

Tehnička škola Ruđera Boškovića

# EdSim51

Milan Korać, dipl.ing., profesor savjetnik

Zagreb, 2016

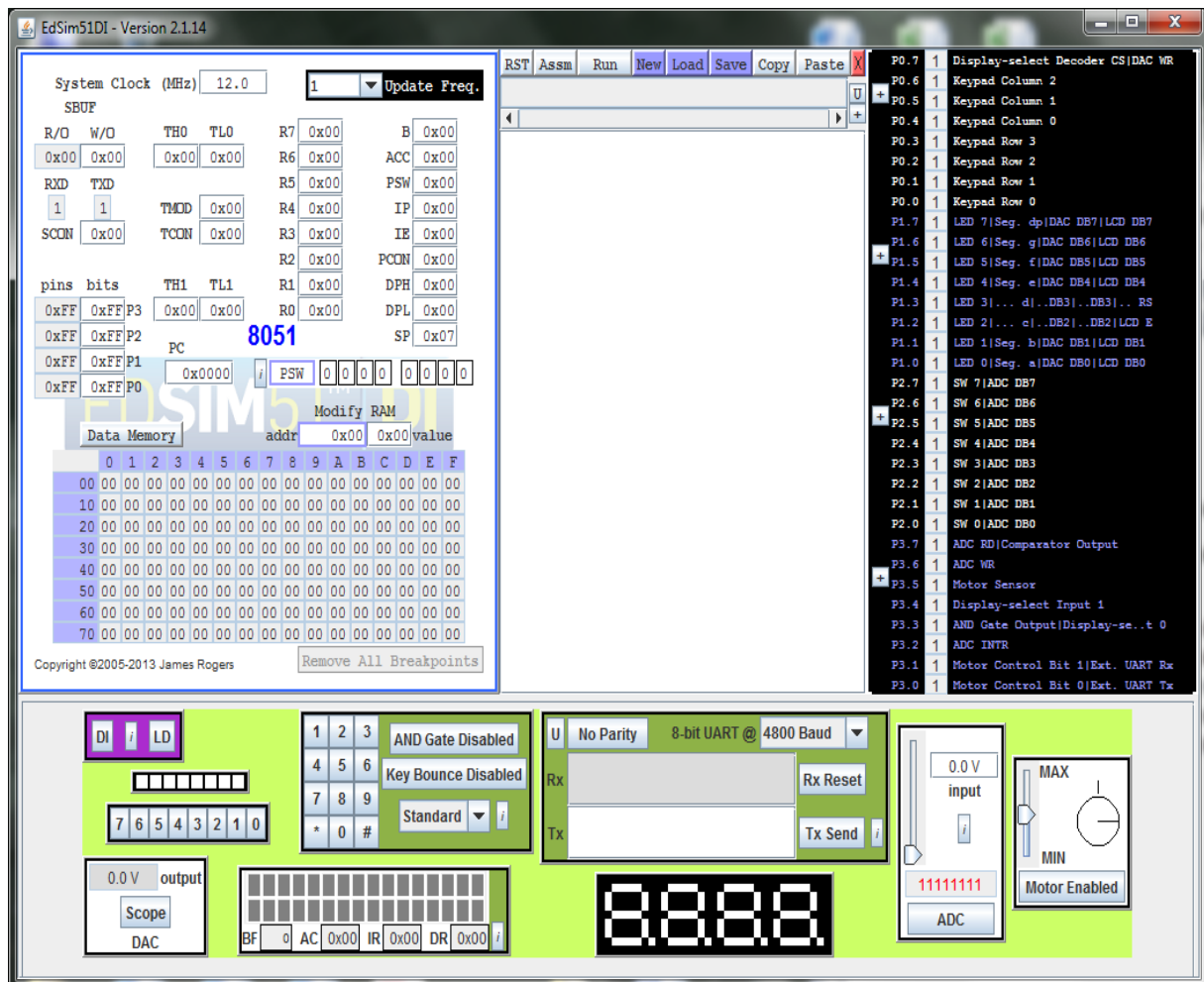
## 1 Sadržaj

2	EdSim51 .....	2
3	Panel mikrokontrolera.....	4
3.1	Registri procesora .....	5
3.2	Kodna memorija .....	6
3.3	Podatkovna memorija.....	7
4	Assembler panel.....	8
4.1	Alatna traka .....	8
4.2	Primjer izvršavanja assembler koda .....	9
4.2.1	Zbrajanje i komplement.....	11
4.2.2	Množenje i logičke operacije .....	14
5	Program status word (PSW).....	17
5.1	Carry flag.....	17
5.2	Parity bit .....	17
5.3	Overflow zastavica .....	17
6	Primjer i objašnjenje izvođenja programa .....	18
6.1	Zapis programa u assembleru.....	18
6.2	Objašnjenje koda.....	20
7	Periferija .....	27
8	Literatura .....	33

## 2 EdSim51

EdSim51 je simulator rada Intelovog procesora 8051. Prednost ovog programa nad ostalim simulatorima je sučelje koje nam daje daleko više informacija od ostalih simulatora.

Program je vrlo jednostavan za korištenje i instalaciju. Može se preuzeti sa EdSim-ove stranice besplatno i nije potrebna nikakva instalacija (uz preduvjet da je na računalu instalirana Java). Program se može preuzeti sa stranice : <http://www.edsim51.com>. Program pokrećemo dvostrukim klikom na edsim51.jar i pojavljuje se sljedeći prozor:



Grafičko sučelje programa EdSim51 je organizirano pomoću prozorima, gdje svaki prozor predstavlja drukčiji tip informacija.

Program ima 4 dijela:

- Lijevo – Mikrokontrolerski panel
- Sredina – Assembler panel
- Desno – Lista Ulazno/Izlaznih pinova
- Dolje – Panel perifernih jedinica

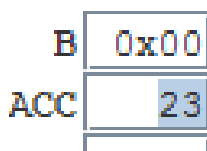
### 3 Panel mikrokontrolera

U panelu mikrokontrolera nalaze se svi registri, akumulator, zastavice, sistemski sat, podatkovna i kodna memorija koja je potrebna za rad procesora.

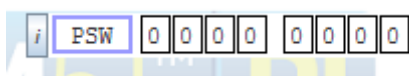
The screenshot shows the EdSim51 microcontroller panel. At the top, the System Clock is set to 12.0 MHz. Below it, the SBUF register is shown with R/O and W/O bits set to 0x00. The TH0 and TL0 registers are also shown with values 0x00. The R7-R0 registers are listed with values 0x00. The B register is 0x00. The ACC register is 0x00. The PSW register is 0x00. The IP register is 0x00. The IE register is 0x00. The PCON register is 0x00. The DPH register is 0x00. The DPL register is 0x00. The SP register is 0x07. The PC register is 0x0000. The PSW register is shown with bits 00000000. The Data Memory table shows addresses 00-70 with values 00-00. The Modify RAM section shows address 0x00 and value 0x00. The Remove All Breakpoints button is visible at the bottom.

addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Korisnik nakon upisivanja i pokretanja koda utječe na stanja na pojedinim registrima, memoriji i zastavicama. Korisnik još može mijenjati podatke svih registra bijele boje dok ne može mijenjati registre sive boje. Sadržaj se mijenja pritiskom na registar i upisivanjem željene vrijednosti.



Ovdje se još nalazi i posebna tražilica koja prikazuje sadržaj bilo kojeg registra u binarnom zapisu odvajajući gornja i donja 4 bita.



U tražilici odaberemo koji registar želimo upisom imena registra u bijeli prozorčić.



### 3.1 Registri procesora

R7	0x00	B	0x00
R6	0x00	ACC	0x00
R5	0x00	PSW	0x00
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x00	DPH	0x00
R0	0x00	DPL	0x00
		SP	0x07

051

Procesor 8051 sastoji se od više registara od kojih je najvažniji akumulator (ACC)

ACC 

0x00
------

. U akumulator se smještaju svi podaci logičkih i aritmetičkih operacija i operandi i rezultat operacije. On je povezan sa aritmetičko-logičkom jedinicom procesora.

Kao pomoćni registar tu je registar B 

0x00
------

. U njega se također upisuju podaci koji će ući u aritmetičko-logičku jedinicu, ali samo operandi. On se još naziva i privremeni registar.

R7	0x00
R6	0x00
R5	0x00
R4	0x00
R3	0x00
R2	0x00
R1	0x00
R0	0x00

Procesor ima 8 unutarnjih registra od R0 do R7 koje koristi za pohranjivanje podataka ili adresa operanda. Ovi registri su najbrži jer su izravno spojeni sa unutarnjom sabirnicom procesora. Ovi registri još se nazivaju i registri opće namjene.

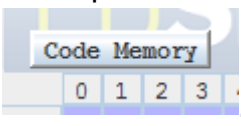
Ovdje se još nalaze bitovi pinova portova P0-P3 i specijalni registri kao npr. SP (stack pointer), TMOD (Timer/Counter Control), DPTR (Data pointer), DPL (Data Pointer low byte), DPH (Data Pointer high byte), PCON (Power Control), PSW (Program status word)...

PSW se sastoji od 8 bitova (zastavica) kojima aritmetičko-logička jedinica utječe na kontrolnu jedinicu postavljajući zastavice. PSW se sastoji od: CY carry zastavice (PSW.7), AC pomoćne carry zastavice (PSW.6), F0 zastavice 0 koja se koristi u općenite svrhe (PSW.5), RS1 register bank select bit 1(PSW.4), RS0 register bank select bit0(PSW.3), OV overflow zastavica (PSW.2), - zastavica koju korisnik određuje (PSW.1) i P zastavica pariteta koja označava da li je broj bitova u log. 1 rezultata paran (PSW.0).

### 3.2 Kodna memorija

Code Memory		Modify Code															
		addr 0x0000 0x00 value															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Kodna memorija zajedno s podatkovnom memorijom čini radnu memoriju. Korisnik može mijenjati između prikaza kodne i podatkovne memorije pritiskom na ime trenutne

memorije: 

U kodnu memoriju se upisuju kodovi instrukcija koje se izvršavaju po redu. Kodna memorija se sama popuni nakon asembliranja koda ili se može promijeniti upisivanjem adrese u memoriji i upisivanjem koda instrukcije u Modify Code koja će se izvršiti kada programsko brojilo dođe do te adrese.

Code Memory		Modify Code															
		addr 0x0000 0xE5 value															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00		E5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Code Memory		Modify Code															
		addr 0x0001 0x23 value															
		1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0
01		23	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
11		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Code Memory	
00	E5 23 00 00
10	00 00 00 00

Ovaj kod će izvršiti instrukciju MOV A,#23h. Kod instrukcije MOV koji prebacuje vrijednost operanda u akumulator je E5.

### 3.3 Podatkovna memorija

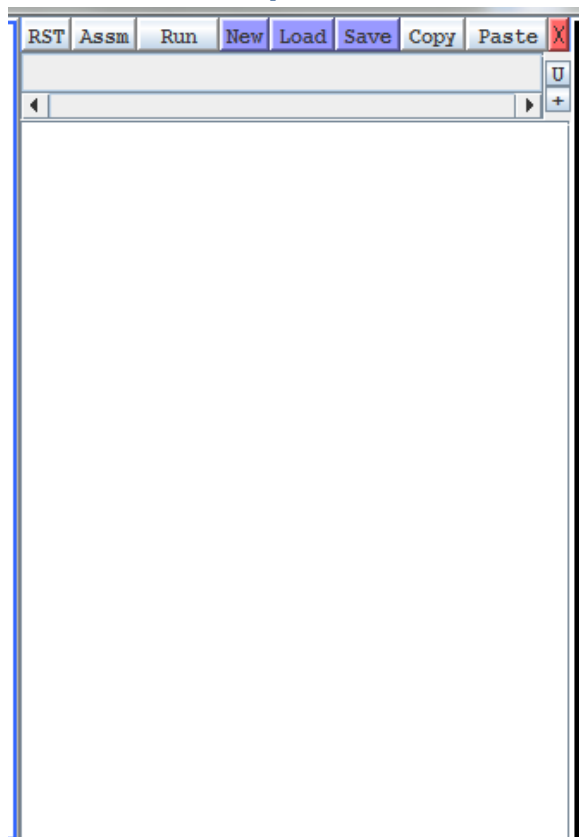
Data Memory		Modify RAM																			
		addr		0x00		0x00		value													
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F				
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				

U podatkovnu memoriju se upisuju podaci koji su potrebni za izvršavanje koda ili rada procesora. U podatkovnu memoriju podaci se mogu upisati putem koda naredbe MOV (adresa\_u\_memoriji),#(podatak) ili putem mikrokontrolerskog panela upisivanjem adrese i podatka u bijeli prozorčić Modify RAM iznad prozora memorije i pritiskom na Enter:

		Modify RAM																			
		addr		0x05		0x20		value													
		4	5	6	7	8	9	A	B	C	D	E	F								
00	20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				

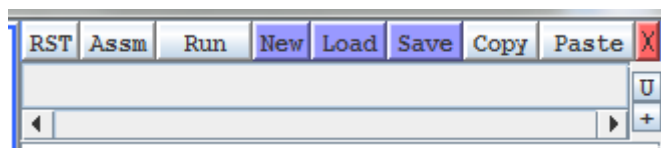


## 4 Assembler panel

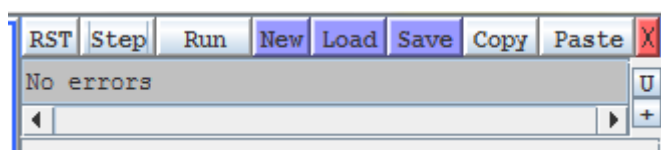


U ovaj panel se upisuje kod koji će procesor izvršavati. Kod se može izvršiti korak po korak ili odjednom ovisno o potrebi. Sve naredbe upisane utječu na stanja u registrima i memoriji procesora koje možemo pratiti na panelu mikrokontrolera.

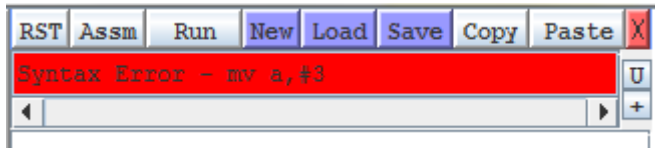
### 4.1 Alatna traka



Ovdje upravljamo izvršavanjem koda i uređivanjem koda. Tri ljubičasta polja: New, Load i Save služe za upravljanje samim kodom na način da otvorimo novi projekt (New), učitamo stari (Load) ili spremimo trenutni projekt (Save). Lijevo od ovih naredbi nalaze se polja za upravljanje izvršavanja koda: RST, Assm i Run. Pritiskom Assm EdSim51 će assemblerirati kod i unijeti ga u kodnu memoriju. U isto vrijeme će ispitati ima li sintaksnih pogrešaka u napisanom kodu. Ukoliko nema polje Assm će se pretvoriti u polje Step i ispisati će se poruka No errors:

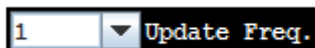


Ukoliko dođe do pogreške u pisanju koda program će izbaciti poruku o dijelu koda koji nije ispravan:

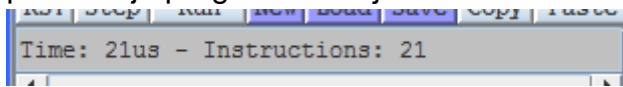


Pritiskom na Step program će izvršavati liniju po liniju koda. Ovo omogućuje jednostavno praćenje promjena u registrima i zastavicama.

Pritiskom na Run program će se izvršiti od početka do kraja bez stajanja po frekvenciji koja je zadana. Frekvenciju možemo mijenjati upisom broja u Update Freq:



. Nakon pritiska Run prikazat će se vrijeme proteklo od pokretanja programa i broj izvršenih instrukcija:



i pojavit će se polje Pause kojim pauziramo

izvođenje instrukcija: 

Nakon izvršavanja koda potrebno je resetirati assembler kako bi se kod izmijenio ili ponovo pokrenuo. To radimo pritiskom na gumb RST.

Desno se nalaze dva polja: Copy i Paste koji služe za kopiranje dijelova koda ili kopiranje teksta iz Clipboarda.

## 4.2 Primjer izvršavanja asemblerskog koda

Procesor 8051 ima 256 instrukcija. Svaka instrukcija ima svoj kod. Iste instrukcije imaju više različitih kodova zbog upravljanja adresama. Npr. instrukcija MOV ima 58 različitih kodova zbog toga što moraju postojati različiti kodovi za premještanje svih kombinacija operanda, adresa i registra.

## Instructions by opcode

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0a	0x0b	0x0c	0x0d	0x0e	0x0f
0x00	<a href="#">NOP</a>	<a href="#">AJMP</a>	<a href="#">LJMP</a>	<a href="#">RR</a>	<a href="#">INC</a>	<a href="#">INC</a>	<a href="#">INC</a>	<a href="#">INC</a>	<a href="#">INC</a>	<a href="#">INC</a>	<a href="#">INC</a>	<a href="#">INC</a>	<a href="#">INC</a>	<a href="#">INC</a>	<a href="#">INC</a>	<a href="#">INC</a>
0x10	<a href="#">JBC</a>	<a href="#">ACALL</a>	<a href="#">LCALL</a>	<a href="#">RRC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>	<a href="#">DEC</a>
0x20	<a href="#">JB</a>	<a href="#">AJMP</a>	<a href="#">RET</a>	<a href="#">RL</a>	<a href="#">ADD</a>	<a href="#">ADD</a>	<a href="#">ADD</a>	<a href="#">ADD</a>	<a href="#">ADD</a>	<a href="#">ADD</a>	<a href="#">ADD</a>	<a href="#">ADD</a>	<a href="#">ADD</a>	<a href="#">ADD</a>	<a href="#">ADD</a>	<a href="#">ADD</a>
0x30	<a href="#">JNB</a>	<a href="#">ACALL</a>	<a href="#">RETI</a>	<a href="#">RLC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>	<a href="#">ADDC</a>
0x40	<a href="#">JC</a>	<a href="#">AJMP</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>	<a href="#">ORL</a>
0x50	<a href="#">JNC</a>	<a href="#">ACALL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>	<a href="#">ANL</a>
0x60	<a href="#">JZ</a>	<a href="#">AJMP</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>	<a href="#">XRL</a>
0x70	<a href="#">JNZ</a>	<a href="#">ACALL</a>	<a href="#">ORL</a>	<a href="#">JMP</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>
0x80	<a href="#">SJMP</a>	<a href="#">AJMP</a>	<a href="#">ANL</a>	<a href="#">MOVC</a>	<a href="#">DIV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>
0x90	<a href="#">MOV</a>	<a href="#">ACALL</a>	<a href="#">MOV</a>	<a href="#">MOVC</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>	<a href="#">SUBB</a>
0xa0	<a href="#">ORL</a>	<a href="#">AJMP</a>	<a href="#">MOV</a>	<a href="#">INC</a>	<a href="#">MUL</a>	?	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>
0xb0	<a href="#">ANL</a>	<a href="#">ACALL</a>	<a href="#">CPL</a>	<a href="#">CPL</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>	<a href="#">CJNE</a>
0xc0	<a href="#">PUSH</a>	<a href="#">AJMP</a>	<a href="#">CLR</a>	<a href="#">CLR</a>	<a href="#">SWAP</a>	<a href="#">XCH</a>	<a href="#">XCH</a>	<a href="#">XCH</a>	<a href="#">XCH</a>	<a href="#">XCH</a>	<a href="#">XCH</a>	<a href="#">XCH</a>	<a href="#">XCH</a>	<a href="#">XCH</a>	<a href="#">XCH</a>	<a href="#">XCH</a>
0xd0	<a href="#">POP</a>	<a href="#">ACALL</a>	<a href="#">SETB</a>	<a href="#">SETB</a>	<a href="#">DA</a>	<a href="#">DJNZ</a>	<a href="#">XCHD</a>	<a href="#">XCHD</a>	<a href="#">DJNZ</a>	<a href="#">DJNZ</a>	<a href="#">DJNZ</a>	<a href="#">DJNZ</a>	<a href="#">DJNZ</a>	<a href="#">DJNZ</a>	<a href="#">DJNZ</a>	<a href="#">DJNZ</a>
0xe0	<a href="#">MOVX</a>	<a href="#">AJMP</a>	<a href="#">MOVX</a>	<a href="#">MOVX</a>	<a href="#">CLR</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>
0xf0	<a href="#">MOVX</a>	<a href="#">ACALL</a>	<a href="#">MOVX</a>	<a href="#">MOVX</a>	<a href="#">CPL</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>	<a href="#">MOV</a>

- [ACALL](#) - Absolute Call
- [ADD, ADDC](#) - Add Accumulator (With Carry)
- [AJMP](#) - Absolute Jump
- [ANL](#) - Bitwise AND
- [CJNE](#) - Compare and Jump if Not Equal
- [CLR](#) - Clear Register
- [CPL](#) - Complement Register
- [DA](#) - Decimal Adjust
- [DEC](#) - Decrement Register
- [DIV](#) - Divide Accumulator by B
- [DJNZ](#) - Decrement Register and Jump if Not Zero
- [INC](#) - Increment Register
- [JB](#) - Jump if Bit Set
- [JBC](#) - Jump if Bit Set and Clear Bit
- [JC](#) - Jump if Carry Set
- [JMP](#) - Jump to Address
- [JNB](#) - Jump if Bit Not Set
- [JNC](#) - Jump if Carry Not Set
- [JNZ](#) - Jump if Accumulator Not Zero
- [JZ](#) - Jump if Accumulator Zero
- [LCALL](#) - Long Call
- [LJMP](#) - Long Jump
- [MOV](#) - Move Memory
- [MOVC](#) - Move Code Memory
- [MOVB](#) - Move Extended Memory
- [MUL](#) - Multiply Accumulator by B
- [NOP](#) - No Operation
- [ORL](#) - Bitwise OR
- [POP](#) - Pop Value From Stack
- [PUSH](#) - Push Value Onto Stack
- [RET](#) - Return From Subroutine
- [RETI](#) - Return From Interrupt
- [RL](#) - Rotate Accumulator Left
- [RLC](#) - Rotate Accumulator Left Through Carry
- [RR](#) - Rotate Accumulator Right
- [RRC](#) - Rotate Accumulator Right Through Carry
- [SETB](#) - Set Bit
- [SJMP](#) - Short Jump
- [SUBB](#) - Subtract From Accumulator With Borrow
- [SWAP](#) - Swap Accumulator Nibbles
- [XCH](#) - Exchange Bytes
- [XCHD](#) - Exchange Digits
- [XRL](#) - Bitwise Exclusive OR
- [Undefined](#) - Undefined Instruction

Kao primjer assembler koda uzet ću zbrajanje i množenje operanda 2Ah i 33h.

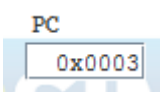
#### 4.2.1 Zbrajanje i komplement

Upisati operande 2Ah na adresu 20h i operand 33h na adresu 30h. Operande zbrojiti, a rezultat komplementirati.



Operandi s kojima radimo (spremamo, računamo...) u kodu se pišu sa # oznakom prije vrijednosti. Adrese se pišu bez ikakve oznake prije broja adrese. Iza svakog broja može stajati indikator koji određuje iz kojeg je brojevnog sustava broj. Taj indikator može biti **h**, **b** ili ništa. Slovo **h** označava da broj pripada heksidecimalnom sustavu, **b** označava da pripada binarnom sustavu, a ostavimo li prazno, broj pripada decimalnom sustavu.

Izvršavanje pokrećemo postupno Step po Step. Svaki put nakon pritiska Step program kreće u novi red i programsko brojilo (PC) se pomiče za određeni broj. Programsko brojilo služi za pohranjivanje adrese sljedeće naredbe. Ono se najčešće inkrementira (poveća za 1) nakon svake izvršene instrukcije. Upravljačka jedinica čita adresu u programskom brojiu i dohvaća instrukciju. Kod se pohranjuje u instrukcijski registar nakon čeka se ona

izvršava, a programsko brojilo se inkrementira.  - nakon izvršavanja prve linije assembler koda. To znači da je programsko brojilo krenulo brojati od 0000 i pribavilo instrukcije s kodom 75. Instrukcija koda 75 je MOV instrukcija koja neku vrijednost zadanu u kodu premješta u određenu adresu u memoriji. U ovom slučaju naredba MOV će u adresu 20h premjestiti vrijednost 2Ah.

Code Memory				
	0	1	2	3
00	75	20	2A	75
10	00	00	00	00

Nakon izvršavanja MOV instrukcije u memoriji na adresi 20 se pojavila vrijednost 2A.

Data Memory			
	0	1	2
00	00	00	00
10	00	00	00
20	2A	00	00
30	00	00	00
40	00	00	00
50	00	00	00

Ponovo klik na Step i počinje se izvršavati sljedeća instrukcija.

Code Memory																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	75	20	2A	75	30	33	E5	20	25	30	F4	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

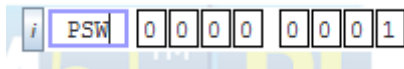
Opet se izvršava MOV instrukcija koja sprema na adresu 30h vrijednost 33h.

	0	1	2	3
00	00	00	00	00
10	00	00	00	00
20	2A	00	00	00
30	33	00	00	00
40	00	00	00	00
50	00	00	00	00

Programsko brojilo se povećalo za 3 i sljedeća instrukcija koja će se izvršiti je instrukcija MOV koja iz neke memorijske adrese vrijednost prebacuje u akumulator.

ACC | 0x2A

Nakon izvršavanja vrijednost s adrese 20h je prebačena u akumulator kako bi se mogle izvršiti aritmetičke operacije.



U tražilicu upisati PSW kako bi mogli pratiti promjene na zastavicama procesora. Nakon prebacivanja vrijednosti u akumulator, u PSW se upalila zastavica P (Parity, PSW.0) koja označava da broj bitova u log. 1 nije paran.  $2A_{16} = 00101010_2$ .

Sljedeća naredba ima kod 25 i nalazi se na adresi 0008h u kodnoj memoriji. To je naredba ADD koja akumulatoru dodaje vrijednost s neke adrese u memoriji. Akumulatoru je dodana vrijednost s adrese 30h.

$$\begin{array}{r}
 2A \\
 +33 \\
 \hline
 5D
 \end{array}
 \qquad
 \text{Binarno:}
 \qquad
 \begin{array}{r}
 00101010 \\
 +00110011 \\
 \hline
 01011101
 \end{array}$$

ACC | 0x5D

- stanje u akumulatoru

Svi rezultati aritmetičkih i logičkih operacija vraćaju se u akumulator.

Nakon zbrajanja dolazi instrukcija CPL na adresi 000Ah koja je ujedno i zadnja instrukcija. Ona će vrijednosti u akumulatoru zamijeniti 0 i 1.

ACC | 0xA2

01011101 – 5D

10100010 – A2

Ovime dolazimo do naredbe END koda i kraja izvršavanja koda.

#### 4.2.2 Množenje i logičke operacije

Upisati operand 2Ah na adresu 40h i operand 33h na adresu 50h. Operandu 2Ah obrisati donja 4 bita koristeći logičku operaciju I, a zatim pomnožiti s operandom 33h.

```

    org 0000h
0000| mov 40h,#2Ah
0003| mov 50h,#33h
0006| mov a,40h
0008| anl a,#11110000b
000A| mov b,50h
000D| mul ab
    end

```

Nakon klika na gumb **Assm** kod je učitani u kodnu memoriju i izvršavanje može početi.

PC  
0x0000 Programsko brojilo je u 0 što znači da će sljedeća instrukcija biti ona na adresi 0000h.

Code Memory	addr	0x0000	0x75	value												
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	75	40	2A	75	50	33	E5	40	54	F0	85	50	F0	A4	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

To je instrukcija MOV s kodom 75 koja prebacuje na adresu 40h vrijednost 2Ah. Nakon nje se izvršava instrukcija na adresi 0003h – opet MOV koja prebacuje na adresu 50 u memoriji vrijednost 33h.

30	00	00
40	2A	00
50	33	00
60	00	00

- Vrijednosti su prebačene u memoriju, a programsko brojilo je sada 0006h

PC  
0x0006 . Sljedeća naredba je na adresi 0006h, a to je MOV

E5 40 naredba koja s adrese 40h prebacuje vrijednost u akumulator  
ACC 0x2A . PSW je opet poprimio vrijednost 1 jer broj jedinica nije paran.

D	UXUU
ACC	0x2A
PSW	0x01

Sljedeća instrukcija je na adresi 0008h zbog toga što naredba MOV s kodom E5 zauzima 2 ciklusa.

```

0008| anl a,#11110000b
000A| mov b,50h

```



Logička operacija I ima kod 54 i izvršava se između operanda 2Ah i operanda 11110000b (F0h) kako bi se pomoću digitalne logike „pobrisala“ donja 4 bita. **54 F0**

Operacija I daje logičku 1 samo ako su oba broja u logičkoj 1. To znači ako je samo jedan broj od operanda u 0 rezultat će biti 0.

2A	00101010
log. I F0	log. I 11110000
20	
	00100000

B	0x33
ACC	0x20
PSW	0x01

Rezultat se sprema u akumulator.

PSW	0	0	0	0	0	0	0	1
-----	---	---	---	---	---	---	---	---

PSW je opet u 1 zbog pariteta.

Sljedeća instrukcija je MOV koja prebacuje vrijednost na adresi 50h u privremeni registar B.

B	0x33
ACC	0x20
PSW	0x01

Njezin kod je 85.

Aritmetička operacija množenja može se izvršiti samo između vrijednosti akumulatora i privremenog registra B što nije bio slučaj kod ADD operacije. Zbog toga smo drugi operand morali prebaciti u registar B. Zbog toga što je moguće da vrijednost umnoška bude veća od FFh (koja je maksimalna veličina registra) gornji bajt se sprema u registar B, donji bajt u akumulator nakon izvršavanja operacije.

B	0x06
ACC	0x60
PSW	0x01

20	Decimalno: 32	Binarno: 00100000
* 33	* 51	* 00110011
660	1632	00000110   01100000

Donji bajt jednak je 60 ili 01100000b, a gornji bajt jednak je 6 ili 00000110b

PSW	0	0	0	0	0	1	0	0
-----	---	---	---	---	---	---	---	---

U PSW je ovaj put upisana zastavice OV (PSW.2). Ona označava prelijevanje vrijednosti u viši bajt tj. privremeni registar B. Zastavica P je u 0 jer vrijednosti ima paran broj jedinica. Nakon množenja dolazimo do naredbe END i program se završava.

## 5 Program status word (PSW)

Bit	Symbol	Address	Description
PSW.7	CY	D7H	Carry flag
PSW.6	AC	D6H	Auxiliary carry flag
PSW.5	F0	D5H	Flag 0
PSW.4	RS1	D4H	Register bank select 1
PSW.3	RS0	D3H	Register bank select 0
PSW.2	OV	D2H	Overflow flag
PSW.1	--	D1H	Reserved
PSW.0	P	D0H	Even parity flag

### 5.1 Carry flag

Carry zastavica ima dvije funkcije:

- Koristi se za pohranu bita prijenosa kod aritmetičkih operacija
- Koristi se kod Booleove algebre

### 5.2 Parity bit

Bit parnosti se automatski stavlja ako je broj jedinica (binarnih) u akumulatoru neparan ili miče ako je broj jedinica paran.

Bit parnosti se najčešće koristi za otkrivanje pogrešaka u prijenosu podataka.

### 5.3 Overflow zastavica

Koristi se kada rezultat usred aritmetičkih operacija prelazi -128 ili 127 . Ako prelazi 127 ili je manja od -128 zastavica je postavljena.

## 6 Primjer i objašnjenje izvođenja programa

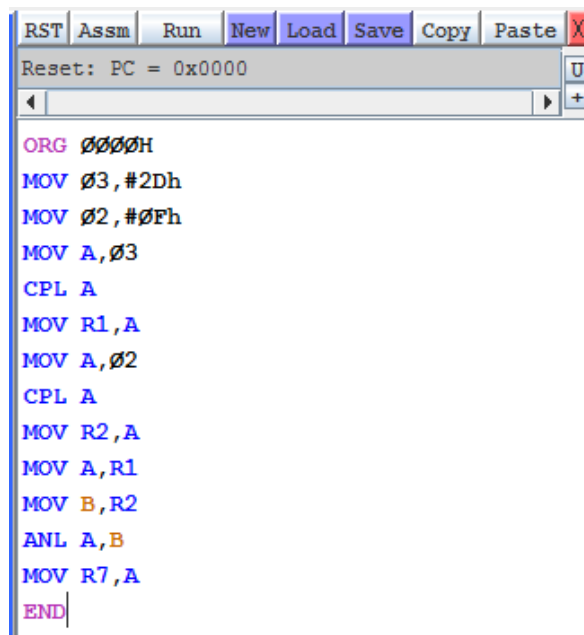
### Zadatak:

Izvesti logičku operaciju I nad komplementima operanda 2Dh i 0Fh koji se nalaze u memoriji na adresi 03 i 02, rezultat pohraniti u registar R7. Međurezultate spremiti u registre R1 i R2.

### 6.1 Zapis programa u assembleru

```
ORG 0000H
MOV 03,#2Dh
MOV 02,#0Fh
MOV A,03
CPL A
MOV R1,A
MOV A,02
CPL A
MOV R2,A
MOV A,R1
MOV B,R2
ANL A,B
MOV R7,A
END
```

Upisani program u simulatoru:



The screenshot shows the EdSim51 simulator window. The title bar contains buttons for RST, Assm, Run, New, Load, Save, Copy, Paste, and a close button (X). Below the title bar, it displays 'Reset: PC = 0x0000' and a 'U' button. A scroll bar is visible above the code area. The code area contains the following assembly code:

```
ORG 0000H
MOV 03,#2Dh
MOV 02,#0Fh
MOV A,03
CPL A
MOV R1,A
MOV A,02
CPL A
MOV R2,A
MOV A,R1
MOV B,R2
ANL A,B
MOV R7,A
END
```

Prikaz zapisa u programskoj memoriji

The screenshot shows the EdSim51 interface with the 'Code Memory' window open. The assembly code is as follows:

```

ORG 0000H
0000| MOV 03, #2Dh
0003| MOV 02, #0Fh
0006| MOV A, 03
0008| CPL A
0009| MOV R1, A
000A| MOV A, 02
000C| CPL A
000D| MOV R2, A
000E| MOV A, R1
000F| MOV B, R2
0011| ANL A, B
0013| MOV R7, A
END
    
```

The 'Code Memory' table shows the following data:

addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	75	03	2D	75	02	0F	E5	03	F4	F9	E5	02	F4	FA	E9	8A
10	FO	55	FO	FF	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Prikaz sadržaja podatkovne memorije i registara procesora:

The screenshot shows the EdSim51 interface with the 'Data Memory' window open. The processor registers are as follows:

- R7: 0x00, R6: 0x00, R5: 0x00, R4: 0x00, R3: 0x00, R2: 0x00, R1: 0x00, R0: 0x00
- B: 0x00, ACC: 0x00, PSW: 0x00, IP: 0x00, IE: 0x00, PCON: 0x00, DPH: 0x00, DPL: 0x00, SP: 0x07
- TH0: 0x00, TL0: 0x00, TH1: 0x00, TL1: 0x00
- PC: 0x00000
- PSW: 00000000

The 'Data Memory' table shows the following data:

addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Svi registri procesora i podatkovna memorija su prazni.

## 6.2 Objašnjenje koda

### ➤ ORG 0000h

- Određuje od koje adrese će se u kodnu memoriju upisivati kodovi instrukcija

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	75	03	2D	75	02	0F	E5	03	F4	F9	E5	02	F4	FA	E9	8A
10	F0	55	F0	FF	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

### ➤ MOV 03, #2Dh

- Na adresu 03 u podatkovnoj memoriji upisuje se podatak 2D

The screenshot shows the EdSim51 interface during the execution of the instruction `MOV 03, #2Dh`. The PC register is 8051. The Data Memory window shows the value 2D at address 03. The instruction list on the right shows the assembly code.

Address	Instruction
0000H	ORG 0000H
0003H	MOV 03, #2Dh
0003H	MOV 02, #0Fh
0006H	MOV A, 03
0008H	CPL A
0009H	MOV R1, A
000AH	MOV A, 02
000CH	CPL A
000DH	MOV R2, A
000EH	MOV A, R1
000FH	MOV B, R2
0011H	ANL A, B
0013H	MOV R7, A
	END

Spremljeni podatak na adresi 03

➤ **MOV 02, #0Fh**

- Na adresu 02 u podatkovnoj memoriji pohranjen je podatak 0F

The screenshot shows the EdSim51 interface. On the left, the register and pin status is displayed. The PC register is at 8051. The Data Memory table shows the value 0F at address 02. The assembly code window on the right shows the instruction MOV 02, #0Fh highlighted in red. The status bar indicates the instruction is executed at 0x0003.

addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	00	0F	2D	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Spremljeni podatak na adresi 02

➤ **MOVA, 03**

- Sadržaj lokacije 03 iz podatkovne memorije pohranjuje se u akumulator A

The screenshot shows the EdSim51 interface. The ACC register now contains 0x2D. The assembly code window on the right shows the instruction MOV A, 03 highlighted in red. The status bar indicates the instruction is executed at 0x0006.

addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	00	0F	2D	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Podatak iz 03 spremljen u akumulator

➤ **CPL A**

- Komplementira se sadržaj akumulatora
- Rezultat se nalazi u akumulatoru

The screenshot shows the EdSim51 interface during the execution of the CPL A instruction. The register window on the left shows the Accumulator (ACC) containing 0xD2. The instruction list on the right shows the current instruction at address 0008 is CPL A. The assembly code on the far right shows the sequence of instructions: MOV R3, #2Dh; MOV R2, #0Fh; MOV A, R3; CPL A; MOV R1, A; MOV A, R2; CPL A; MOV R2, A; MOV A, R1; MOV B, R2; ANL A, B; MOV R7, A; END.

Rezultat komplementiranja

➤ **MOV R1, A**

- Kopira sadržaj akumulatora u registar R1

The screenshot shows the EdSim51 interface during the execution of the MOV R1, A instruction. The register window on the left shows register R1 containing 0xD2. The instruction list on the right shows the current instruction at address 0009 is MOV R1, A. The assembly code on the far right shows the sequence of instructions: MOV R3, #2Dh; MOV R2, #0Fh; MOV A, R3; CPL A; MOV R1, A; MOV A, R2; CPL A; MOV R2, A; MOV A, R1; MOV B, R2; ANL A, B; MOV R7, A; END.



### ➤ MOVA, 02

- Sprema u akumulator podatak koji se nalazi na adresi 02

System Clock (MHz) 12.0 | 1 | Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x00	R6	0x00	ACC	0x0F
RxD	TxD	TMOD	0x00	R5	0x00	PSW	0x00
1	1	TCOD	0x00	R4	0x00	IP	0x00
SCON	0x00	TCOD	0x00	R3	0x2D	IE	0x00
				R2	0x0F	PCON	0x00
pins	bits	TH1	TL1	R1	0xD2	DPH	0x00
0xFF	0xFF	0x00	0x00	R0	0x00	DPL	0x00
0xFF	0xFF					SP	0x07
0xFF	0xFF						
0xFF	0xFF						

PC 8051 | PSW 00000000

Data Memory	addr	0x00	0x00	value
00	00	D2	0F	2D
10	00	00	00	00
20	00	00	00	00
30	00	00	00	00
40	00	00	00	00
50	00	00	00	00
60	00	00	00	00
70	00	00	00	00

Executed 0x000A: MOV A,02H | Time: 8us - In

```

ORG 0000H
0000 | MOV 03, #2Dh
0003 | MOV 02, #0Fh
0006 | MOV A, 03
0008 | CPL A
0009 | MOV R1, A
000A | MOV A, 02
000C | CPL A
000D | MOV R2, A
000E | MOV A, R1
000F | MOV B, R2
0011 | ANL A, B
0013 | MOV R7, A
END

```

### ➤ CPL A

- Komplementira podatak u akumulatoru

System Clock (MHz) 12.0 | 1 | Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x00	R6	0x00	ACC	0xF0
RxD	TxD	TMOD	0x00	R5	0x00	PSW	0x00
1	1	TCOD	0x00	R4	0x00	IP	0x00
SCON	0x00	TCOD	0x00	R3	0x2D	IE	0x00
				R2	0x0F	PCON	0x00
pins	bits	TH1	TL1	R1	0xD2	DPH	0x00
0xFF	0xFF	0x00	0x00	R0	0x00	DPL	0x00
0xFF	0xFF					SP	0x07
0xFF	0xFF						
0xFF	0xFF						

PC 8051 | PSW 00000000

Data Memory	addr	0x00	0x00	value
00	00	D2	0F	2D
10	00	00	00	00
20	00	00	00	00
30	00	00	00	00
40	00	00	00	00
50	00	00	00	00
60	00	00	00	00
70	00	00	00	00

Executed 0x000C: CPL A | Time: 9us - Instru

```

ORG 0000H
0000 | MOV 03, #2Dh
0003 | MOV 02, #0Fh
0006 | MOV A, 03
0008 | CPL A
0009 | MOV R1, A
000A | MOV A, 02
000C | CPL A
000D | MOV R2, A
000E | MOV A, R1
000F | MOV B, R2
0011 | ANL A, B
0013 | MOV R7, A
END

```

Komplementiran podatak



➤ **MOV R2, A**

- Kopira sadržaj akumulatora u registar R2

The screenshot shows the EdSim51 interface during the execution of the instruction `MOV R2, A`. The register window on the left shows the following values:

R7	0x00	B	0x00
R6	0x00	ACC	0xF0
R5	0x00	PSW	0x00
R4	0x00	IP	0x00
R3	0x2D	IE	0x00
<b>R2</b>	<b>0xF0</b>	PCON	0x00
R1	0xD2	DPH	0x00
R0	0x00	DPL	0x00
		SP	0x07

The PC register is 8051. The assembly code window on the right shows the instruction `MOV R2, A` at address `000D`, which is highlighted with a red box. The status bar indicates the instruction is executed at `0x000D: MOV R2,A` with a time of 10us.

Kopirani podatak iz akumulatora u R2

➤ **MOVA, R1**

- U akumulator upisan podatak iz registra R1

The screenshot shows the EdSim51 interface during the execution of the instruction `MOV A, R1`. The register window on the left shows the following values:

R7	0x00	B	0x00
R6	0x00	<b>ACC</b>	<b>0xD2</b>
R5	0x00	PSW	0x00
R4	0x00	IP	0x00
R3	0x2D	IE	0x00
R2	0xF0	PCON	0x00
R1	0xD2	DPH	0x00
R0	0x00	DPL	0x00
		SP	0x07

The PC register is 8051. The assembly code window on the right shows the instruction `MOV A, R1` at address `000E`, which is highlighted with a red box. The status bar indicates the instruction is executed at `0x000E: MOV A,R1` with a time of 11us.

- **Mov B, R2**
  - podatak iz R2 u registar B

System Clock (MHz) 12.0    1    Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	0x00	B	0xF0
0x00	0x00	0x00	0x00	R6	0x00	ACC	0xD2
RXD	TXD			R5	0x00	PSW	0x00
1	1	TMOD	0x00	R4	0x00	IP	0x00
SCON	0x00	TCON	0x00	R3	0x2D	IE	0x00
				R2	0xF0	PCON	0x00
pins	bits	TH1	TL1	R1	0xD2	DPH	0x00
0xFF	0xFF	P3	0x00	0x00	0x00	DPL	0x00
0xFF	0xFF	P2				SP	0x07
0xFF	0xFF	P1					
0xFF	0xFF	P0					

PC 8051    PSW 00000000

Data Memory		addr	0x00	0x00	value										
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	D2	F0	2D	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Executed 0x000F: MOV 0F0H,R2 | Time: 13us

```

ORG 0000H
0000| MOV 03, #2Dh
0003| MOV 02, #0Fh
0006| MOV A, 03
0008| CPL A
0009| MOV R1, A
000A| MOV A, 02
000C| CPL A
000D| MOV R2, A
000E| MOV A, R1
000F| MOV B, R2
0011| ANL A, B
0013| MOV R7, A
END

```

- **ANL A,B**
  - Obavlja logičku funkciju I nad podacima koji se nalaze u registrima A i B
  - Rezultat će biti zapisan u akumulatoru

System Clock (MHz) 12.0    1    Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	0x00	B	0xF0
0x00	0x00	0x00	0x00	R6	0x00	ACC	0xD0
RXD	TXD			R5	0x00	PSW	0x01
1	1	TMOD	0x00	R4	0x00	IP	0x00
SCON	0x00	TCON	0x00	R3	0x2D	IE	0x00
				R2	0xF0	PCON	0x00
pins	bits	TH1	TL1	R1	0xD2	DPH	0x00
0xFF	0xFF	P3	0x00	0x00	0x00	DPL	0x00
0xFF	0xFF	P2				SP	0x07
0xFF	0xFF	P1					
0xFF	0xFF	P0					

PC 8051    PSW 00000001

Data Memory		addr	0x00	0x00	value										
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	D2	F0	2D	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Executed 0x0011: ANL A,0F0H | Time: 14us

```

ORG 0000H
0000| MOV 03, #2Dh
0003| MOV 02, #0Fh
0006| MOV A, 03
0008| CPL A
0009| MOV R1, A
000A| MOV A, 02
000C| CPL A
000D| MOV R2, A
000E| MOV A, R1
000F| MOV B, R2
0011| ANL A, B
0013| MOV R7, A
END

```

Rezultat logičke operacije I nad operandima F0 i D2

➤ **Mov R7, A**

- Kopira podatak iz akumulatora u registar R7

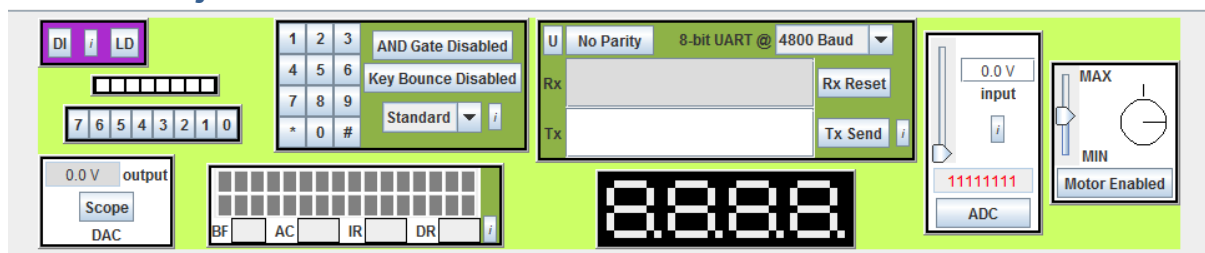
The screenshot displays the EdSim51 interface. On the left, the register file shows R7 containing 0xD0. The PC register is 8051. The assembly code window on the right shows the instruction MOV R7, A at address 0013, which is highlighted with a red box. The status bar indicates the instruction is executed at 0x0013 in 15us.

Rezultat iz akumulatora kopiran u R7

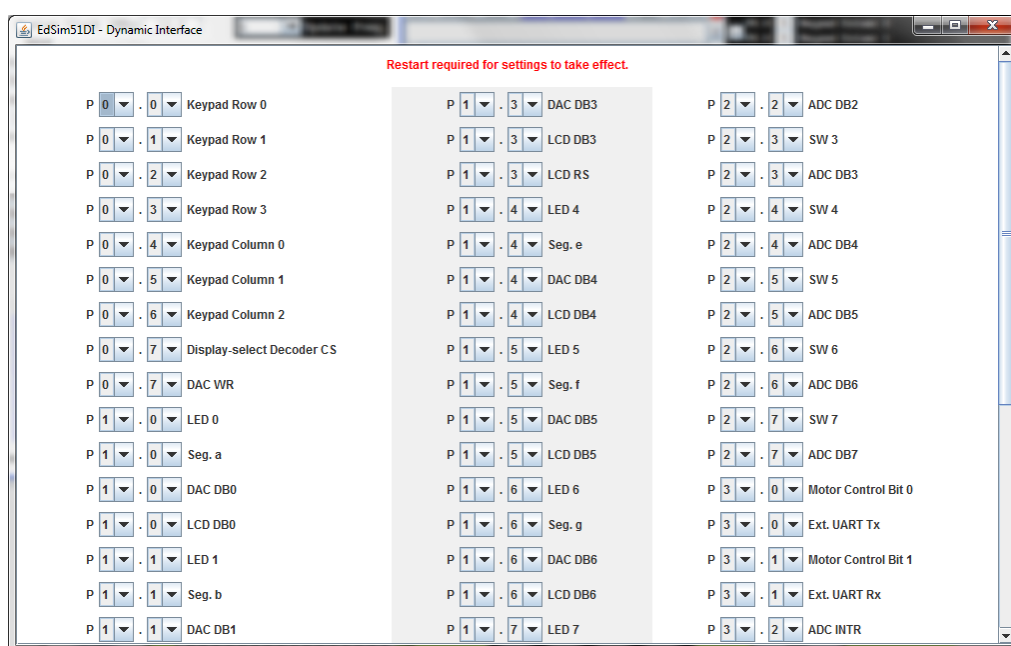
➤ **END**

- Oznaka za kraj programa

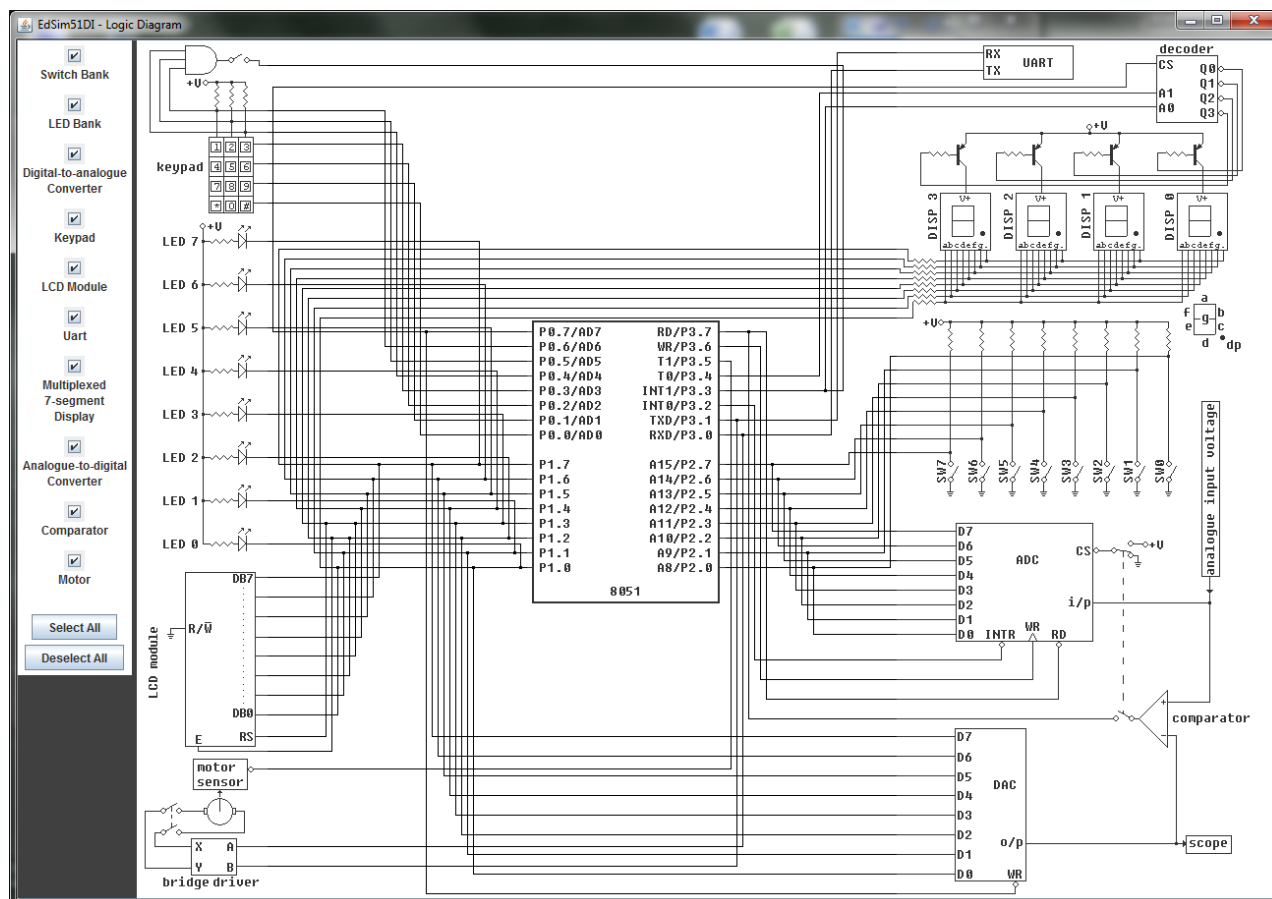
## 7 Periferija



Svako računalo mora moći primiti podatke iz okoline i negdje ispisati rezultate. Za to nam služi periferno sučelje. Ono se može podesiti pritiskom na DI nakon čega se program treba ponovo pokrenuti kako bi se primijenile promjene.



Pritiskom na LD otvara se logički dijagram ulazno/izlaznog sučelja.



Periferne jedinice uključene u program:

- ADC
- Komparator
- 4 7-segmentna displaya
- LCD modul
- UART
- Keypad
- LED bank
- DAC

Sa lijeve strane prozora nalazi se popis portova i uređaja koji su spojeni na svaki od portova.

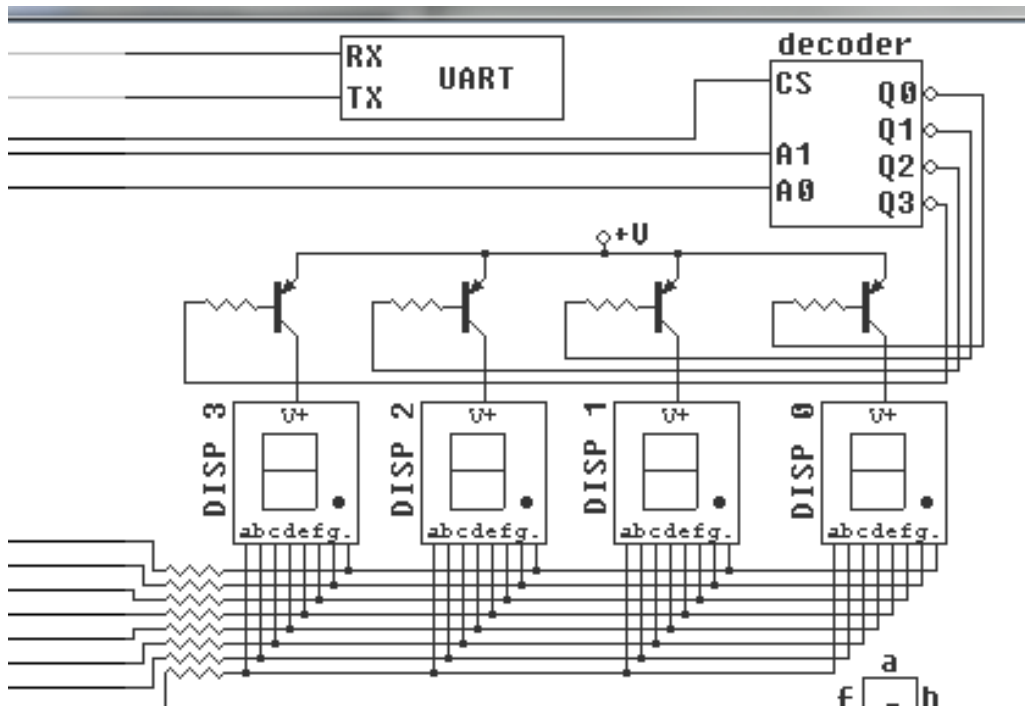
P0.7	1	Display-select Decoder CS DAC WR
P0.6	1	Keypad Column 2
P0.5	1	Keypad Column 1
P0.4	1	Keypad Column 0
P0.3	1	Keypad Row 3
P0.2	1	Keypad Row 2
P0.1	1	Keypad Row 1
P0.0	1	Keypad Row 0
P1.7	1	LED 7 Seg. g DAC DB7 LCD DB7
P1.6	1	LED 6 Seg. f DAC DB6 LCD DB6
P1.5	1	LED 5 Seg. e DAC DB5 LCD DB5
P1.4	1	LED 4 Seg. d DAC DB4 LCD DB4
P1.3	1	LED 3 ... d .DB3 .DB3... RS
P1.2	1	LED 2 ... c .DB2 .DB2 LCD E
P1.1	1	LED 1 Seg. b DAC DB1 LCD DB1
P1.0	1	LED 0 Seg. a DAC DB0 LCD DB0
P2.7	1	SW 7 ADC DB7
P2.6	1	SW 6 ADC DB6
P2.5	1	SW 5 ADC DB5
P2.4	1	SW 4 ADC DB4
P2.3	1	SW 3 ADC DB3
P2.2	1	SW 2 ADC DB2
P2.1	1	SW 1 ADC DB1
P2.0	1	SW 0 ADC DB0
P3.7	1	ADC RD Comparator Output
P3.6	1	ADC WR
P3.5	1	Motor Sensor
P3.4	1	Display-select Input 1
P3.3	1	AND Gate Output Display-se..t 0
P3.2	1	ADC INTR
P3.1	1	Motor Control Bit 1 Ext. UART Rx
P3.0	1	Motor Control Bit 0 Ext. UART Tx

Ukoliko želimo vidjeti portove uređaja u posebnom prozoru možemo pritisnuti na +.

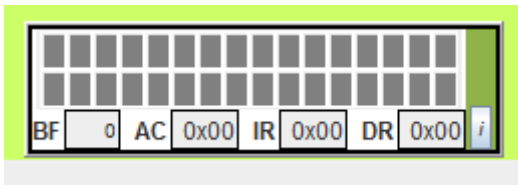
PORT 2		
P2.7	1	SW 7 ADC DB7
P2.6	1	SW 6 ADC DB6
P2.5	1	SW 5 ADC DB5
P2.4	1	SW 4 ADC DB4
P2.3	1	SW 3 ADC DB3
P2.2	1	SW 2 ADC DB2
P2.1	1	SW 1 ADC DB1
P2.0	1	SW 0 ADC DB0

always on top

LED bank, DAC i 7-segmentni display dijele iste port linije na port 1. Odabir između 4 7-segmentna display se obavlja pomoću P3.3 i P3.4 porta. Ovi pinovi portova dolaze na dekodler čija se vrijednost na izlazu primjenjuje kako bi odabrao jedan od 4 displaya.

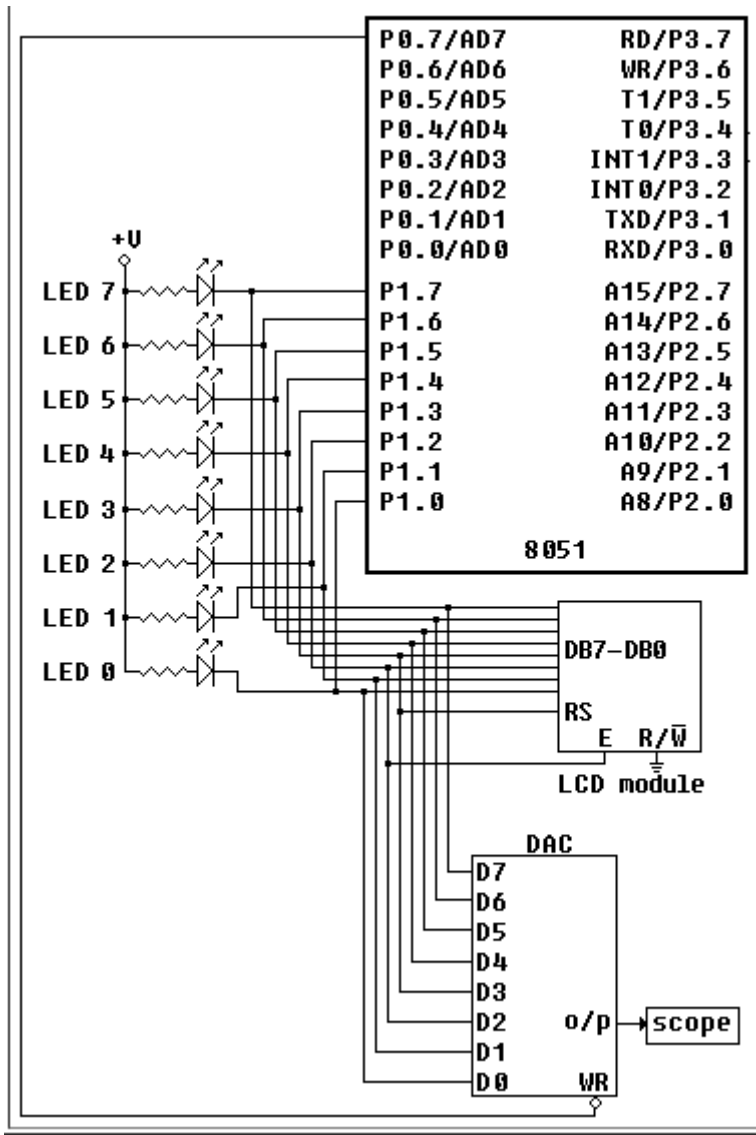


LCD modul dijeli port 1 sa LED displayem i DAC-om.



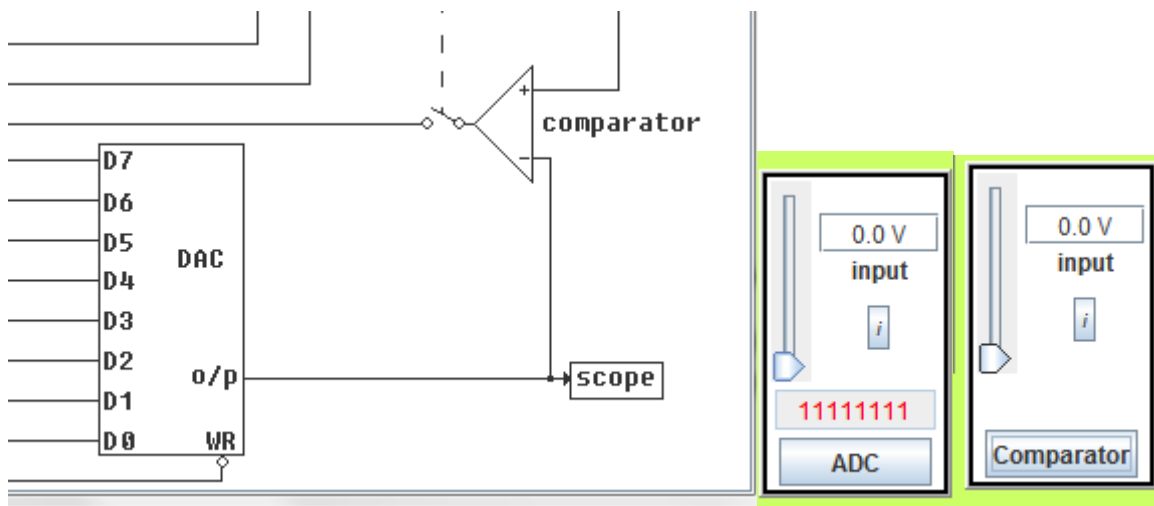
LCD modul može raditi u 4-bitnom i 8-bitnom modu.

U 4-bitnom modu on je simulacija Hitachi HD44780. Pinovi P1.7 do P1.4 spojeni su na DB7 do DB4 na DAC-u dok je P1.3 pin spojen kao odabirni bit, a P1.2 kao enable pin. Moguće je samo upisivati na modul, a ne i ispisivati.



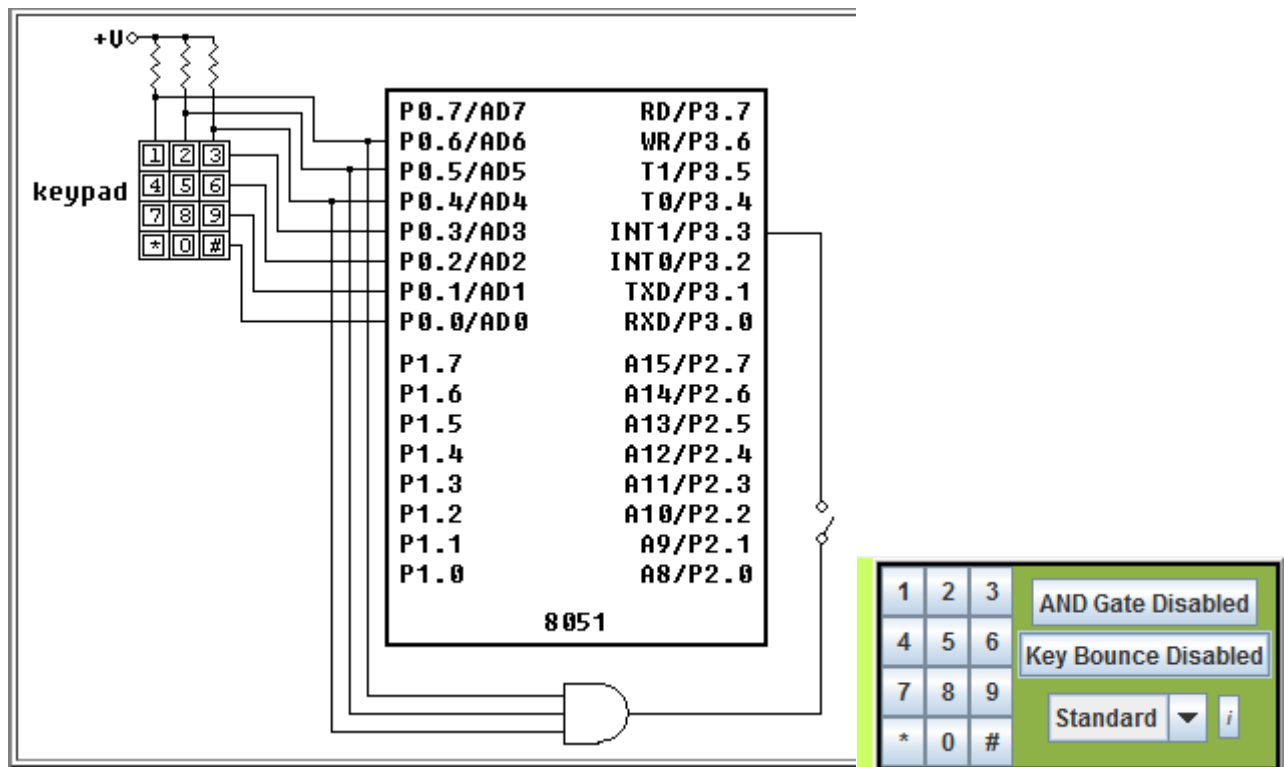
Početno je LCD modul u 4-bitnom modu. Da bi se prebacio u rad u 8-bitni mod potrebno je u DI prozoru promijeniti pinove porta.

Komparator i DAC:





Keypad:



Brojevi na keypadu mogu se mijenjati desnim klikom na broj ili u DI prozoru.

## 8 Literatura

<http://www.edsim51.com/index.html>

<http://pdf1.alldatasheet.com/datasheet-pdf/view/107780/INTEL/8051.html>