**EdSim51**

**Milan Korać @ Racunarstvo**

Sadržaj

[EdSim51 2](#_Toc436906916)

[Panel mikrokontrolera 3](#_Toc436906917)

[Registri procesora 4](#_Toc436906918)

[Kodna memorija 5](#_Toc436906919)

[Podatkovna memorija 6](#_Toc436906920)

[Assembler panel 6](#_Toc436906921)

[Alatna traka 7](#_Toc436906922)

[Primjer izvršavanja assembler koda 8](#_Toc436906923)

[Zbrajanje i komplement 9](#_Toc436906924)

[Množenje i logičke operacije 11](#_Toc436906925)

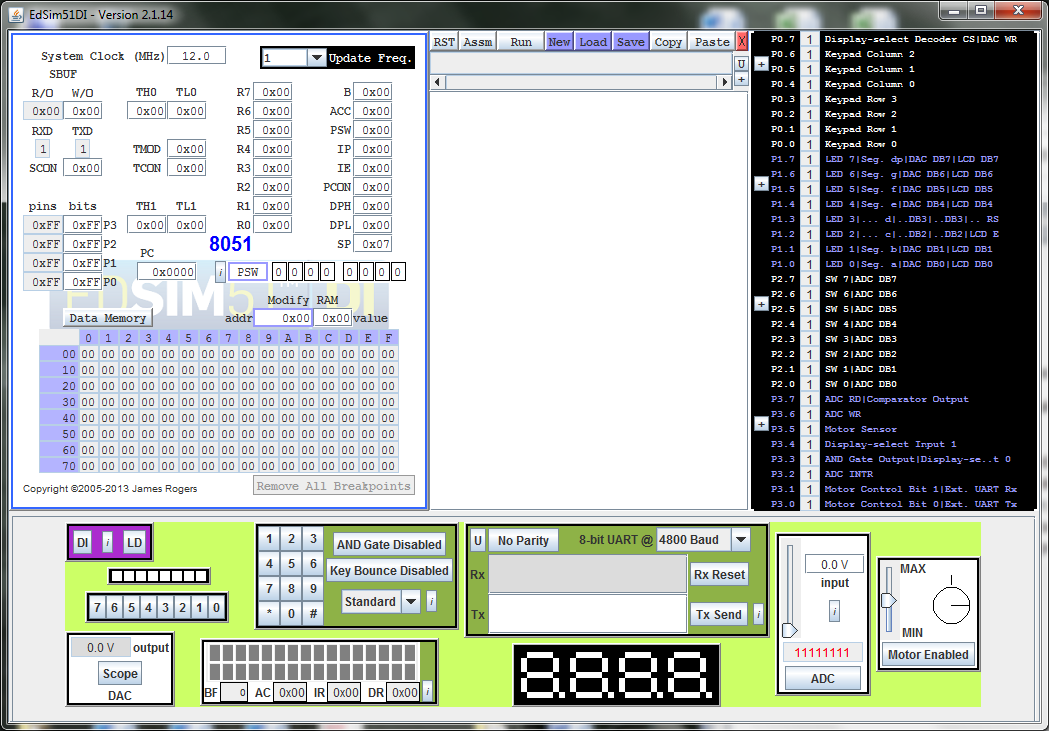
[Periferija 14](#_Toc436906926)

[Zaključak 18](#_Toc436906927)

# EdSim51

EdSim51 je simulator rada Intel-ovog 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:

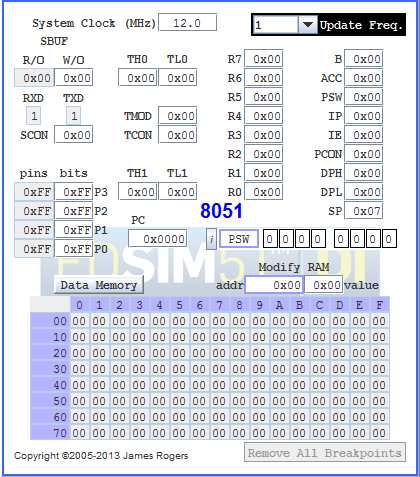


Grafičko sučelje programa EdSim51 je organizirano pomoću prozorima, gdje svaki prozor predstavlja drukčiji tip informacija.

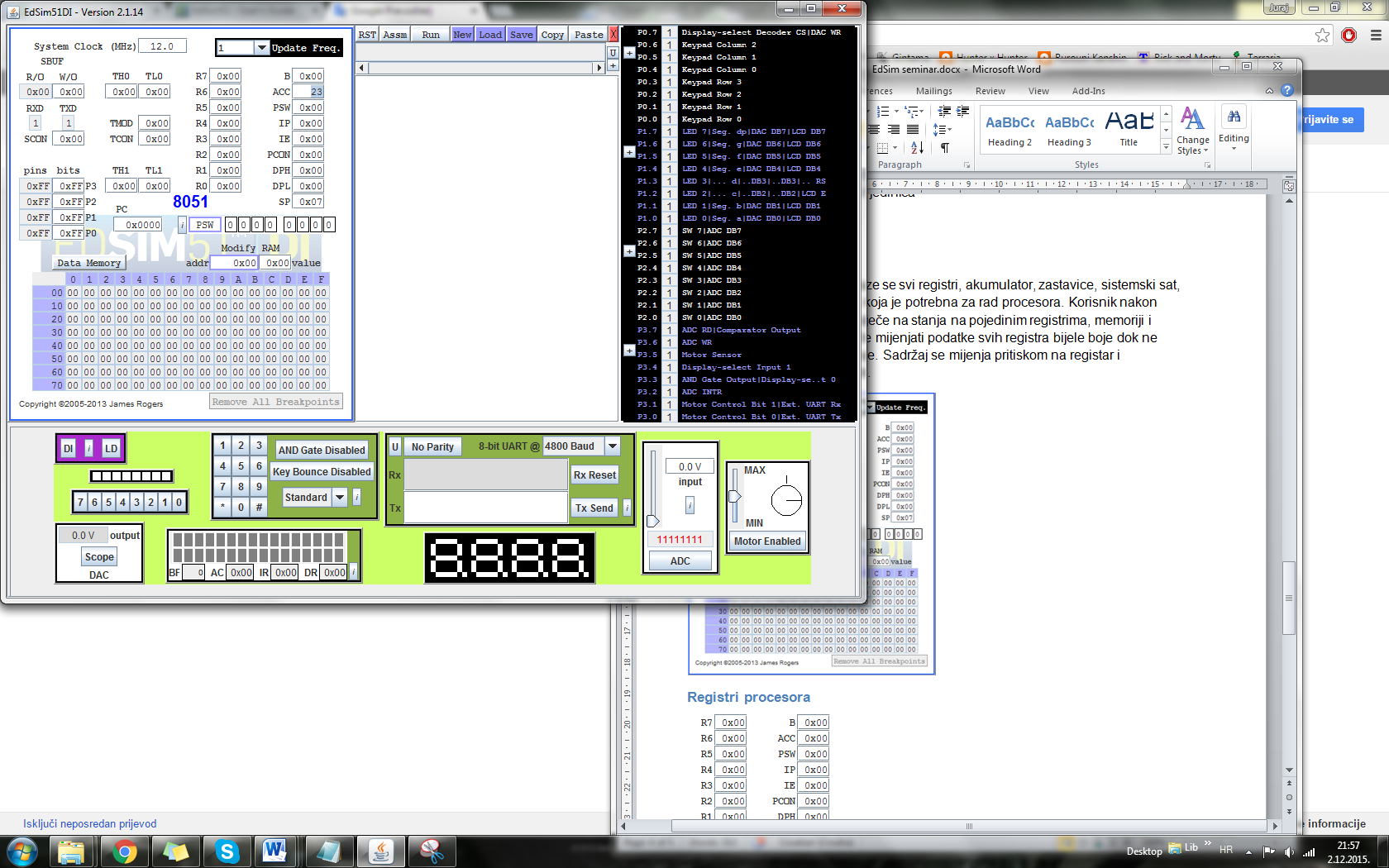
Program ima 4 dijela:

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

# Panel mikrokontrolera

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

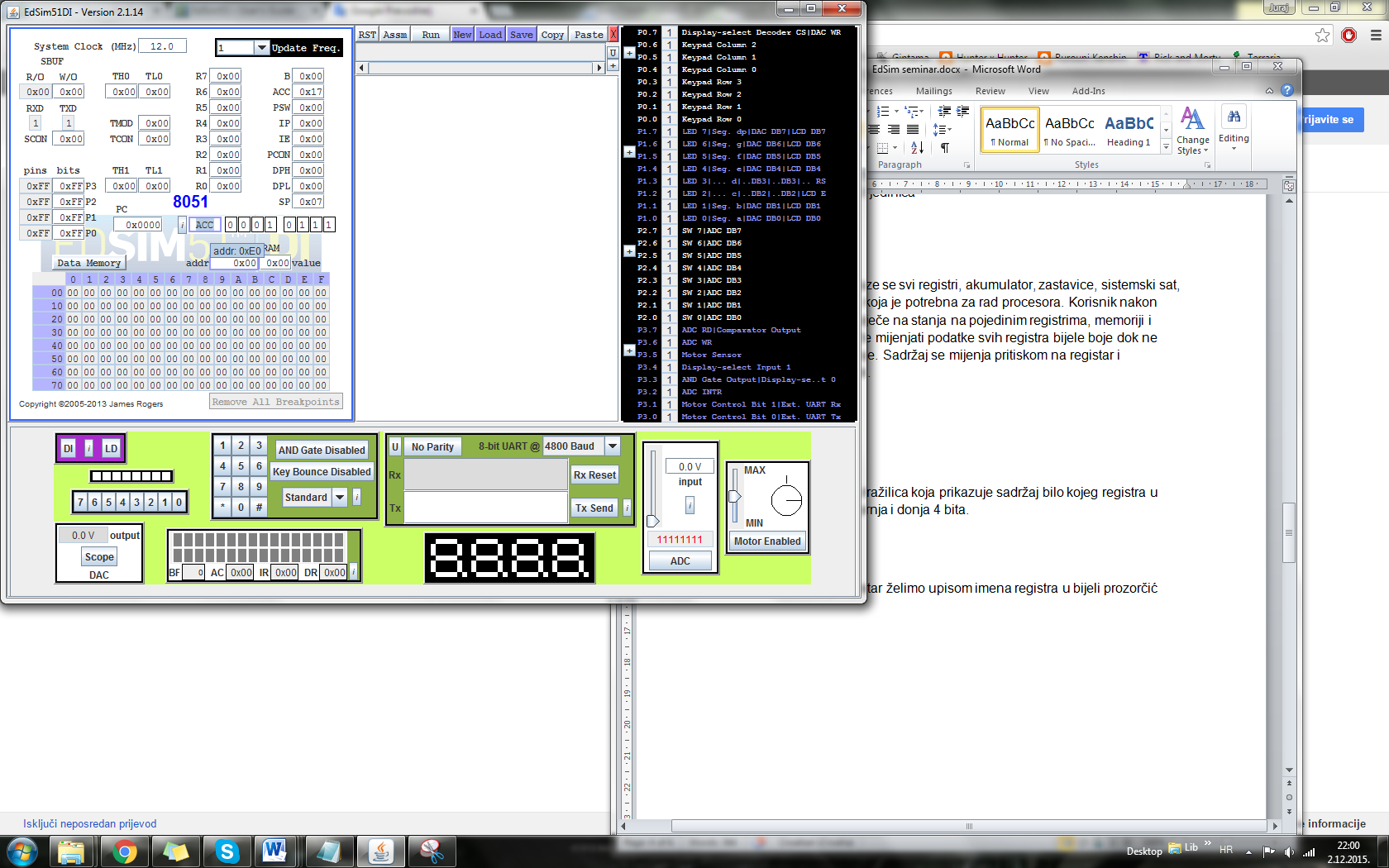
Korisnik nakon upisivanja i pokretanja koda utječe na stanja na pojedinim registrima, memoriji i zastavicama. Korisnik još može mijenjati podatke svih registra bijele boje dok ne može mijenjati registre sive boje. Sadržaj se mijenja pritiskom na registar i upisivanjem željene vrijednosti.



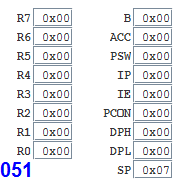
Ovdje se još nalazi i posebna tražilica koja prikazuje sadržaj bilo kojeg registra u binarnom zapisu odvajajući gornja i donja 4 bita.



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

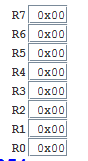


## Registri procesora



Procesor 8051 sastoji se od više registara od kojih je najvažniji akumulator (ACC) . U akumulator se smještaju svi podatci logičkih i aritmetičkih operacija i operandi i rezultat operacije. On je povezan sa aritmetičko-logičkom jedinicom procesora.

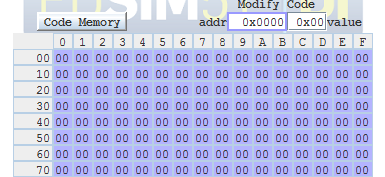
Kao pomoćni registar tu je registar B . U njega se također upisuju podatci koji će ući u aritmetičko-logičku jedinicu, ali samo operandi. On se još naziva i privremeni registar.

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)...

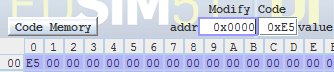
PSW se sastoji od 8 bitova (zastavica) kojima aritmetičko-logička jedinica utječe na kontrolnu jedinicu postavljajući zastavice. PSW se sastoji od: CY carry zastavice (PSW.7), AC pomoćne carry zastavice (PSW.6), F0 zastavice 0 koja se koristi u općenite svrhe (PSW.5), RS1 register bank select bit 1(PSW.4), RS0 register bank select bit 0(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).

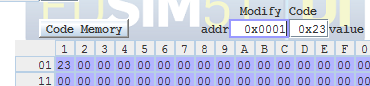
## Kodna memorija

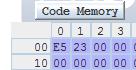


Kodna memorija zajendo s podatkovnom memorijom čini radnu memoriju. Korisnik može mijenjati između prikaza kodne i podatkovne meorije pritiskom na ime trenutne memorije: .

U kodnu memoriju se upisuju kodovi instrukcija koje se izvršavaju po redu. Kodna memorija se sama popuni nakon assembliranja 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.

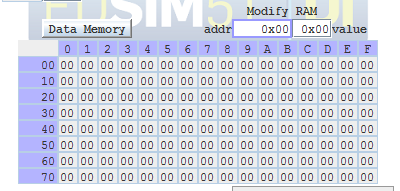


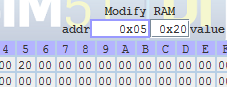




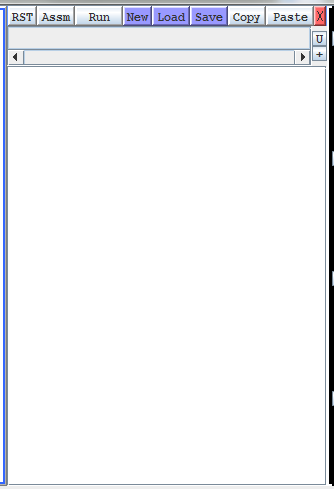
Ovaj kod će izvršiti instrukciju MOV A,#23h. Kod instrukcije MOV koji prebacuje vrijednost operanda u akumulator je E5.

## Podatkovna memorija



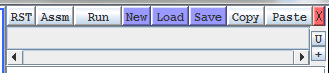
U podatkovnu memoriju se upsuju podatci koji su potrebni za izvršavanje koda ili rada procesora. U podatkovnu memoriju podaci se mogu upisati putem koda naredbe MOV (adresa\_u\_memoriji),#(podatak) ili putem mikrokontrolerskog panela upisivanjem adrese i podatka u bijeli prozorčić Modify RAM iznad prozora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

# Assembler panel

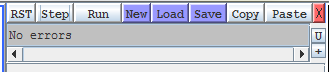


U ovaj panel se upisuje kod koji će procesor izvršavati. Kod se može izvršiti korak po korak ili odjednom ovisno o potrebi. Sve naredbe upisane utječu na stanja u registrima i memoriji procesora koje možemo pratiti na panelu mikrokontrolera.

## 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 assemblirati kod i unjeti 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 svakim pritiskom na Step. Ovo omogućuje jednostavno pračenje promjena u registrima i zastavicama.

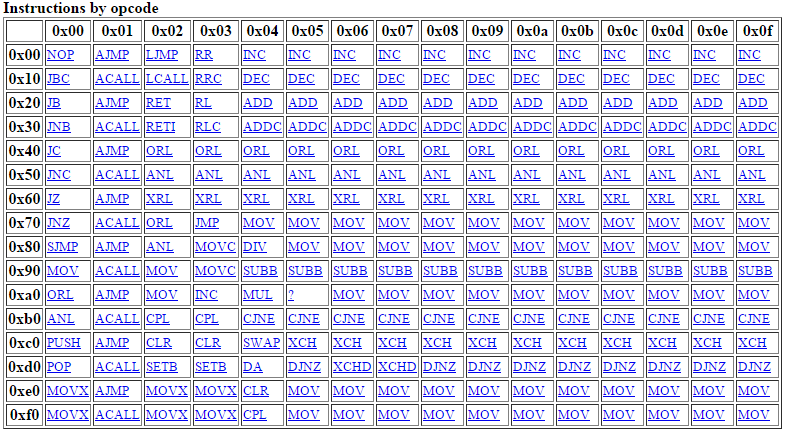
Pritiskom na Run program će se izvršiti od početka do kraja bez stajanja po frekvenciji koja je zadana. Frekvenciju možemo mijenjati upisom broja u Update Freq: . Nakon pritiska Run prikazat će se vrijeme proteklo od pokretanja programa i broj izvršenih instrukcija:  i pojavit će se polje Pause kojim pauziramo izvođenje instrukcija: .

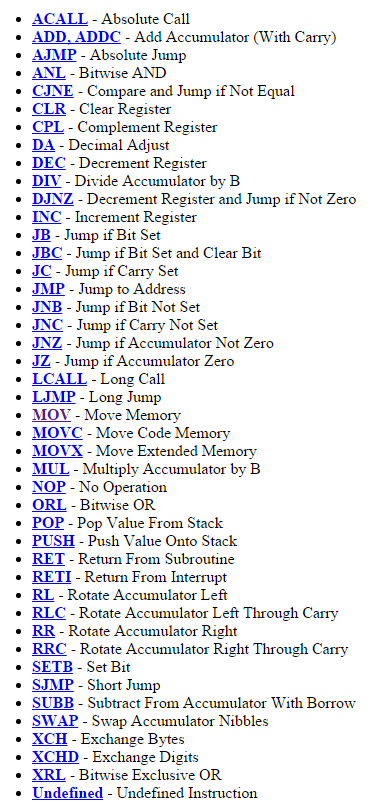
Nakon izvršavanja koda potrebno je resetirati assembler kako bi se kod izmijenio ili ponovo pokrenuo. To radimo pritikom na gumb RST.

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

## Primjer izvršavanja assembler koda

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



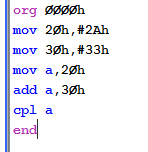


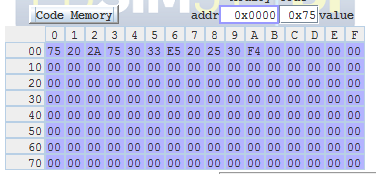
Kao primjer assembler koda uzet ću zbrajanje i množenje operanda 2Ah i 33h.

### Zbrajanje i komplement

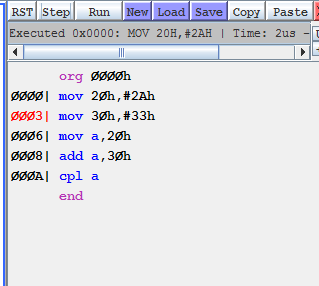
Upisati operande 2Ah na adresu 20h i 33h na adresu 30h. Operande zbrojiti, a rezultat komplementirati.

Assembler kod:



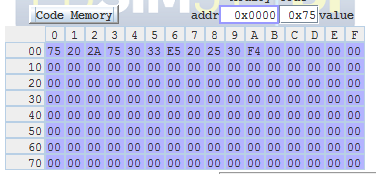


Nakon pritiska Assm u kodnu memoriju su upisane naredbe koje će procesor izvršavati. Kodna memorija počinje na adresi 0000h. To određujemo naredbom ORG.

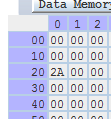


Operandi s kojima radimo (spremamo, računamo...) u kodu se pišu sa # oznakom prije vrijednosti kako se ne bi pomiješali s adresama u memoriji koje se pišu bez ikakve oznake prije broja adrese. Iza svakog broja može stajati indikator koji određuje iz kojeg je brojevnog sustava broj. Taj insikator može 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.

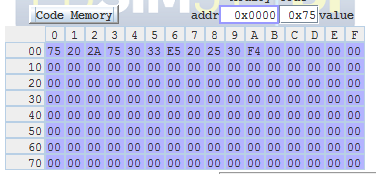
Krenuo sam izvršavati Step po Step. Svaki put nakon pritiska Step program kreće u novi red i programsko brojilo (PC) se pomiče za određeni broj. Programsko brojilo služi za pohranjivanje adrese sljedeće naredbe. Ono se najčešće inkrementira (poveća za 1) nakon svake izvršene instrukcije. Kontrolna jedinica čita adresu u programskom brojilu i dohvaća instrukciju u instrukcijski registar nakon čeka se ona izvršava, a programsko brojilo se inkrementira.  - nakon izvršavanja prve linije assembler koda. To znači da je progrmsko brojalo 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.



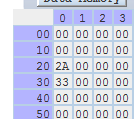
Nakon izvršavanja MOV instrukcije u memoriji na adresi 20 se pojavila vrijednost 2A.



Ponovo sam kliknu Step i kreće se izvršavati sljedeći blok naredbi od PC=0003.



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



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



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



U tražilicu sam upisao PSW kako bih mogao 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.2Ab=00101010.

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 vrijdnost s adrese 30h.

|  |  |  |
| --- | --- | --- |
| 2A | Binarno: | 00101010 |
| +33 |  | +00110011 |
| 5D |  | 01011101 |

- 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.

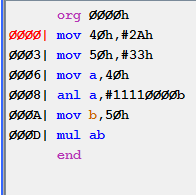


|  |
| --- |
| 01011101 – 5D |
| 10100010 – A2 |

Ovime dolazimo do naredbe END koda i kraja izvršavanja koda.

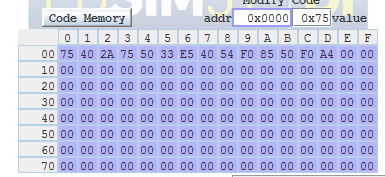
### Množenje i logičke operacije

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.



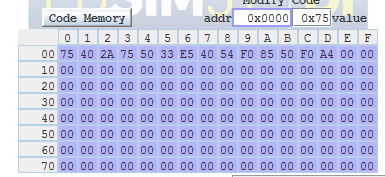
Nakon pritiska Assm kod je učitan u kodnu memoriju i izvršavanje može početi.

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



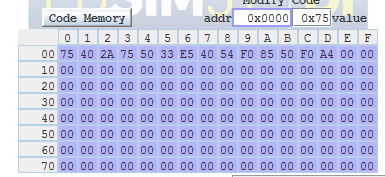
To je instrukcija MOV s kodom 75 koja prebacuje na adresu 40h vrijednost 2Ah. Nakon nje se izvršava instrukcija na adresi 0003h – opet MOV koja prebacuje na adresu 50 u memoriji vrijednost 33h.

 - Vrijednosti su prebačene u memoriju, a programsko brojilo je sada 0006h. Sljedeća naredba je na adresi 0006h, a to je MOV

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

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



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.

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 |

 Rezultat se sprema u akumulator.

PSW je opet u 1 zbog pariteta. 

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

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.



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

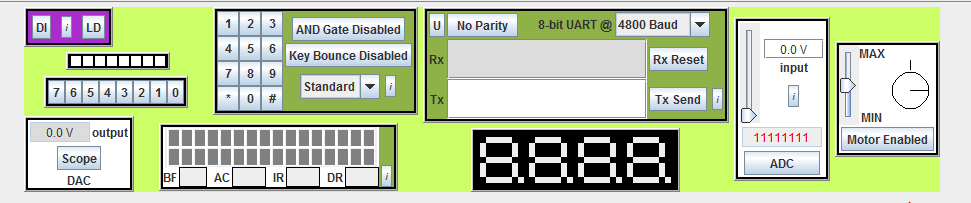
Donji bajt jednak je 60 ili 01100000b,a gornji bajt jednak je 6 ili 00000110b



PSW je ovaj put u 4 zbog zastavice OV (PSW.2). Ona označava preljevanje 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.

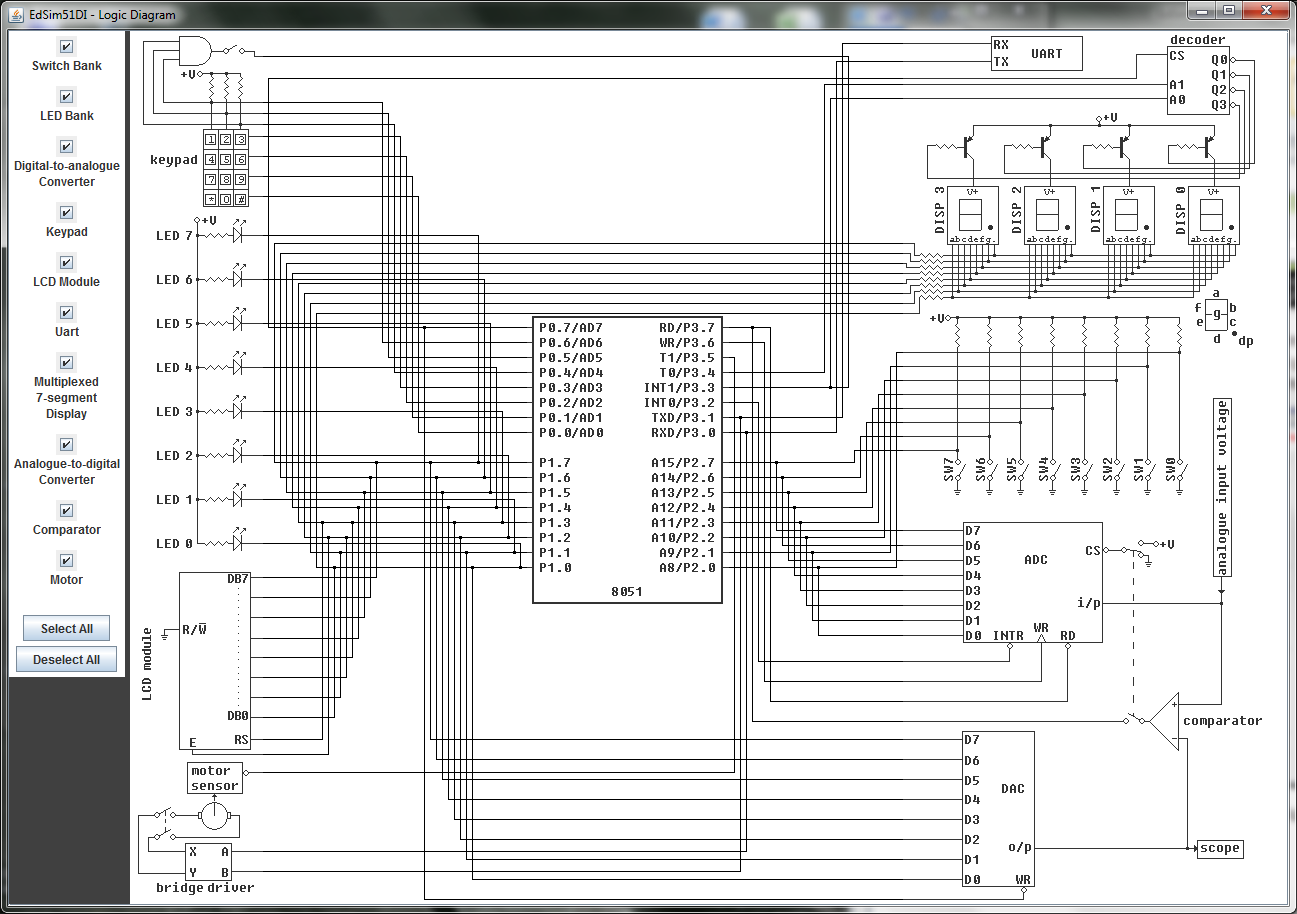
# Periferija



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



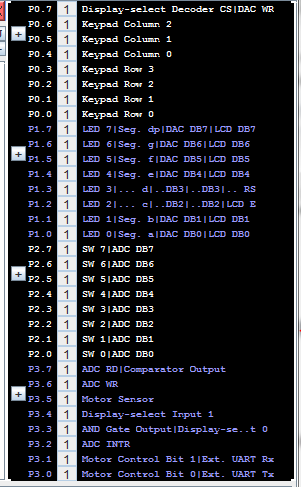
Pritiskom na LD otvara se logički diagram ulazno/izlaznog sučelja.



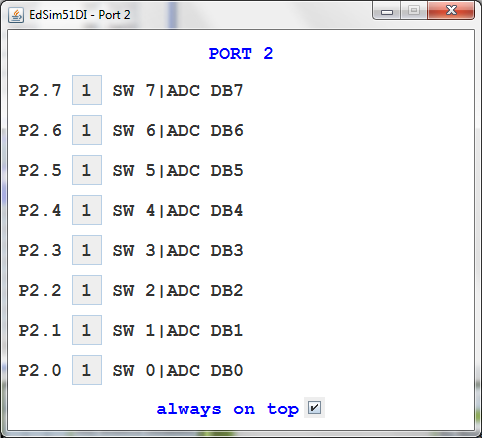
Periferne jedinice uključene u program:

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

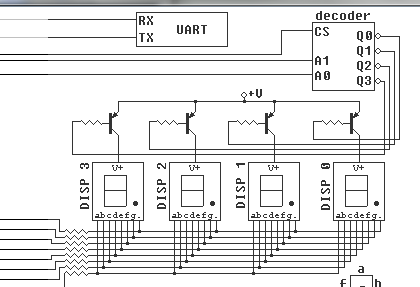
Sa lijeve strane prozora nalazi se popis portova i uređaja koji su spojeni na svaki od portova.



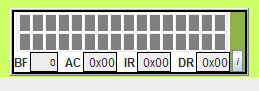
Ukoliko želimo vidjeti portove uređaja u posebnom prozoru možemo pritisnuti na +.



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 dekoder čija se vrijednost na izlazu primjenjuje kako bi odabrao jedan od 4 displaya.

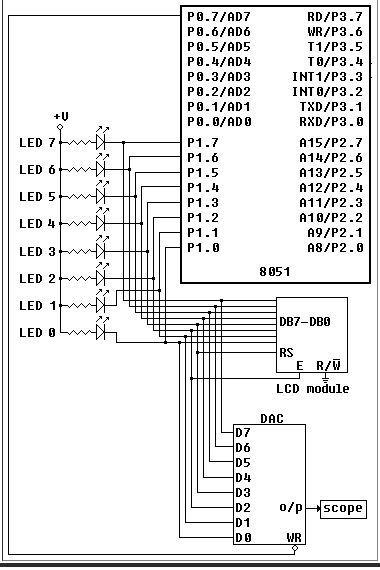


LCD modul dijeli port 1 sa LED displayem i DAC-om.



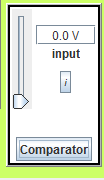
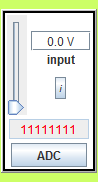
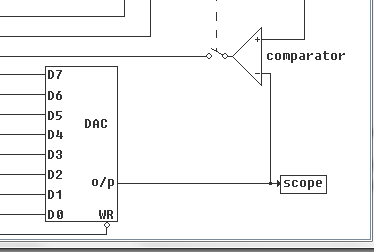
LCD modul može raditi u 4-bitnom i 8-bitnom modu.

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 an modul, a ne i ispisivati.

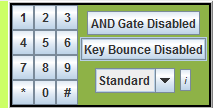
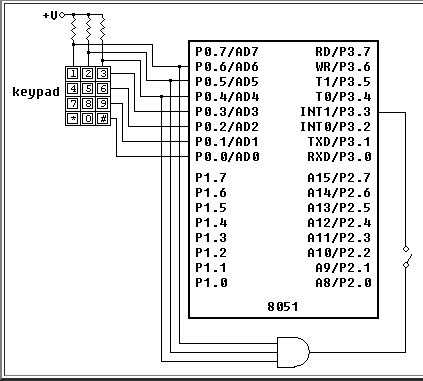


Početno je LCD modul u 4-bitnom modu. Da bi se prebacio u rad u 8-bitni mod potrebno je u DI prozoru promijeniti pinove porta.

Komparator i DAC:



Keypad:



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

[Početak](#_top)