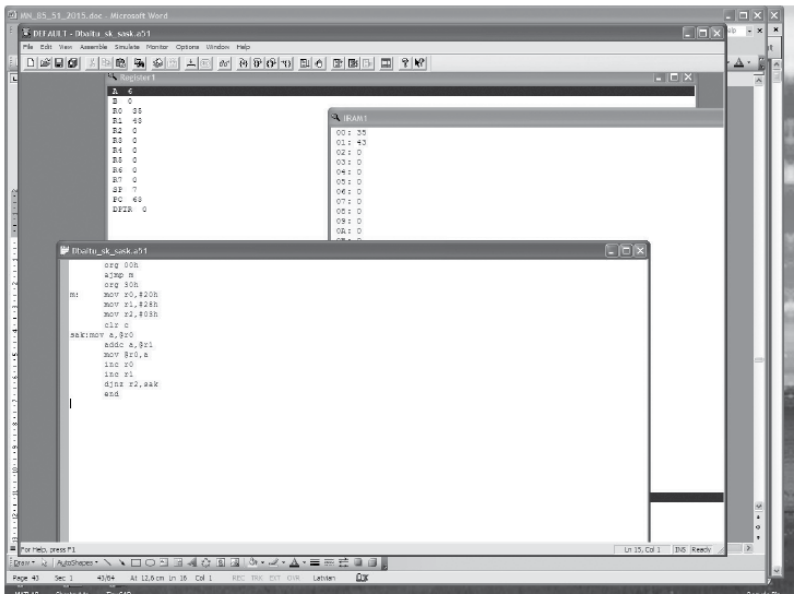
Katrā rindiņā ir sešas sekcijas, kā tas redzams piemērā (vienai rūtiņai atbilst viens heksadecimālais cipars):

:										…			
(1)	(2)		(3)				(4)		(5)			(6)	

(1) Kola simbols ":" — *Intel* HEX faila atpazīšanas identifikators.
(2) Datu baitu skaits rindā, divi heksadecimālie cipari, maksimālā vērtība 10H.
(3) Sākumadrese (četri cipari). Pēc šīs adreses atrodams rindas pirmais datu baits.
(4) Datu tips: "00" apzīmē, ka rindā ir kods, un "01" apzīmē, ka rindā nav koda.
(5) Mašīnkodi — vienā rindā 32 ciparu kods (16 baiti).
(6) Kontrolsumma — visu datu summa pēc moduļa 2.

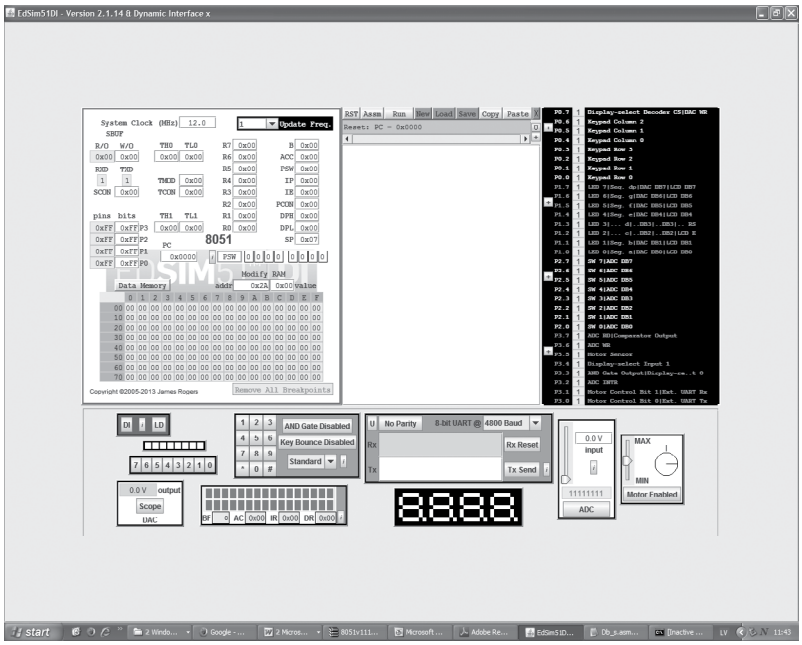
Lai novērstu loģiskās kļūdas programmas rakstīšanas procesā, pirms tās izmantošanas veic simulāciju. Viens no plaši izmantotiem simulatoriem ir *AVSIM51*, kas ietilpst standarta *MK-51* programmatūrā un strādā DOS operacionālā sistēmā. Lai izmantotu šo simulatoru, jāpalaiž fails AVSIM51.exe, pēc tam jāieraksta iepriekš iegūtais fails ar paplašinājumu .hex un tālāk jāseko norādēm. Simulācija ir ļoti uzskatāma, jo ekrānā redzami visi reģistri (porti), datu un programmas atmiņa, kā arī vadības reģistri.

Lai rakstītu programmas *MK-51 Windows* vidē, kā arī lai iegūtu failus ar paplašinājumiem .list un .hex, var izmantot programmu *Windows IDE* (*integrated development environment*) kontrollerim *I8051* (*Win8051*), kuras logi parādīti attēlā. Ar šīs programmas palīdzību var veikt arī uzrakstītās programmas pārbaudi.



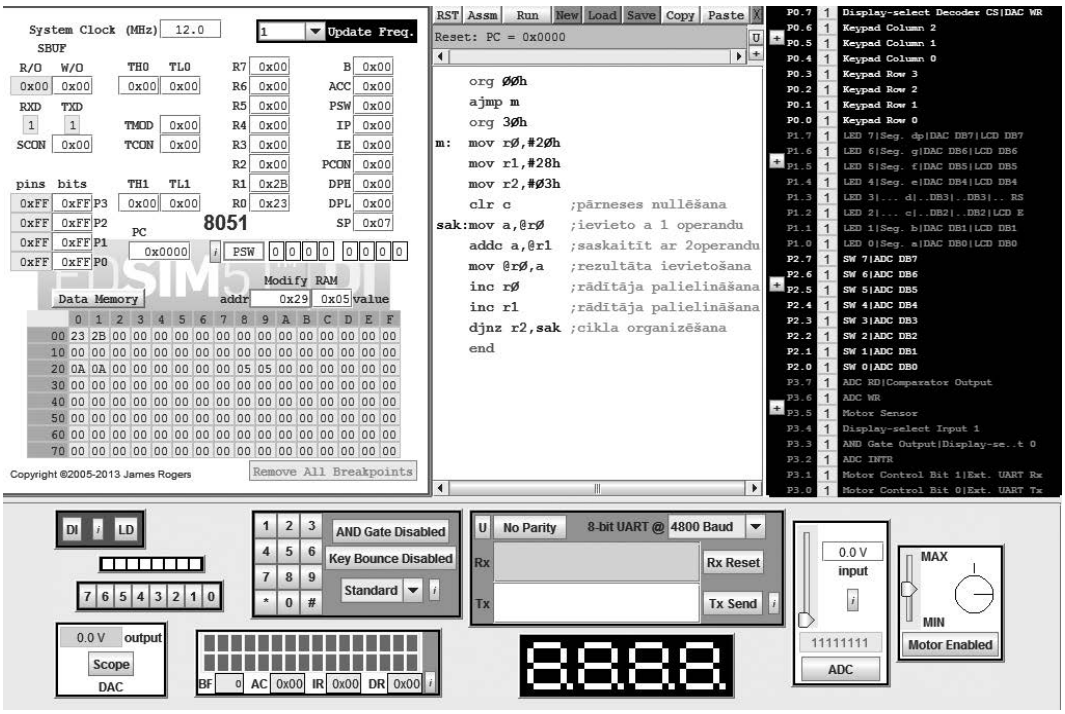
Programmu var iegūt interneta vietnē http://www.acebus.com/win8051.htm. Vienīgā atšķirībano DOS programnodrošinājuma ir nepieciešamība saglabāt asamblera failu ar paplašinājumu .a51.

Ir izstrādāti arī citi simulācijas programmu varianti *Windows* vidē. Mēs izmantosim bezmaksas sistēmu *EdSim51*, kas parocīga arī asamblera faila uzrakstīšanai. Palaižot programmu edsim51di.jar, parādīsies simulēšanas un programmas asamblera faila logi, kas parādīti attēlā.



Programmas kodu raksta attēlā redzamā baltā laukumā, pēc tam to var saglabāt, asamblerēt un pārbaudīt. Saglabāto asamblera failu pēc tam var kompilēt, iegūstot failus ar paplašinājumu .obj un .list. Lai nebūtu kļūdu, failu nosaukumus labāk veidot ar astoņām zīmēm, piemēram, db_s.asm.

Apskatīsim piemēru, kā sastādīt un pārbaudīt programmu daudzbaitu skaitļu saskaitīšanai. Skaitļi novietoti datu atmiņā, sākot ar jaunāko baitu no atmiņas šūnām 20h un 28h attiecīgi, bet skaitļu garums baitos ievietots reģistrā R3. Pēc programmas uzrakstīšanas iegūstam imitācijas programmu.



5. Mikrokontrollera programmēšana

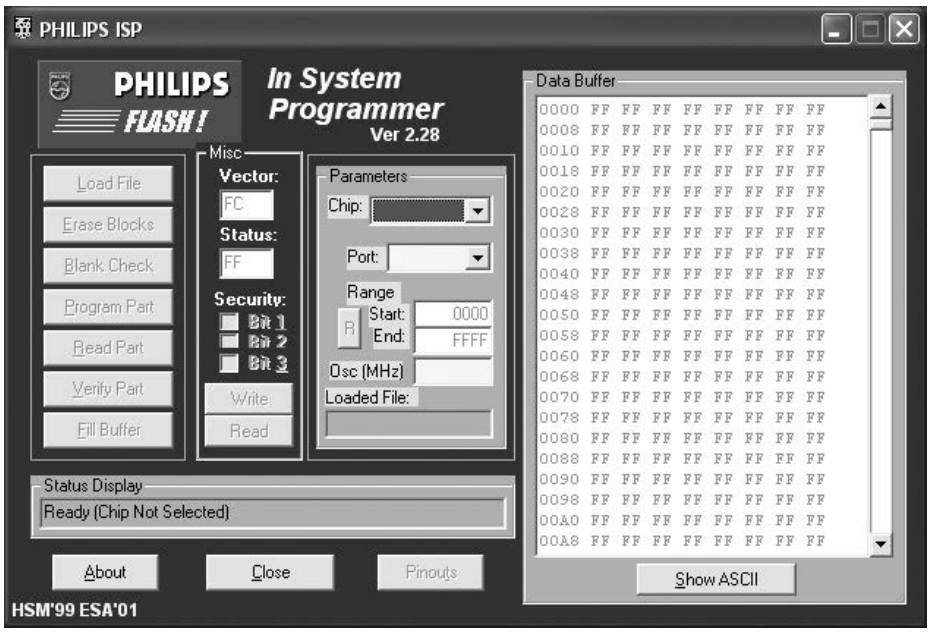
P89C51RD+/P89C51RD2 mikroshēmu sērijai ir programmējamā zibatmiņa, kas nozīmē, ka šos mikrokontrollerus var programmēt vairākkārtīgi (tūkstošiem reižu). Šiem mikrokontrolleriem ir tā saucamais *boot loader*. Tā ir ROM atmiņa (kas atdalīta no mikrokontrollera zibatmiņas), kurā atrodas instrukcijas operācijām ar šūnām, kas ļauj programmēt mikrokontrolleri. *Boot* ROM atmiņa (1 KB) pārklājas ar programmējamo zibatmiņu. Taču *boot* ROM var arī atslēgt (sīkāk tas paskaidrots *P89C51RD+* mikrokontroller datu lapā internetā), tādā veidā iegūstot papildu zibatmiņas kilobaitu, kuru var programmēt. Taču tādā gadījumā būs nepieciešams atsevišķs programmators. *Boot loader* aizņem adrešu telpu no FC00H līdz FCFFH.

Kompānijas, kas ražo mikrokontrollerus, raksta programmatūru, ar kuras palīdzību var programmēt šīs kompānijas produktus. Katrai kompānijai ir savas tehnoloģijas, tāpēc firmas *Atmel* mikrokontrollerim nepieciešama firmas *Atmel* programmatūra un firmas *Philips* mikrokontrolleriem nepieciešama firmas *Philips* programmatūra. *MTS-51* iekārtā tiek izmantots firmas *Philips* mikrokontrolleris *P89C51RD+*, un programmēšanai vajadzīgo programmatūru *WinISP* var lejupielādēt internetā. Jaunākā versija atrodama: http://www.semiconductors.philips.com/product/standard/microcontrollers/support/80c51/insysprog/

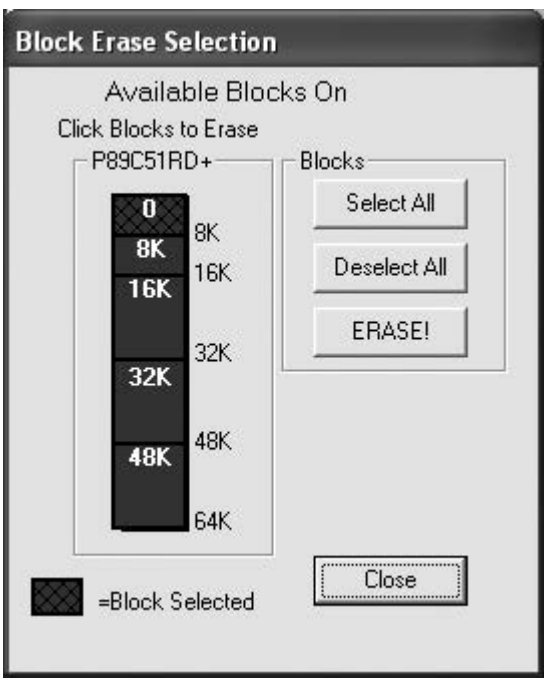
Pirms sākt programmēšanu, jāizvēlas pareizs spriegums *MTS-51* ierīcē atkarībā no mikrokontrollera mikroshēmas. Izmantojot tiltslēgu, jāsavieno JP5 izvadi 2 (EA) un 3 (+12 V), ja tiek izmantota *P89C51RD+* mikroshēma. Ja tiek izmantota *P89C51RD2* mikroshēma, jāsavieno JP5 izvadi 2 (EA) un 1 (+5 V).

Programmēšanas procesu var iedalīt šādos posmos:

1. *Intel* HEX formāta faila veidošana, izmantojot asambleri un savienotāju (linkeri).
2. Uz *MTS-51* iekārtas jānospiež poga SW5 (ISP), kas mainīs pirmo izpildāmo adresi no 0000H uz FC00H (*boot loader*). Lai dators varētu komunicēt ar *MTS-51* ar *RS-232* interfeisa palīdzību, jāieslēdz SW1-3 un SW1-8 slēdži.
3. Izmantojot DB9 kabeli, jāsavieno personālā datora COM ports ar *MTS-51* J1 savienotāju.
4. Ieslēgt barošanu.
5. Palaist *WinISP* programmu, izsaucot tā ikoniņu **START** izvēlnē. Parādīsies programmas galvenais logs.



6. Jāveic noskaņošana konkrētajai mikroshēmai. **Parameters** sadaļā jāaizpilda lauki, kuri ir iekrāsoti dzelteni:
   - ***Chip*** — programmējamā mikroshēma, jāizvēlas tā, kas ir iesprausta iekārtas ligzdā. Šajā gadījumā tā ir *P89C51RD+*;
   - ***Port*** — datora COM ports, pie kura tiek pieslēgta *MTS-51* iekārta. Parasti tas ir COM1 ports;
   - ***OSC*** (MHz) — iekārtas kvarca frekvence MHz vienībās. *MTS-51* iekārtas kvarca ģeneratora frekvence ir 12 MHz;
   - ***Vector Field*** un ***Status Byte*** — jaunai mikroshēmai *status* vērtība ir FF, tā ir jānomaina uz 00. Mikroshēmai, kas tika izmantota iepriekš, *status* vērtība vienmēr ir 00. *Vector* vērtībai ir jābūt FC, jo *Philips P89C51RD+/P89C51RD2* mikrokontrolleru *boot* ROM sākumadrese ir FC00H. *Security* trīs biti jāatstāj tukši.
7. Lai atvērtu HEX failu, kas jāielādē mikrokontrollera atmiņā, jānospiež poga **Load File** un jāizvēlas HEX fails.
8. Ja mikroshēmas atmiņā jau ir kāds kods, pirms jauna koda rakstīšanas ir jāveic atmiņas tīrīšana. To var izdarīt, nospiežot pogu **Erase Blocks**. Parādīsies **Block Erase Selection** logs.



Šajā logā jāizvēlas tie apgabali, kuros ir dati. Procesa paātrināšanai atmiņa ir sadalīta segmentos, lai varētu izvēlēties tieši tos blokus, kuros ir programma, un "netīrīt" jau tīrus atmiņas blokus. Pēc bloku izvēles jānospiež poga **ERASE**!.

9. Datu ieprogrammēšanai jānospiež poga **Program Part**. Pēc veiksmīgas programmēšanas lodziņā parādīsies ziņojums: "Flash Programming Successful".

10. Lai novērotu programmas darbību, jāatspiež poga SW5, jāizvēlas nepieciešamie SW2 slēdži, pieslēdzot spriegumu pie visām vajadzīgām ierīcēm. Pēc pogas **Reset** nospiešanas sāks izpildīties mikrokontrollera zibatmiņā ierakstītā programma.

Ja mikrokontrollera programmēšana nenotika veiksmīgi, jāpārbauda:

1. Vai ir izvēlēta pareiza mikroshēma (*P89C51RD+*)?
2. Vai ISP sprieguma tiltslēdzis JP5 ir pareizi uzstādīts (*P89C51RD+* — 12 V, *P89C51RD2* — 5 V)?
3. Vai ir izvēlēts pareizais COM ports?
4. Vai ir ieslēgti SW1 slēdži ar numuriem 3 un 8 (*RS-232 TXD3* un *RS-232 RXD3*)?
5. Vai *RS-232* raidītājuztvērēja mikroshēma *MAX232* strādā normāli? Izmantojot multimetru, var izmērīt spriegumus uz mikroshēmas *MAX232* (blakus J1 savienotājām) otrās un sestās kājiņas. Šīm vērtībām ir jābūt attiecīgi +10 V un –10 V.

6. Studiju projekta uzdevumi un norādījumi izpildei

Studiju projekts paredzēts teorētisko un praktisko zināšanu nostiprināšanai par transporta mikroprocesoru sistēmām, kā arī elektroniskās aparatūras projektēšanas un shēmu noformēšanas iemaņu apgūšanai. Projekts bakalaura līmenī tiek veidots, izmantojot *MTS-51* sistēmas mezglus. Tāda pieeja ļauj reāli pārbaudīt izstrādātās ierīces un programmatūras pareizu funkcionēšanu. Studiju projektā jāizstrādā uzdevuma risināšanai nepieciešamās ierīces struktūras, funkcionālā un elektriskā principiālā shēma. Visas shēmas jānoformē pēc Latvijas Republikā spēkā esošiem standartiem un rakstlaukumā jānorāda studenta vārds, uzvārds, grupas Nr. un cita informācija (Jurāne, Veide, 2006). Jāaprēķina visi projektētās ierīces elementu nomināli, precizitāte, jauda un citi parametri, kas raksturo to. Pielikumā jādod elementu uzskaitījums pēc spēkā esošiem standartiem.

Projektā jāizstrādā uzdevuma risināšanas algoritms un jāsastāda asamblera programma. Visas sastādītās programmas rindas jāpapildina ar komentāriem. Pēc programmas sastādīšanas un kompilācijas, tā jāieraksta mikrokontrollerī un jāpārbauda, izmantojot *MTS-51* ierīci. Noformējot projektu, tajā iekļaujami pārbaudes rezultāti fotogrāfiju, oscilogramu vai citu mērījumu veidā.

Studiju projekta uzdevumi

1. Diožu vadība. Ieslēgt vienu pēc otras astoņas diodes, sākot no kreisās. Katras diodes degšanas laiks — 0,2 s.
2. Diožu vadība. Ieslēgt vienu pēc otras astoņas diodes, sākot no kreisās. Sasniedzot labo diodi, ieslēgšana notiek apgriezti. Katras diodes degšanas laiks — 0,2 s.
3. Diožu vadība. Ieslēgt divas malējās diodes, pēc tam divas nākamās no malas utt. Sasniedzot vidu, invertēt diožu iedegšanās secību. Katras kombinācijas degšanas laiks — 0,5 s.
4. Diožu vadība. Ieslēgt diodes paketē D5 pēc šādas sakarības: nedeg neviena diode, deg visas, nedeg neviena, deg kreisās četras diodes, nedeg neviena, deg labās četras diodes, pēc tam process atkārtojas. Katrs process notiek 0,2 s. Diožu vadībai izmantot *MCS-51* otro portu.
5. Septiņu segmentu indikatoru vadība (DS3 un DS4), izmantojot otro portu. Pēc kārtas parādīt uz indikatoriem skaitļus no 00 līdz 99 un pēc tam no 99 līdz 00. Pēc tam process atkārtojas. Katra skaitļa indikācijas laiks — 1 s.
6. Septiņu segmentu indikatoru vadība (DS3 un DS4), izmantojot otro portu. Pēc kārtas parādīt uz indikatoriem skaitļus no 00 līdz 99. Pēc tam process atkārtojas. Katra skaitļa indikācijas laiks — 1 s.
7. Septiņu segmentu indikatoru vadība (DS3 un DS4), izmantojot otro portu. Pēc kārtas parādīt uz indikatoriem skaitļus no 99 līdz 00. Pēc tam process atkārtojas. Katra skaitļa indikācijas laiks — 1 s.
8. Matricas diožu vadība. Iedegt kolonnas diodes no pirmās līdz astotajai, pēc tam rindu diodes no pirmās līdz astotajai. Katras kolonnas un rindas diožu degšanas laiks — 0,2 s.
9. Matricas diožu vadība, izmantojot otrā un nulles porta signālus. Parādīt burtu F uz diožu matricas.
10. Matricas diožu vadība, izmantojot otrā un nulles porta signālus. Parādīt burtu H uz diožu matricas.
11. Matricas diožu vadība, izmantojot otrā un nulles porta signālus. Parādīt burtu A uz diožu matricas.
12. Ar matricas tastatūru vadīt diožu paketi D5: nospiežot taustiņu **0**, jādeg visām astoņām diodēm, nospiežot taustiņu **1**, visas diodes mirgo, nospiežot taustiņu **3**, pa vienai diodei iedegas no labās puses, nospiežot taustiņu **4**, pa vienai diodei iedegas no kreisās puses.
13. Ar matricas tastatūru vadīt diožu paketi D5: nospiežot taustiņu **0**, jādeg labajai diodei, nospiežot taustiņu **1** — nākamai no labās puses diodei, nospiežot taustiņu **2** — otrai no labās puses diodei.
14. Ar matricas tastatūru vadīt diožu paketi D5: nospiežot taustiņu **0**, jādeg divām diodēm labajā pusē, nospiežot taustiņu **1** — nākamām divām diodēm no labās puses, nospiežot taustiņu **2** — nākamām divām diodēm no labās puses.

15. Nospiežot taustiņu **7**, uz matricas tastatūras parādīt skaitli 7 uz septiņu segmentu indikatora.
16. Nospiežot taustiņu **4** uz matricas tastatūras parādīt skaitli 4 uz septiņu segmentu indikatora.
17. Soļu dzinēja vadība. Lieciet soļu dzinējam nepārtraukti griezties vienā virzienā.
18. Soļu dzinēja vadība. Vadiet dzinēja griešanos uz vienu un otru pusi, izmantojot taustiņus **1** un **0**.
19. Soļu dzinēja vadība. Izmantojot taustiņus **0**, **1**, **2,**…, **F**, mainiet soļu dzinēja griešanās ātrumu.
20. Septiņu segmentu indikatoru (DS1 un DS2) vadība, izmantojot virknes portu. Parādīt skaitļus no 00 līdz 99 pēc kārtas.
21. Slēdžu (SW3) informācijas ievade, izmantojot virknes portu. Saskaitīt ievadītā skaitļa (SW3) vecākās tetrādes bitus ar jaunākas tetrādes bitiem. Rezultātu parādīt uz septiņu segmentu indikatoriem DS3 un DS4.
22. Fotopārtraukums. Izmantot fotopārtraucēju PH2 soļu dzinēja vadībai. Ja gaismas stars nav bloķēts, motors griežas vienā virzienā, bet, ja stars tiek bloķēts, — pretējā.
23. Skaitītāja lietošana. Saskaitīt impulsus fotopārtraucēja PH1 izejā un rezultātu parādīt uz septiņu segmentu indikatoriem DS3 un DS4.
24. Taimera/skaitītāja lietošana signāla formēšanā. Izmantojot taimeri/skaitītāju nulles režīmā, formēt taisnstūra signālu ar frekvenci 1 kHz porta P2.0 izejā.
25. Taimera/skaitītāja lietošana signāla formēšanā. Izmantojot taimeri/skaitītāju nulles režīmā, formēt taisnstūra signālu ar frekvenci 2 kHz porta P2.0 izejā.
26. Skaļruņa vadība. Nodrošināt skaļruņa darbu ar frekvenci 2 kHz 1 sekundi, tad klusēšanu 1 sekundi utt.
27. Skaļruņa vadība. Nodrošināt skaļruņa darbu ar frekvenci 1 kHz 0,5 sekundes, tad klusēšanu 0,5 sekundes utt.

Izmantotā literatūra un avoti

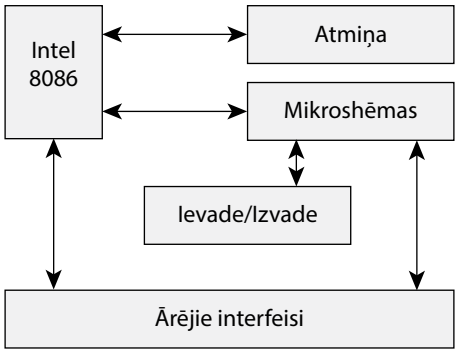
1. Klūga, A. *Mikroprocesori un mikroprocesoru sistēmas*. Mācību līdzeklis. Rīga: RTU Izdevniecība, 2007. 152 lpp.
2. *MTS-51 Microcomputer Trainer*. Taiwan, K&H MFG CO., LTD, 2008.
3. Jurāne, I., Veide, Z. *Rasējumu un dokumentu noformēšana*. Lekciju konspekts. Rīga: RTU Izdevniecība, 2006.
4. Win8051 programmatūra. Interneta vietne: http://www.acebus.com/win8051.htm, [skatīta 2016. gadā].

III. Mikroprocesora *Intel 8086* lietošana

1. Vispārīgā informācija par *MTS-86C*

Pēdējo 30 gadu laikā mikroprocesori ir attīstījušies no vienkāršām četru kārtu ierīcēm ar 16 instrukcijām līdz 64 kārtu ierīcēm ar vairākiem simtiem instrukciju un adresācijas veidu. Liela daļa mūsdienīgo elektronisko sistēmu iekļauj sevī firmas *Intel 8086* vai *8088* arhitektūras mikroprocesoru.

MTS-86C ir mikroprocesoru apmācības sistēma, kuras pamatā ir *Intel 8086* (*I8086*) mikroprocesors. Tā sastāv no piecām sastāvdaļām: mikroprocesora, atmiņas, mikroshēmu komplekta, ievadizvades ierīcēm un ārējiem interfeisiem (skat. 3.1. attēlu).



3.1. att. *MTS-86C* mikroprocesoru sistēmas struktūra.

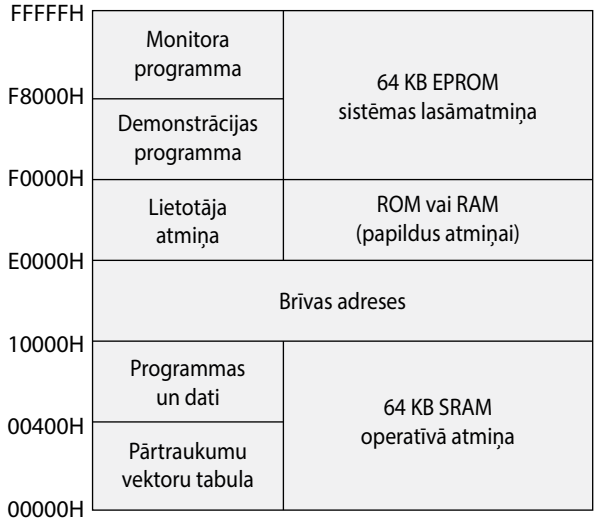
MTS-86C mikroprocesoru apmācības sistēmā ietverts:

- mikroprocesors — firmas *Intel* 16 bitu mikroprocesors *I8086*, to izmanto kā centrālo procesoru, līdzīgi, kā tas notiek personālajā datorā.
- atmiņa — sastāv no lasāmatmiņas, operatīvās atmiņas un lietotāja atmiņas:
  - lasāmatmiņa (ROM) — divas *27256 UV EPROM* mikroshēmas (32 KB atmiņa katrai), kas tiek izmantotas sistēmas monitora programmai un demonstrācijas programmu uzglabāšanai;

  - operatīvā atmiņa (RAM) — divas *62256 SRAM* mikroshēmas (32 KB atmiņa katrai), kas tiek izmantotas lietotāja programmu/datu un pārtraukumu vektora tabulas uzglabāšanai. Dati operatīvajā atmiņā tiek uzglabāti arī tad, kad *MTS-86C* iekārta ir izslēgta, izmantojot 3,6 V bateriju;
  - lietotāja atmiņa — *MTS-86C* ir arī iespēja paplašināt atmiņu ar vēl divām mikroshēmām (RAM/ROM pēc nepieciešamības);
- mikroshēmu komplekta uzskatāmība — visas mikroshēmas ir novietotas uz priekšējā paneļa ar uzrakstiem:
  - *8255* — programmējamais paralēlais ievadizvades interfeiss (trīs mikroshēmas),
  - *8251* — virknes datu pārraides interfeiss (divas mikroshēmas),
  - *8259* — programmējamais pārtraukumu kontrolleris,
  - *8253* — programmējamais skaitītājs/taimers,
  - *8279* — programmējamais tastatūras/displeja kontrolleris,
  - *DAC0808* — astoņu bitu ciparanalogu pārveidotājs,
  - *ADC0809* — astoņu bitu astoņu kanālu analogciparu pārveidotājs.
- ievadizvades ierīces:
  - astoņi TACT tipa slēdži (vienā korpusā),
  - astoņas gaismas diodes,
  - astoņu segmentu indikators (bez dekodētāja),
  - tastatūra — 4 × 6 taustiņi (seši taustiņi vienā rindā, pavisam četras rindas),
  - šķidro kristālu displejs (LCD), divas rindas pa 16 simboliem katrā,
  - skaļrunis ar jaudu 2 W (skaļumu var regulēt ar VR1 potenciometru),
  - ACP ieejas kanāli:
    - mikrofons,
    - mainīga pretestība (potenciometrs),
    - termopretestība,
    - fototranzistors.
- Ārējie interfeisi:
  - PPI-1 — pirmās *8255* mikroshēmas paralēlā porta izvadi,
  - PPI-2 — otrās *8255* mikroshēmas paralēlā porta izvadi,
  - PIT/PIC/AD/DA — ārējais interfeiss taimerim, ACP/CAP un programmējamajam pārtraukumu kontrollerim,
  - *System Bus — 8086* procesora sistēmas kopne (sastāv no adrešu, vadības un datu kopnēm),
  - *RS232-1* — pirmās *8251* mikroshēmas virknes porta interfeiss,
  - *RS232-2* — pirmās *8251* mikroshēmas virknes porta interfeiss.

2. *MTS-86C* mikroprocesoru sistēmas atmiņa

MTS-86C sistēmas adrešu telpas sadalījums ir parādīts 3.2. attēlā. *I8086* mikroprocesors operē ar 20 bitu adresēm, kas dod iespēju adresēt 2^{20} baitus jeb 1048 576 unikālās adreses (viens megabaits, no 00000H līdz FFFFFH).

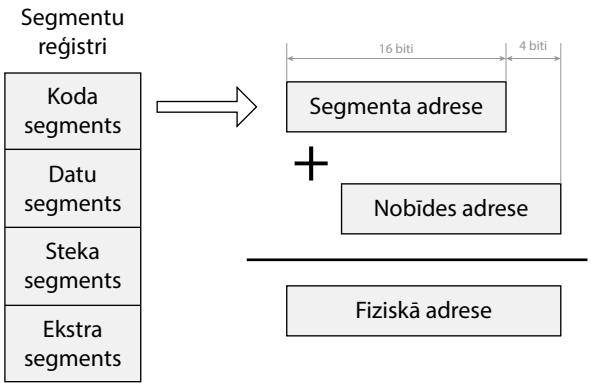


3.2. att. *MTS-86C* adrešu telpas sadalījums.

Kā var redzēt 3.2. attēlā, programmējamās operatīvās atmiņas 64 KB aizņem adrešu telpu no 00000H līdz 0FFFFH, sistēmas lasāmatmiņas 64 KB aizņem adrešu telpu no F0000H līdz FFFFFH. Savukārt adrešu telpa no E0000 līdz EFFFF tiek rezervēta lietotāja papildatmiņai (ja tā ir vajadzīga). Papildatmiņai var izmantot gan RAM, gan arī ROM mikroshēmas.

Fiziskās adreses formēšana ir parādīta 3.3. attēlā. *8086* sistēmas loģiskā adrese tiek uzdota ar segmentu un nobīdi. Gan segments, gan nobīde ir 16 kārtu skaitļi, jo visi reģistri operē ar 16 bitu skaitļiem. Tomēr *8086* mikroprocesoram fiziskai adresei ir atvēlētas 20 kārtas, tātad fiziskās adreses veidojas kā kombinācija no 16 bitu nobīdes un 16 bitu bāzes vērtības, kas atrodas vienā no četriem segmentu reģistriem:

- CS (*code segment*) — koda segments, kurā glabājas izpildāmās instrukcijas;
- DS (*data segment*) — datu segments, kurā glabājas programmas dati (piemēram, tabulas);
- SS (*stack segment*) — steka segments, kurā pārtraukumu apakšprogrammas saglabā atgriešanās adresi;
- ES (*extra segment*) — papildu segments, kas tiek izmantots jauktajiem datiem.



3.3. att. *MTS-86C* fiziskās adreses ģenerēšana.

Kā var redzēt 3.3. attēlā, galvenā ideja ir tāda, ka segmenta adresei tiek pierakstīti četri nulles biti jaunākajās kārtās. Iegūtajai 20 bitu adresei pieskaita klāt 16 bitu nobīdes adresi, tādā veidā formējot fizisko adresi no 20 bitiem.

Visus četrus segmentus nav nepieciešams atsevišķi definēt. Ir pieļaujams, ka visi segmenti pilnīgi pārklājas (CS = DS = SS = ES).

3. *MTS-86C* atmiņā esošās programmas palaišana

Visām demonstrācijas programmām, kuru saraksts ir dots 3.1. tabulā, tiek norādītas loģiskās adreses. Paņemsim, piemēram, melodiju programmu (5. programma). Tās segmenta adrese ir F040H, bet nobīdes adrese 0000H. Rezultātā iegūsim fizisko adresi kā summu: F0400H + 00000H = F0400H.

Ar *MTS-86C* tastatūras palīdzību var ievadīt segmenta un nobīdes adrešu vērtības, lai piekļūtu pie atmiņas ar fizisko adresi. Lai to panāktu, jāizpilda šādas procedūras:

1. Ieslēgt *MTS-86C*. Uz LCD ekrāna parādīsies šāds uzraksts:

```
MTS - 86C
K&H MFG
```

2. Nospiest pogu **GO** uz *MTS-86C* tastatūras. Uz LCD ekrāna parādīsies šāds uzraksts:

```
Seg. Offs. Data
0000:0000_ 22 GO
```

3.1. tabula

Demonstrācijas programmas

Nr.	Adrese	Saturs	DIPSW3 slēdži	Piezīmes
01	F000 : 0	8255 (LED mirgošana)		Izvade: LED
02	F010 : 0	8255 (no slēdžiem uz LED)		Ievade: TACT slēdži Izvade: LED
03	F020 : 0	74LS373 (astoņu segmentu indikators)		Izvade: FND
04	F030 : 0	8255/74LS373 (no slēdžiem uz astoņu segmentu indikatoru FND)		Ievade: TACT slēdži Izvade: FND
05	F040 : 0	8253 (melodija)	2	Izvade: skaļrunis
06	F060 : 0	8255/8253/74LS373 (piano)	2	Ievade: TACT slēdži Izvade: skaļrunis
07	F080 : 0	8255 (melodija)	1	Ievade: TACT slēdži 0, 1 Izvade: skaļrunis
08	F0E0 : 0	8253/8259 (skaitītājs)	4	Izvade: LCD
09	F0F0 : 0	8259 (pārtraukums)		Ievade: poga IR0 Izvade: LED
10	F100 : 0	Nemaskējamais pārtraukums (NMI)		Ievade: NMI taustiņš Izvade: FND
11	F110 : 0	8279 (tastatūra)		Ievade: tastatūra Izvade: LCD
12	F120 : 0	DAC0808 (zāģveida impulsi)	3	Izvade: TP1 D/A OUTPUT (oscilogrāfs)
13	F130 : 0	DAC0808 (trijstūra impulsi)	3	Izvade: TP1 D/A OUTPUT (oscilogrāfs)
14	F140 : 0	DAC0808 (taisnstūra impulsi)	3	Izvade: TP1 D/A OUTPUT (oscilogrāfs)
15	F150 : 0	DAC0808 (sinusoidāls signāls)	3	Izvade: TP1 D/A OUTPUT (oscilogrāfs)
16	F160 : 0	ADC0809 (mainīga pretestība)	8	Ievade: VR2 Izvade: LED un LCD
17	F170 : 0	ADC0809 (fototranzistors)	8	Ievade: PHOTO Izvade: LED un LCD
18	F180 : 0	ADC0809 (termopretestība)	8	Ievade: THER Izvade: LED un LCD
19	F1A0 : 0	ADC0809, DAC0808 (runas ierakstīšana)	3, 8	Ievade: mikrofons Izvade: skaļrunis
20	F1C0 : 0	LCD		Izvade: LCD

3. Ievadīt heksadecimālo skaitli — segmenta adresei — F040.
4. Nospiest taustiņu kols (**:**) uz tastatūras, lai pārietu uz nobīdes adresi.
5. Ievadīt heksadecimālo skaitli — nobīdes adresei — 0000.
6. Nospiest taustiņu punkts (**.**) uz tastatūras, lai izpildītu programmu.
7. Uzstādīt DIPSW3 otro slēdzi stāvoklī **ON**, lai ieslēgtu skaļruni.
8. Lai izpildītu citu programmu, jānospiež poga **Reset** un jāsāk no 3. punkta.

Eksistē vēl viens paņēmiens, kā var palaist šīs demonstrācijas programmas. Lai palaistu demonstrācijas programmu, ir jāizpilda šādas operācijas:

1. Ieslēgt *MTS-86C*. Uz ekrāna parādīsies šāds uzraksts:

```
MTS - 86C
K&H MFG
```

   Tas nozīmē, ka mikroprocesoram izpildās apakšprogramma — sistēmas monitors, kas glabājas lasāmatmiņas mikroshēmā *27256*.
2. Nospiest taustiņu punkts (**.**) uz *MTS-86C* tastatūras. Parādīsies šāds uzraksts:

```
<DEMO PROGRAM>
SELECT 01-20
```

3. Ievadīt divu ciparu programmas numuru no 01 līdz 20 uz *MTS-86C* tastatūras, lai ielādētu izvēlēto eksperimentu. Rezultātu varēs novērot uz ievadizvades ierīcēm, kuras tiek izmantotas programmā.
4. Uzstādīt DIPSW3 slēdžus **ON** stāvoklī, ja tas ir nepieciešams perifērijas ierīcēm. DIPSW3 slēdžu stāvokļi ir parādīti 3.1. tabulas 4. kolonnā. Piemēram, kad 05. programma tiek izpildīta, jāuzstāda otrais bits DIPSW3 slēdzim stāvoklī **ON**. Pretējā gadījumā skaļrunis būs atvienots no strāvas.
5. Lai izpildītu citu programmu, jānospiež poga **Reset** un jāatkārto vēlreiz visu, sākot no 1. punkta.

4. *MTS-86C* ievadizvades portu adreses

Dati par visām ievadizvades ierīcēm, kas fiziski pievienotas pie *I8086* mikroprocesora sistēmas, ir apkopoti 3.2. tabulā. Lielāko ievadizvades ierīču daļu var vadīt ar **IN** un **OUT** instrukcijām. Jāatzīmē, ka paralēlie ievadizvades interfeisi Nr. 1 un Nr. 2 tiek izmantoti prototipa veidošanai, savukārt interfeisu Nr. 3 var izmantot eksperimentos.

3.2. tabula

***MTS-86C* ievadizvades porti**

Porta Adrese	Porta funkcijas		Piezīmes
FFFFH	8255-1	Vadības vārda reģistrs	<Prototipa mērķiem>
FFFDH	8255-1	C ports	pirmais paralēlais
FFFBH	8255-1	B ports	ievades/izvades
FFF9H	8255-1	A ports	interfeiss, PPI-1
FFFEH	8255-2	Vadības vārda reģistrs	<Prototipa mērķiem>
FFFCH	8255-2	C ports	otrais paralēlais
FFFAH	8255-2	B ports	ievades/izvades
FFF8H	8255-2	A ports	interfeiss, PPI-2
FFF2H	8251-1	Vadības komandas	RS232C pirmais ports
FFF0H	8251-1	Dati	
FFEAH	8279	Stāvokļa vārda komandas	Tastatūras vadība
FFE8H	8279	Dati	
FFDEH	8253	Vadība	Skaitītājs un taimers
FFDCH	8253	Otrais skaitītājs	
FFDAH	8253	Pirmais skaitītājs	
FFD8H	8253	Nulltais skaitītājs	
FFD2H	8251-2	Vadības komandas	RS232C otrais ports
FFD0H	8251-2	Dati	
FFCAH	8259	Vadība2 (ICW2_3_4; OCW1)	Pārtraukumu vadība
FFC8H	8259	Vadība1 (ICW1; OCW2_3)	
FFC5H	LCD	Datu ierakstīšana	LCD
FFC3H	LCD	Instrukcijas (stāvokļa) nolasīšana	
FFC1H	LCD	Instrukcijas (komandas) ierakstīšana	
3FF0H	FND		Astoņu segmentu indikators
3FD8H	D/A		Astoņu bitu ciparanalogu pārveidotājs
3FD6H	8255-3	Vadības vārda reģistrs	<PPI-3>
3FD4H	8255-3	C ports	portC — uz skaļruni
3FD2H	8255-3	B ports	portB — uz LED
3FD0H	8255-3	A ports	portA — uz TACT slēdžiem
3FCEH	A/D	IN3 vai IN7	Astoņu bitu
3FCCH	A/D	IN2 vai IN6	analogciparu pārveidotājs.
3FCAH	A/D	IN1 vai IN5	IN0–IN3, IN4–IN7
3FC8H	A/D	IN0 vai IN4	izvēlas ar DIPSW3

5. Programmas asamblēšana

Programmas tekstu var rakstīt jebkurā teksta redaktorā, piemēram, *Notepad*. Uzrakstītā programma ir jāsaglabā ar ASM paplašinājumu, piemēram, LED.ASM. Piemērā ir dots kods programmai, kura ieslēdz visas gaismas diodes uz *MTS-86C*. Uzskatīsim, ka tā ir uzrakstīta un saglabāta kā LED.ASM fails.

```
CNT3    EQU     3FD6H           ; 8255 vadības vārda reģistra adrese
BPORT3  EQU     3FD2H           ; 8255 porta B adrese

CODE    SEGMENT
        ASSUME  CS:CODE, DS:CODE

        ORG     0

START:  MOV     SP,4000H        ; uzstādīt steka rādītāju
        MOV     AL,90H          ; vadības vārda reģistra informācija
        MOV     DX,CNT3         ; izvades porta izvēle
        OUT     DX,AL           ; izvade

        MOV     AL,0FFH         ; izvades dati (vieninieki visām diodēm)
        MOV     DX,BPORT3       ; izvades porta izvēle (diodes)
        OUT     DX,AL           ; izvade
        HLT
CODE    ENDS
        END     START
```

Uz *MTS-86C* kompaktdiska ir iekļauti līdzekļi programmu asamblēšanai. Tie ir uzskaitīti 3.3. tabulā.

3.3. tabula

***MTS-86C* asamblēšanas faili**

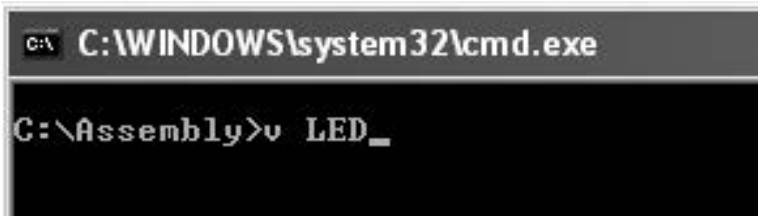
Faila nosaukums	Apraksts
V.bat	Skripta (*batch*) fails, pārveido *.asm failu *.hex failā
Bin2Hex.exe	Pārveido bināro failu heksadecimālajā formātā
Exe2Bin.exe	Pārveido izpildāmo failu binārajā formātā
Link.exe	*Microsoft Linker*
Masm.exe	*Microsoft* asambleris
NE.com	*Norton* redaktors

Ērtības nolūkos tika izveidots skripta fails, kurš izpilda visas nepieciešamas operācijas pēc kārtas. Rezultātā lietotājam nav nepieciešams patstāvīgi palaist programmas visos etapos: asamblēšanā, sasaistē (linkēšanā) un formāta pārveidošanā.

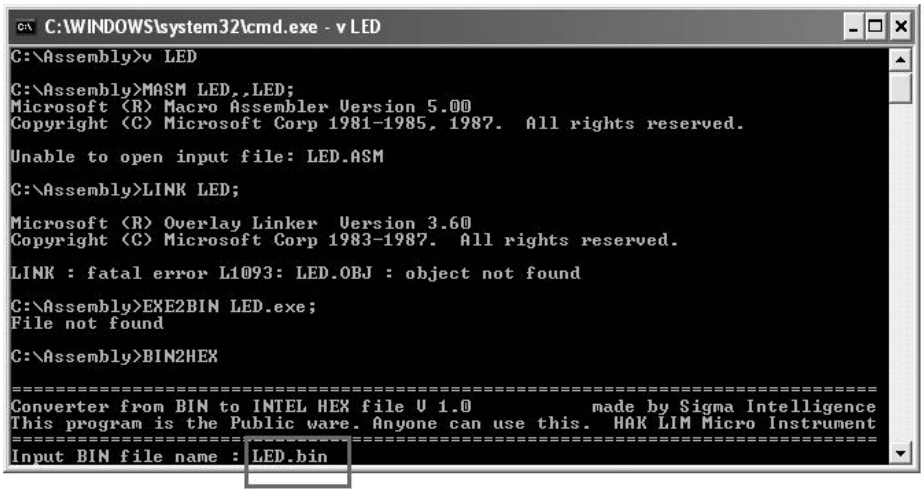
Lai pārveidotu *.asm failu *.hex failā, ir jāizpilda šādas operācijas:

1. Ievietot uzrakstītās programmas koda failu kopā ar failiem no 3. tabulas vienā direktorijā, piemēram, **ASM86**.
2. Palaist DOS komandu rindu, pāriet direktorijā ar failiem, izmantojot **cd** komandu (**cd**, pāriet uz diska saknes mapi).

3. Ievadīt komandu **V LED**, kā tas parādīts piemērā. **Svarīgi** — nevajag ievadīt faila paplašinājumu (nepareizi: "V LED.ASM"), jo tas var izjaukt uzrakstīto kodu.



4. Pēc **Enter** taustiņa nospiešanas skripta fails sāks izpildīties un drīzumā tas prasīs ievadīt binārā faila nosaukumu, kā tas ir parādīts piemērā.



5. Pēc binārā faila nosaukuma ievades jānospiež poga **Enter** divas reizes (sākumadrese pēc noklusēšanas ir 0000). Skripts beidz savu darbību un rezultātā veidojas LED.HEX fails direktorijā, kurā tika palaists pakešfails.

6. Programmas ielādēšana un palaišana

MTS-86C var ieprogrammēt divos veidos. Pirmais veids — ierakstīt programmas mašīnkodus uzreiz *MTS-86C*, izmantojot LCD un tastatūru. Lai tas būtu iespējams, vispirms jāiegūst šie kodi no *.lst faila, kurš tiek veidots asamblēšanas procesā un satur sevī šos kodus (skat LED.LST faila saturu 3.4. tabulā).

3.4. tabula

LED.lst fails ar programmas mašīnkodiem

Nobīdes adrese	Mašīnas kods	Iezīme	Asamblera kods
0000	BC 4000	START:	MOV SP,4000H
0003	B0 90		MOV AL,90H
0005	BA 3FD6		MOV DX,CNT3
0008	EE		OUT DX,AL
0009	B0 FF		MOV AL,0FFH
000B	BA 3FD2		MOV DX,BPORT3
000E	EE		OUT DX,AL
000F	F4		HLT
0010			CODE ENDS

Pirms programmas ierakstīšanas tā ir jāpārkārto. Ja koda garums ir divi baiti (viens vārds), pirmais nāk jaunākais baits un pēc tam — vecākais. Tas ir uzskatāmi parādīts piemērā.

BC	00	40	B0	90	BA	D6	3F	EE	B0	FF	BA	D2	3F	EE	F4

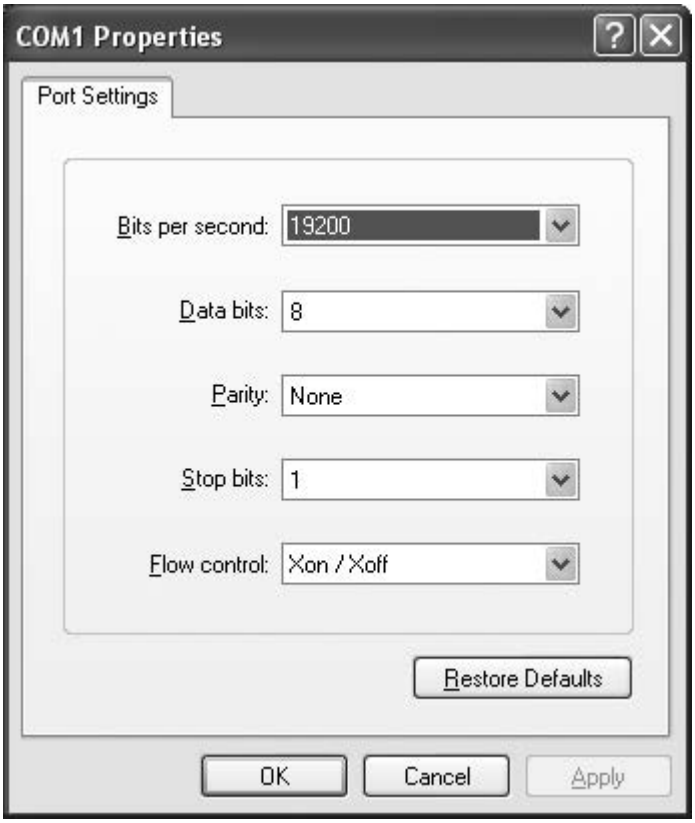
Tagad viss ir gatavs programmas ievadīšanai. Lai ievadītu programmu ar *MTS-86C* tastatūras palīdzību, ir jāizpilda šādas operācijas:

1. Ieslēgt *MTS-86C* vai pārstartēt to, ja tas bija ieslēgts, ar pogu **Reset**.
2. Nospiest taustiņu **EB/AX** (*examine byte*), lai mainītu interesējošo baitu operatīvajā atmiņā.
3. Ievadīt adresi — segmenta adresi, pēc tam seko kola simbols un nobīdes adrese. Adreses ievadi apstiprināt ar taustiņu komats (**,**).
4. Ievadīt vajadzīgo baita vērtību — programma sākas ar pirmo baitu BC. Pēc baita ievades nospiest komatu un ievadīt nākamo baitu.
5. Kad tika ievadīts pēdējais baits, nospiest pogu punkts (**.**), lai izietu no ievades režīma. Nospiest pogu **Reset**, lai varētu palaist programmu.
6. Programmu palaiž tāpat, kā tika palaistas demonstrācijas programmas (pirmā metode). Nepieciešams nospiest pogu **GO**, norādīt pirmā programmas baita adresi (segments:nobīde; tā pati, kas bija trešajā posmā) un nospiest taustiņu punkts (.).

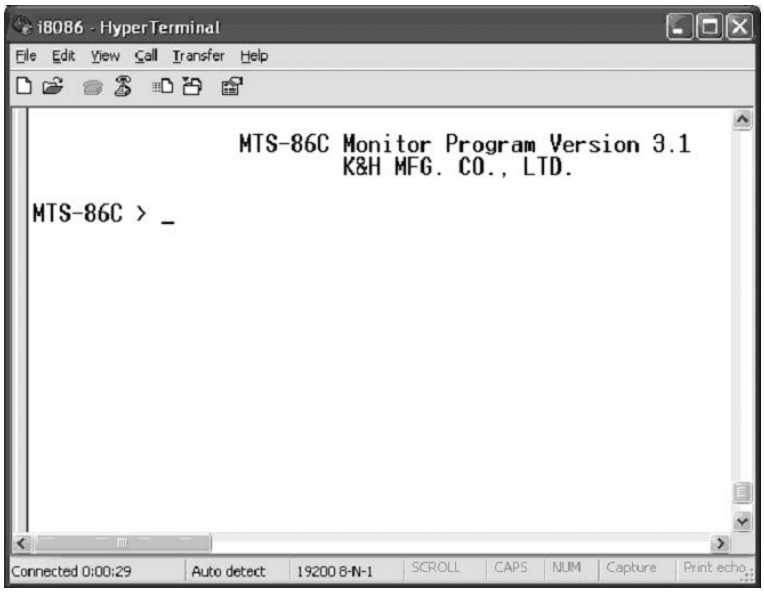
Pēc taustiņu punkts (.) nospiešanas uz *MTS-86C* priekšējā paneļa var novērot visu astoņu gaismas diožu indikāciju.

Programmas ielādēšana un palaišana ar personālā datora palīdzību. *MTS-86C* spēj komunicēt ar parasto personālo datoru, izmantojot *RS232* interfeisu. Lai veiktu šo procesu, jāizpilda šādi soļi:

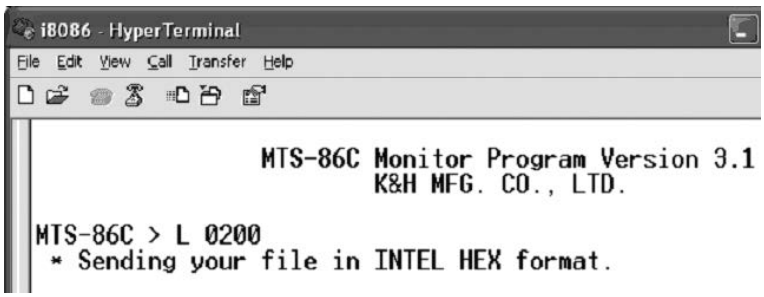
1. Savienot datora COM portu ar *RS232-1* savienotāju uz *MTS-86C*, izmantojot kabeli ar 25 piniem vienā galā (*MTS-86C*) un deviņiem piniem otrā (personālajam datoram). Ieslēgt *MTS-86C*.
2. Palaist *Hyperterminal*, izsaucot caur **START** izvēlni: **START → Programs → Accessories → Communications → Hyperterminal**.
3. Ievadīt nosaukumu seansa failam, kas glabā visus uzstādījumus. Nākamajā lodziņā nepieciešams izvēlēties datora COM portu, pie kura tiek pieslēgts *MTS-86C*, parasti tas ir pirmais — COM1.
4. Nākamajā lodziņā jāveic savienojuma uzstādījumi. Nepieciešamie uzstādījumi ir parādīti piemērā.



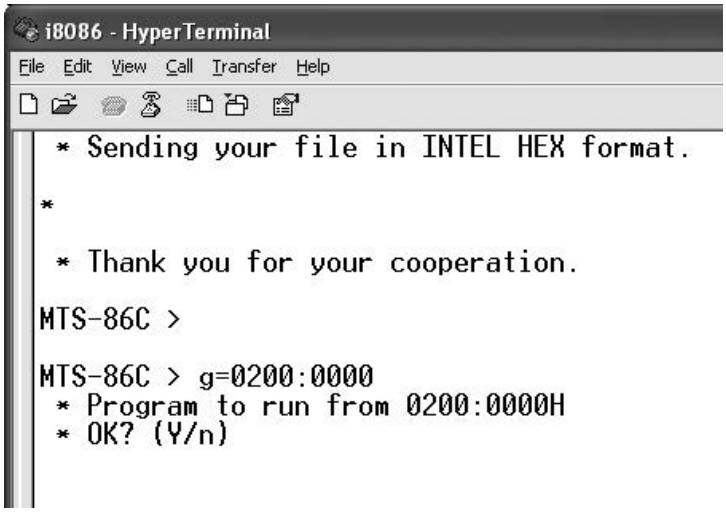
5. Kad parametri ir ievadīti, nospiest pogu **OK**. Uz *MTS-86C* nospiest pogu **Reset**, pēc tam jebkuru burta pogu no **A** līdz **F** uz *MTS-86C* tastatūras. Ja *MTS-86C* ir pieslēgts un viss izdarīts pareizi, parādīsies *MTS-86C* ziņojums, kā tas redzams piemērā.



6. Lai ielādētu programmu *MTS-86C* sistēmā, no sākuma jānorāda adrese, kur atradīsies pirmais baits. Tā kā nobīdes adrese tiek uzdota asamblēšanas procesā un glabājās HEX failā, nepieciešams norādīt tikai segmenta adresi. To var izdarīt ar komandu **L 0200**, kā tas parādīts piemērā.



7. Rezultātā parādīsies ziņojums, ka *MTS-86C* gaida failu *Intel* HEX formātā. Lai to aizsūtītu, no galvenās izvēlnes jāizsauc **Transfer → Send Text File**… un jānorāda led.hex fails.
8. Lai palaistu šo programmu, var izmantot jau agrāk apskatīto komandu **GO**, kuru var ievadīt uzreiz uz *MTS-86C* tastatūras. Var arī palaist, izmantojot *Hyperterminal* programmu. Lai to izdarītu, jāievada pilna adrese (segments:nobīde) šādā veidā: g=0200:0000, un jāapstiprina programmas palaišana, nospiežot taustiņu **y** uz personālā datora tastatūras.



7. Studiju projekta uzdevumi mikroprocesora *Intel 8086* programmēšanai

Studiju projekts paredzēts teorētisko un praktisko zināšanu nostiprināšanai par mikroprocesoru *I8086*, kā arī elektroniskās aparatūras projektēšanas un shēmu noformēšanas iemaņu apgūšanai. Projekts maģistra līmenī tiek veidots, pamatā izmantojot *MTS-86* sistēmas mezglus. Tāda pieeja ļauj reāli pārbaudīt izstrādātās ierīces un programmatūras pareizu funkcionēšanu. Studiju projektā jāizstrādā uzdevuma risināšanai nepieciešamās ierīces struktūras, funkcionālā un elektriskā principiālā shēma. Visas shēmas jānoformē pēc Latvijas Republikā spēkā esošiem standartiem un rakstlaukumā jānorāda studenta vārds, uzvārds, grupas Nr. un cita informācija (Jurāne, Veide, 2006). Jāaprēķina visi projektētās ierīces elementu nomināli, precizitāte, jauda un citi parametri, kas raksturo to. Pielikumā jādod elementu uzskaitījums pēc spēkā esošiem standartiem.

Projektā jāizstrādā uzdevuma risināšanas algoritms un jāsastāda asamblera programma. Visas sastādītās programmas rindas jāpapildina ar komentāriem. Pēc programmas sastādīšanas un kompilācijas, tā jāieraksta atmiņā un jāpārbauda, izmantojot *MTS-86* ierīci. Projekta aprakstā iekļaujami pārbaudes rezultāti fotogrāfiju, oscilogrammu vai citu nepieciešamo mērījumu veidā.

Piedāvātie studiju projekta uzdevumi ir šādi:

1. Datu pārraide, izmantojot *RS232* pirmo portu. Programmas izpildes laikā ievadītie simboli no personālā datora (PD) tiek nosūtīti uz *MTS-86* un atpakaļ uz PD pa virknes datu pārraides kabeli *RS232-1* ar ātrumu 19 200 bodi.
2. Tastatūras kontrole. Programmas izpildes laikā datu ievade ar *MTS-86* tastatūru parādās uz LCD indikatora.

3. Skaņas ierakstīšana un atskaņošana. Programmas izpildes laikā skaņas signāls, kas ievadīts no mikrofona, tiek atskaņots skaļrunī. Darba režīms tiek indicēts uz PD monitora.
4. Termopretestības informācijas parādīšana. Programmas izpildes laikā sprieguma lielums no termopretestības tiek pārveidots binārā kodā un parādīts uz gaismas diodēm, bet heksadecimālā formā uz PD monitora.
5. Gaismas tranzistora informācijas parādīšana. Programmas izpildes laikā sprieguma lielums no gaismas tranzistora tiek pārveidots binārā kodā un parādīts uz gaismas diodēm, bet heksadecimālā formā uz PD monitora.
6. Maiņpretestības informācijas parādīšana. Programmas izpildes laikā sprieguma lielums no maiņpretestības tiek pārveidots binārā kodā un parādīts uz gaismas diodēm, bet heksadecimālā formā uz PD monitora.
7. Sinusoidāla un zāģveida signāla formēšana. Programmas izpildes laikā uz diskrētā analogā pārveidotāja izejas spailes formējas sinusa vai zāģveida formas signāls atkarībā no ievadītās programmas.
8. Laika signāla skaitīšana. Programmas izpildes laikā taimers skaita impulsus un rezultāts parādās uz PD monitora.
9. Pārtraukumu kontrole. Programmas izpildes laikā slēdža SW11 nospiešana izsauc gaismas diožu informācijas nobīdi par vienu bitu.
10. *Piano*. Programmas izpildes laikā dati no astoņiem slēdžiem tiek parādīti uz gaismas diodēm un astoņu segmentu indikatora, kā arī pārveidoti attiecīgā skaņā, kas padodas uz skaļruni.
11. Astoņu segmentu indikators. Programmas izpildes laikā dati no astoņiem slēdžiem tiek padoti uz astoņām gaismas diodēm un astoņu segmentu indikatoru.
12. Gaismas diožu izmantošana informācijas parādīšanai. Programmas izpildes laikā dati no astoņiem slēdžiem tiek padoti uz astoņām gaismas diodēm.
13. Gaismas diožu mirdzēšana. Programmas izpildes laikā nepieciešams veidot noteiktu diožu iedegšanās secību astoņām gaismas diodēm.

Izmantotā literatūra

1. Klūga, A. *Mikroprocesori un mikroprocesoru sistēmas*. Mācību līdzeklis. Rīga: RTU Izdevniecība, 2007, 152 lpp.
2. *MTS-86 Microcomputer Trainer.* Taiwan, K&H MFG CO., LTD, 2008.
3. Jurāne, I., Veide, Z. *Rasējumu un dokumentu noformēšana*. Lekciju konspekts. Rīga: RTU Izdevniecība, 2006.