Mikroprocesorska elektronika

Predavanje VII

Sadržaj predavanja

- Mikrokontroler Intel 8051
- Arhitektura mikrokontrolera 8051
- Organizacija memorije
- Interfejs
- Skup instrukcija
- Asembler za mikrokontroler 8051

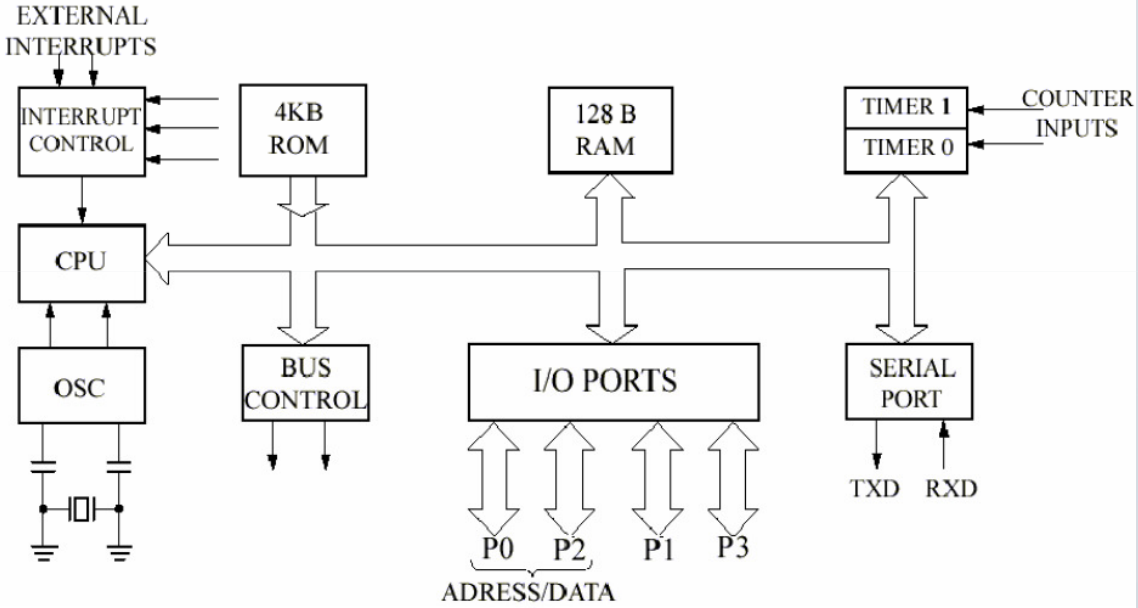
Mikrokontroler Intel 8051

Mikrokontroler Intel 8051

- Originalno razvijen od strane kompanije Intel 1981. godine
- Namenjem za korišćenje u embeded sistemima
- 8-bitni procesor
- CISC set instrukcija, instrukcije različite dužine (1 do 3 bajta)
- Harvard arhitektura
- Originalno implementiran bez protočne obrade

Osnovne karakteristike mikrokontrolera Intel 8051

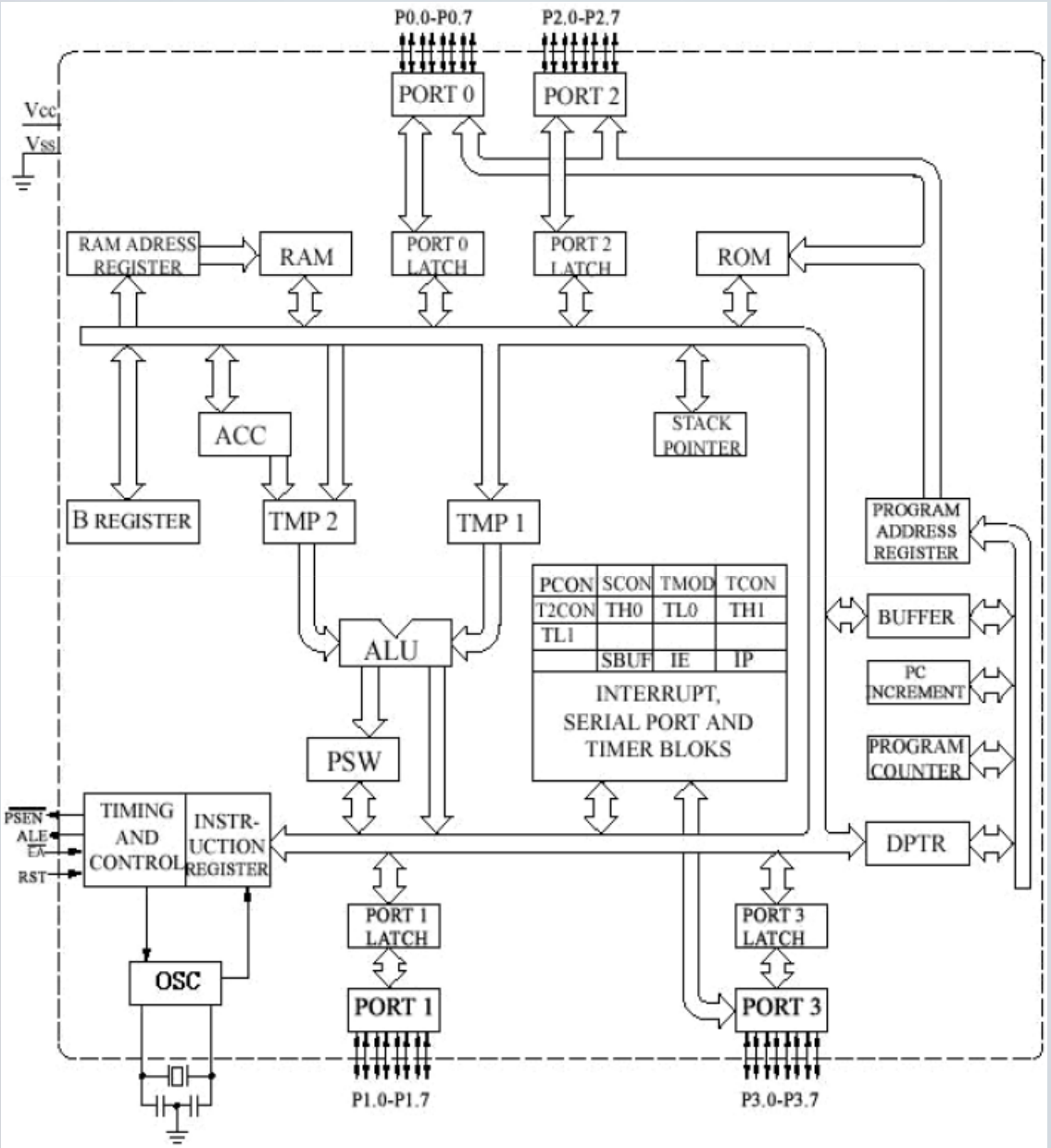
- 8-bitni CPU koji izvršava 111 instrukcija
- 4 KB (8 KB za 8052) interne ROM ili EPROM memorije za čuvanje programa
- 128 (256 za 8052) bajtova RAM memorije namenjene za upisivanje i čitanje podataka
- 64 KB adresnog prostora memorije podataka
- 64 KB adresnog prostora programske memorije
- Dva (tri za 8052) programabilna 16-bitna tajmera/brojača
- Programabilni serijski komunikacioni kontroler
- Četiri 8-bitna ulazno/izlazna priključka (portovi P0, P1, P2 i P3)
- Tajmerski i ulazno/izlazni prekidi sa dva nivoa prioriteta



Arhitektura mikrokontrolera 8051

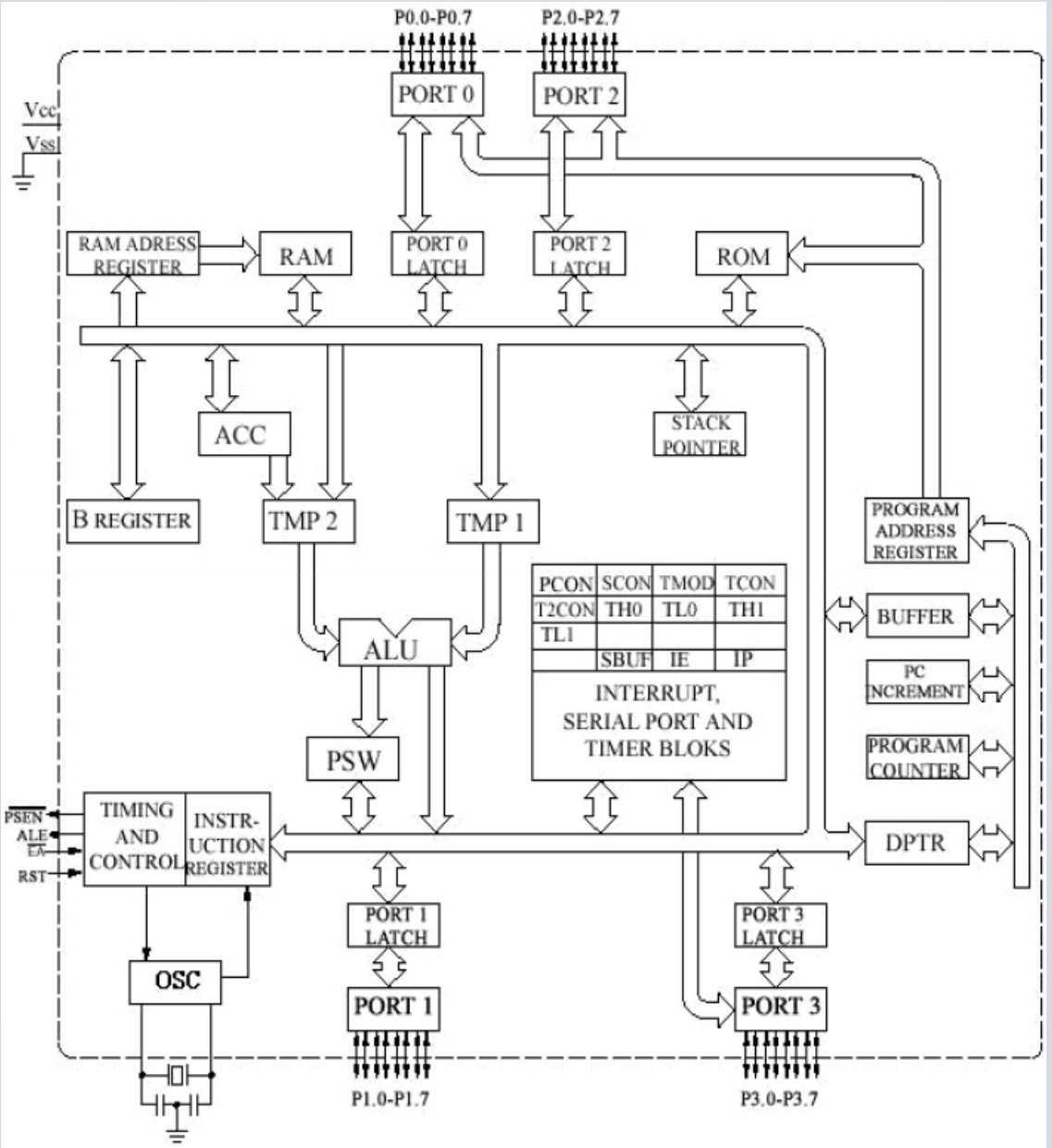
Detaljna arhitektura mikrokontrolera 8051 I

- Aritmetičko-logička (ALU) jedinica koja realizuje:
  - aritmetičke operacije sabiranja, oduzimanja, množenja i deljenja,
  - logičke operacije I, ILI, EXILI, komplement i negacija
- Par registara TMP1 i TMP2 za privremeno čuvanje operanada koji će biti korišćeni u ALU
- Akumulator (ACC)
- Pomoćni registar B – služi za smeštanje drugog operanda za aritmetičke operacije množenja i deljenja. Posle izvršene operacije množenja ili deljenja u njemu se nalazi viši bajt rezultata množenja ili ostatak deljenja, respektivno.
- Statusni registar (PSW) – 8-bitni registar sa ukupno 4 statusna bita
- Registart instrukcija (Instruction Register)
- Programski brojač (Program Counter)



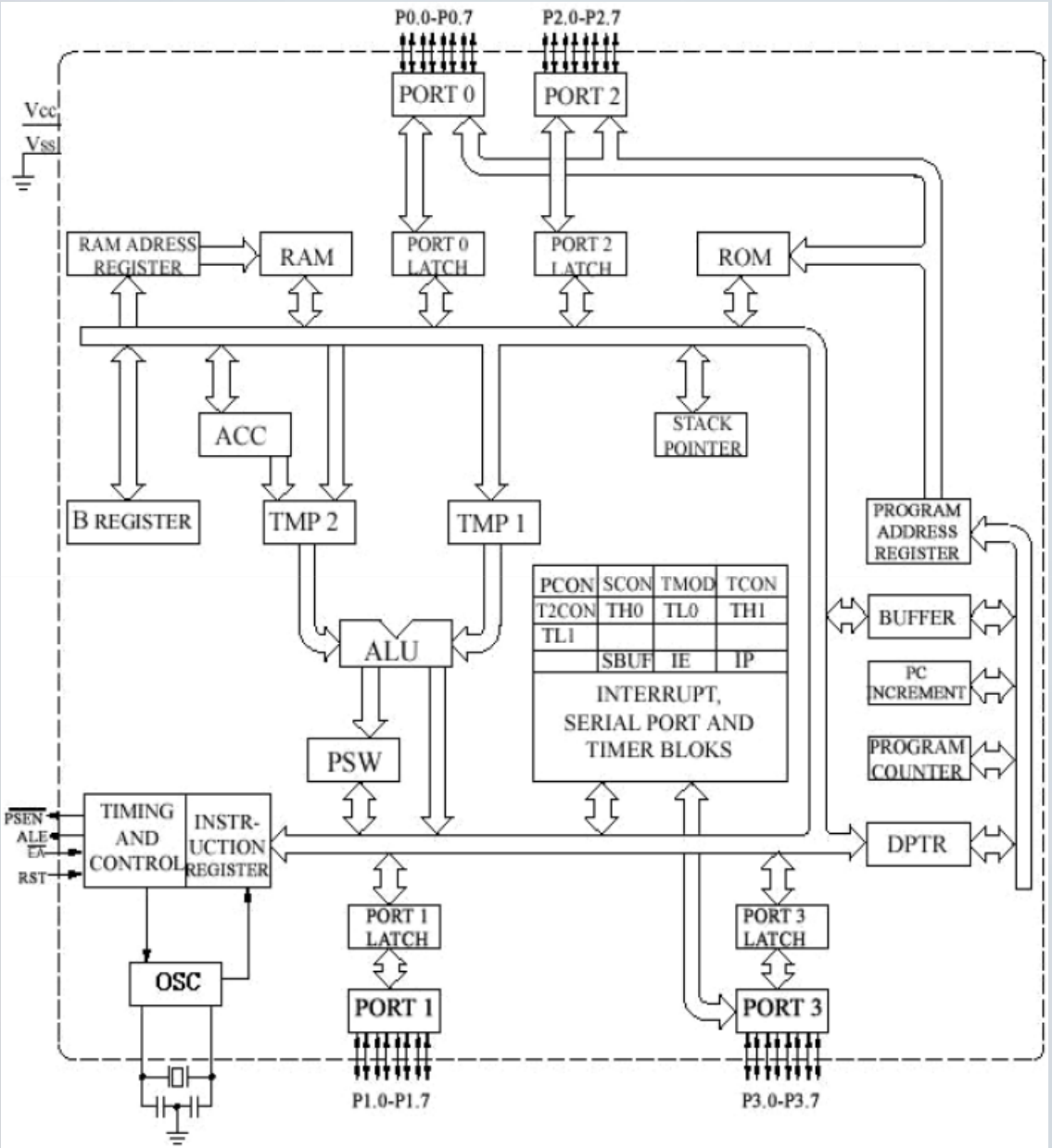
Detaljna arhitektura mikrokontrolera 8051 II

- Ukupno 8 registara opšte namene, **R0**-**R7**, fižički realizovanih unutar internog RAM-a
- Pokazivač steka (**SP**-Stack Pointer) služi za adresiranje vrha (najviše lokacije) stek memorije.
- Mikrokontroler INTEL 8051 poseduje 4 prihvatna registra (**LATCH**-a) za čuvanje stanja izlaza na portovima P0, Pl, P2 i P3.
- Registar serijskog prenosa (**SBUF**) služi za upis podatka koji se šalje i čitanje podatka koji se prima preko serijskog interfejsa.
- Registarski parovi (**TH0, TL0**) i (**TH1, TL1**) čine dva 16-bitna tajmera ili brojača.



Detaljna arhitektura mikrokontrolera 8051 III

- Za upravljanje radom prekidnog kontrolera, tajmera, brojača i za serijski kontroler koriste se registri specijalne namene **IP**, **IE**, **TMOD**, **TCON**, **SCON** i **PCON**
- **IP** služi za odrenivanje nivoa prioriteta prekida
- **IE** za maskiranje (dozvolu ili zabranu) prekida
- **TMOD** i **TCON** određuju način rada tajmera i brojača
- **SCON** i **PCON** kontrolišu rad serijskog komunikacionog kontrolera.



Struktura statusnog registra I

7	6	5	4	3	2	1	0
C	AC	F0	RS1	RS0	OV	-	P

- **C (PSW.7)** Carry Flag (Bit prenosa). Ovo je ekstenzija (deveti bit) za sve aritmetičke operacije i instrukcije pomeranja (šiftovanja), a takone je i glavni registar za 1-bitne operacije.

- **AC (PSW.6)** Auxiliary Carry Flag (pomoćni bit prenosa), samo za BCD operacije, a odnosi se na prenos izmenu nižeg i višeg nibla (donja i gornja grupa od po 4 bita). Koristi ga uglavnom mikrokontroler preko naredbi za BCD konverziju.

- **F0 (PSW.5)** Fleg 0 stoji na raspolaganju programeru kao bit za univerzalnu upotrebu.

Struktura statusnog registra II

7	6	5	4	3	2	1	0
C	AC	F0	RS1	RS0	OV	-	P

- **RS1 i RS0 (PSW.4 i PSW.3)** Register Select, služe za izbor dela internog RAM-a u kome je smeštena tekuća grupa registara (registarska banka) **R0**-**R7**, u skladu sa sledećom tabelom:

RS1	RS0	Mesto u RAM-u
0	0	Grupa 0 00h-07h
0	1	Grupa 1 08h-0Fh
1	0	Grupa 2 10h-17h
1	1	Grupa 3 18h-1Fh

Struktura statusnog registra III

7	6	5	4	3	2	1	0
C	AC	F0	RS1	RS0	OV	-	P

- **OV (PSW.2)** Overflow (prekoračenje). Setuje se (postavlja na 1) ako je rezultat aritmetičke operacije sa predznakom takav da ne može da se prikaže u jednom bajtu, tj. done do prekoračenja opsega (na primer, sabiranjem dva pozitivna broja dobije se negativan broj ili obrnuto). Ako nema prekoračenja, ovaj bit će biti 0.

- **(PSW.1)** Rezervisano od stane proizvonača za budući razvoj čipa.

- **P (PSW.0)** Parnost. Ako je broj setovanih bita u akumulatoru paran, ovaj bit će biti postavljen na 1, a ako je neparan, biće resetovan na 0. Uglavnom se koristi za generisanje bita parnosti kod slanja bajta na serijski port ili za testiranje parnosti posle serijskog prijema.

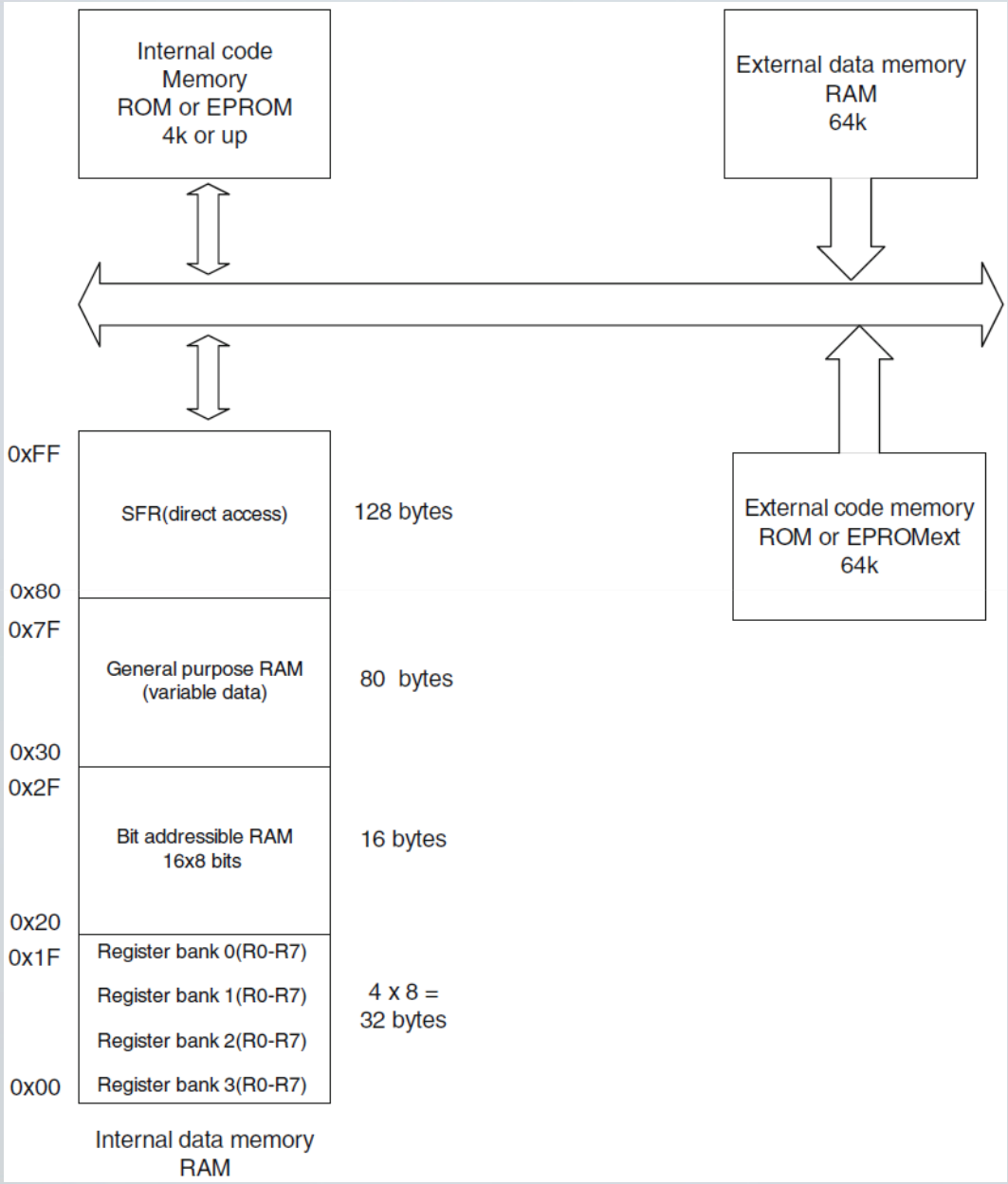
Organizacija memorije

Organizacija memorije I

- Memorijski adresni prostor mikrokontrolera INTEL 8051 je podeljen na dva osnovna dela:
  - Adresni prostor rezervisan za programe (*Code Address Space*)
  - Adresni prostor rezervisan za podatke (*Data Address Space*)

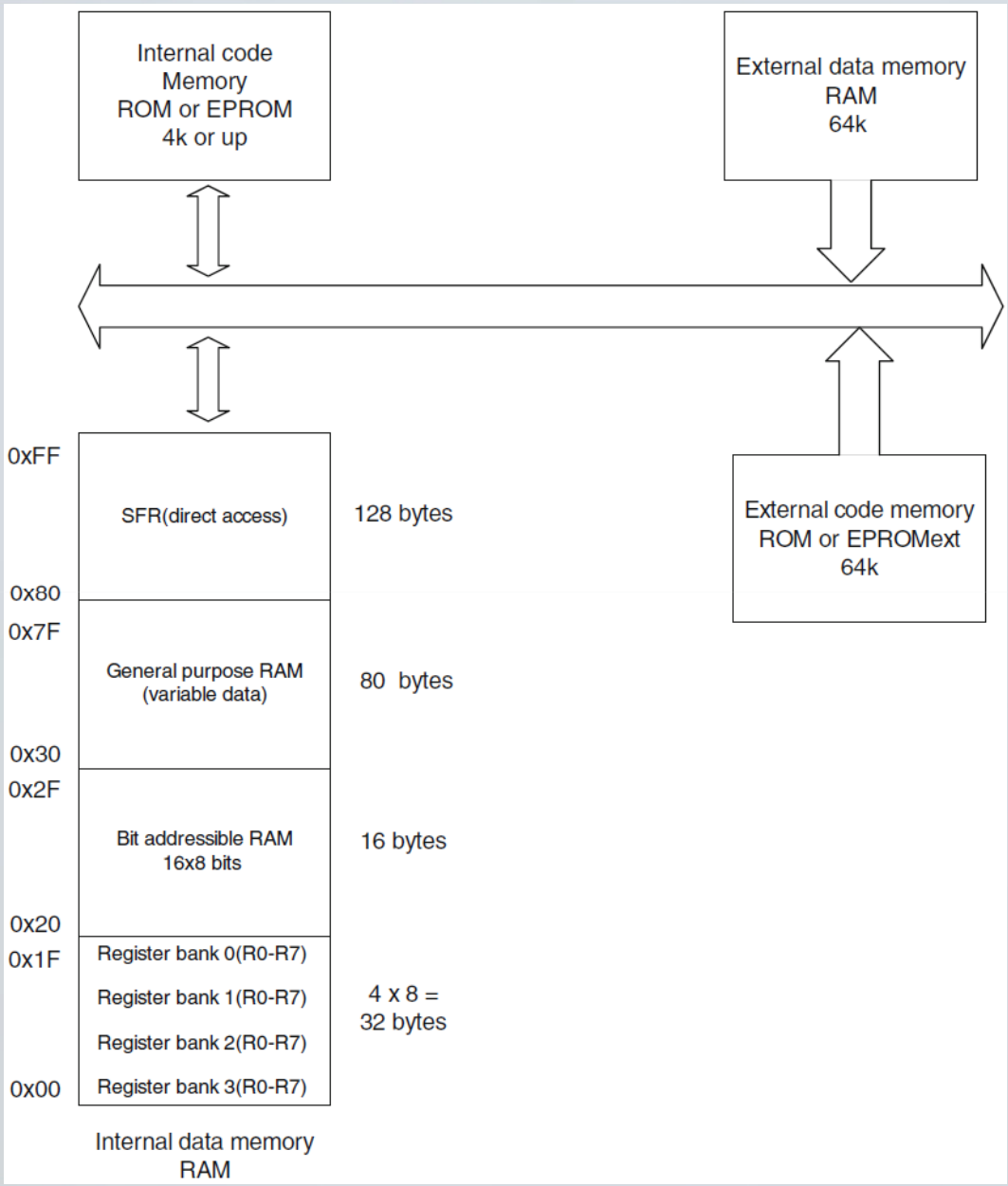
- Mikrokontroler može da adresira 64KB programske memorije; interno u samom čipu ima 4KB, a ostatak od 60KB je predviđen kao spoljašnja memorija.

- *Programska memorija* je tipa ROM (u nekim varijantama i EPROM ili FLASH), i u nju se smeštaju programi i eventualno neke konstante.



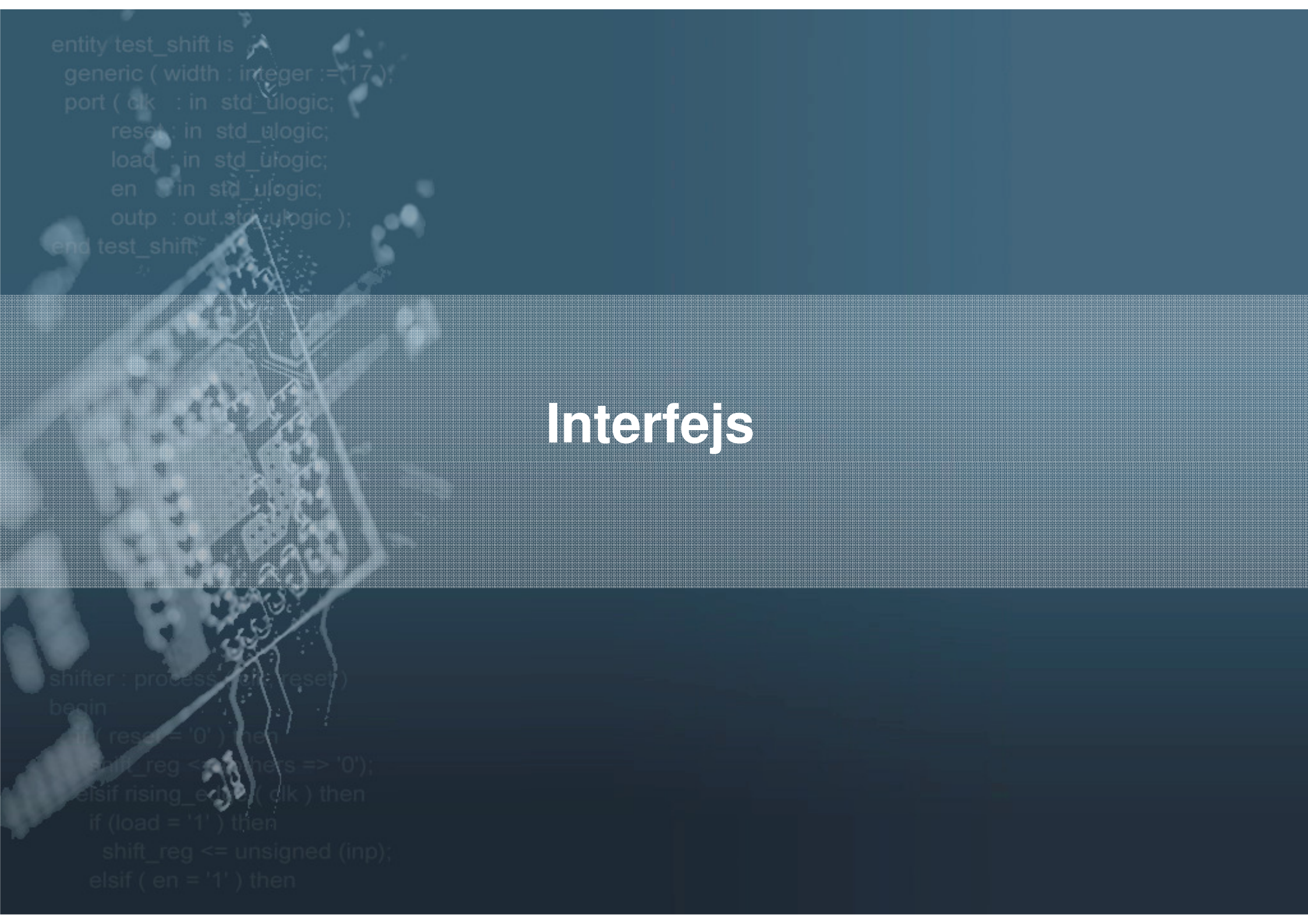
Organizacija memorije II

- *Memorija podataka* je RAM tipa, a sastoji se iz:
  - interne memorije (128/256 bajtova)
  - eksterne memorije (do 64KB).
- U okviru interne memorije, 16 bajtova moguće je adresirati svaki bit (*Bit Addressable RAM*)
- U zoni interne memorije (u opsegu adresa 128 - 255) se nalazi i blok specijalnih registara (*S*pecial *F*unction *R*egisters - *SFR*) za kontrolu periferijskih jedinica i rada mikrokontrolera.
- U okviru interne memorije podataka nalazi se i *stek*, koji se koristi se za privremeno čuvanje sadržaja brojača naredbi prilikom poziva na potprograme i za pamćenje adrese izvršenja programa u slučaju prekida
- Eksternoj RAM memoriji uvek se pristupa indirektno, preko odgovarajućih registara.



Memorijska mapa SFR zone

F8h									FFh
F0h	B								F7h
E8h									EFh
E0h	ACC								E7h
D8h	PSW								DFh
D0h									D7h
C8h	(T2CON)	(T2MOD)	(RCAP2L)	(RCAP2H)	(TL2)	(TH2)			CFh
C0h									C7h
B8h	IP								BFh
B0h	P3								B7h
A8h	IE								AFh
A0h	P2								A7h
98h	SCON	SBUF							9Fh
90h	P1								97h
88h	TCON	TMOD	TL0	TL1	TH0	TH1			8Fh
80h	P0	SP	DPL	DPH				PCON	87h

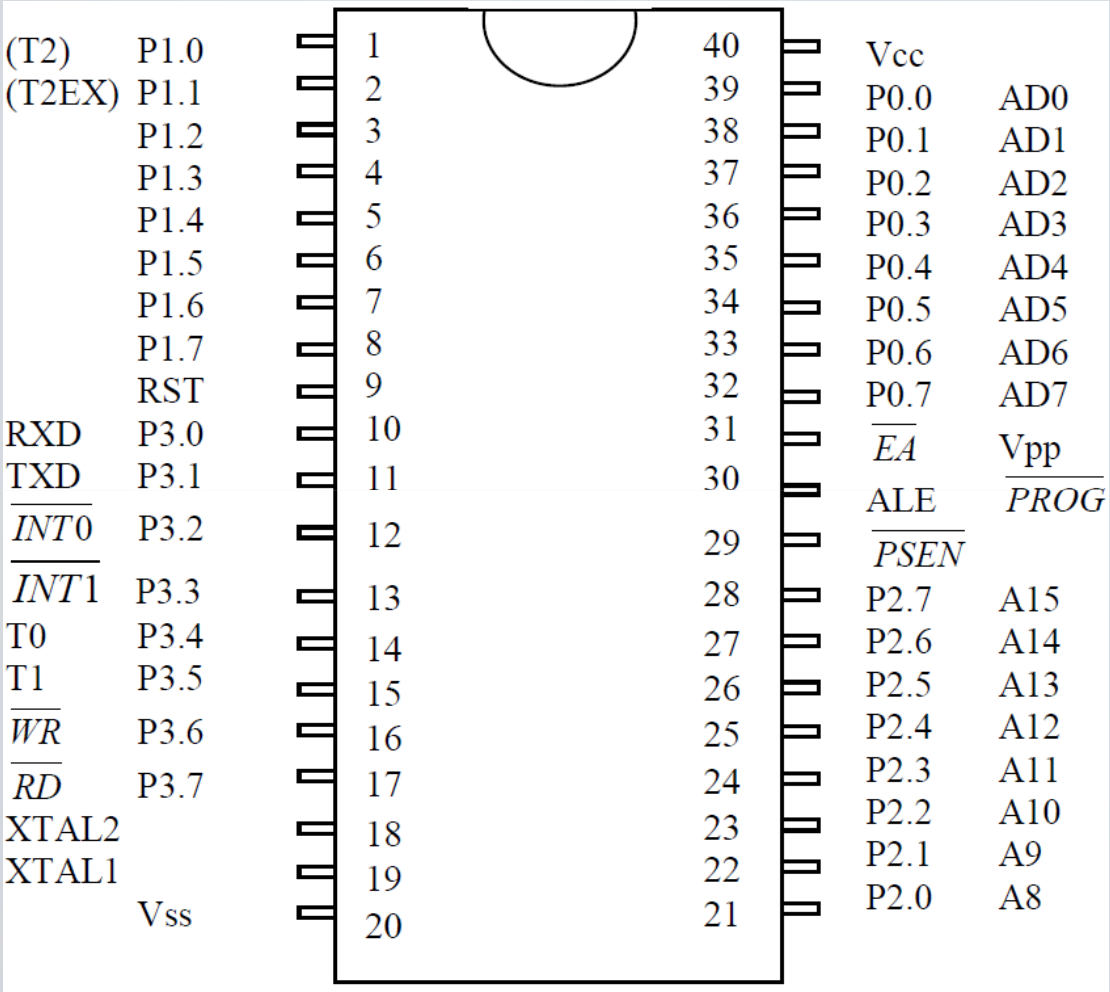

entity test_shift is
generic (width : integer := 17);
port (clk : in std_ulogic;
reset : in std_ulogic;
load : in std_ulogic;
en : in std_ulogic;
outp : out std_ulogic);
end test_shift;
shifter : process (clk, reset)
begin
if (reset = '0') then
shift_reg <= (others => '0');
elsif rising_edge (clk) then
if (load = '1') then
shift_reg <= unsigned (inp);
elsif (en = '1') then

Interfejs

Interfejs I

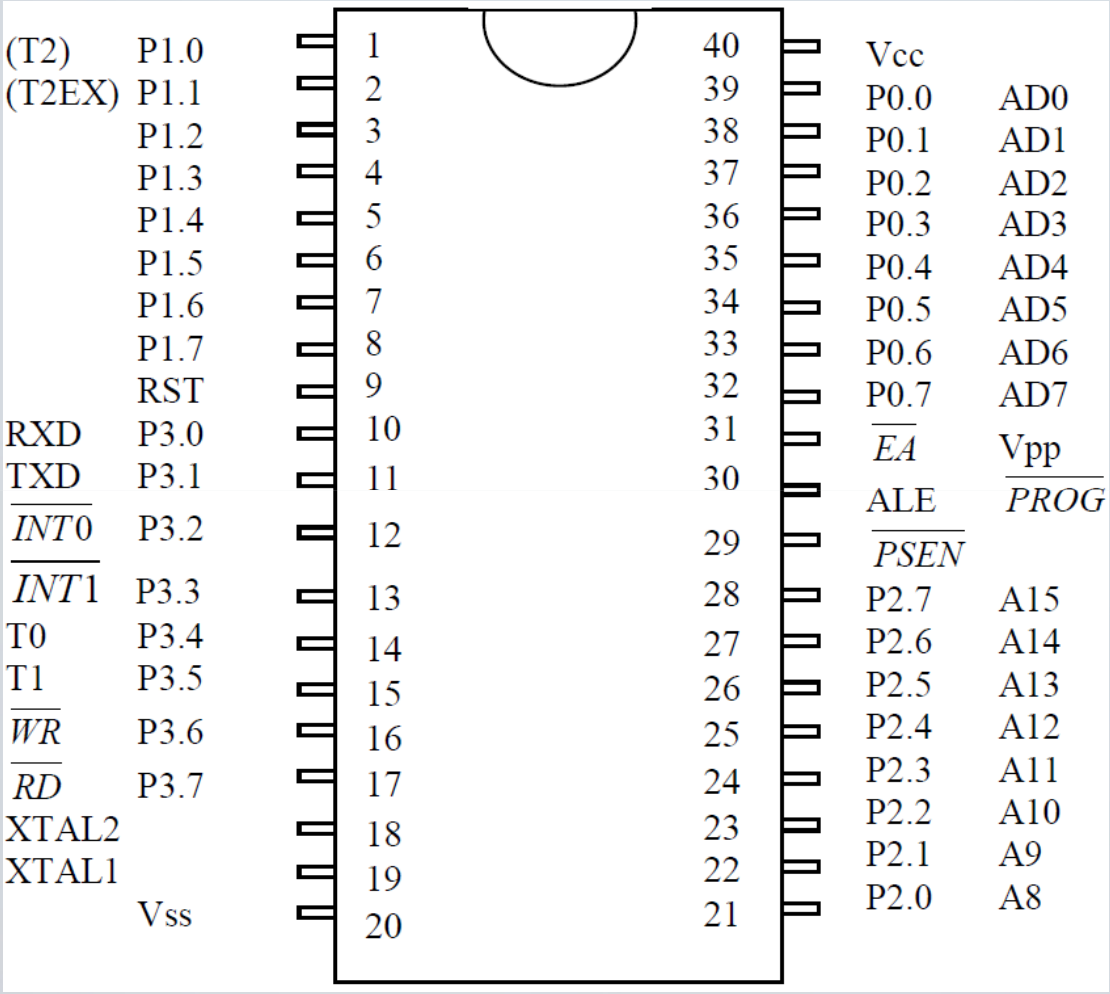
- *Nožice 1 do 8 - **Port 1**:*
  - Svaka od ovih nožica može da se koristi kao ulazni ili izlazni priključak, prema potrebi. Za 8052, **P1.0** (**T2**) je spoljašnji brojački ulaz za tajmer 2 a **P1.1** (**T2EX**) je spoljašnja kontrola (triger) za tajmer 2.

- *Nožica 9 - **Reset**:*
  - Visok logički nivo na ovom ulazu resetuje sve interne registre (registre dovodi u stanje 00000000), sa sledećim izuzecima:
    - **P0**, **P1**, **P2** i **P3** (izlazni registri svih spoljnih portova) se dovode u stanje 11111111
    - **SBUF** se ne menja
    - **SP** se dovodi u stanje 00000111 (07h)
    - Neki biti u registrima **IP**, **IE** i **PCON** fizički ne postoje, pa tako ne mogu ni da se resetuju
    - Sadržina celog internog RAM-a se ne menja
  - Najvažnija posledica aktiviranja **Reset** ulaza je da se **PC** (Program Counter) resetuje, tako da će započeti izvršavanje programa od adrese 0000h..



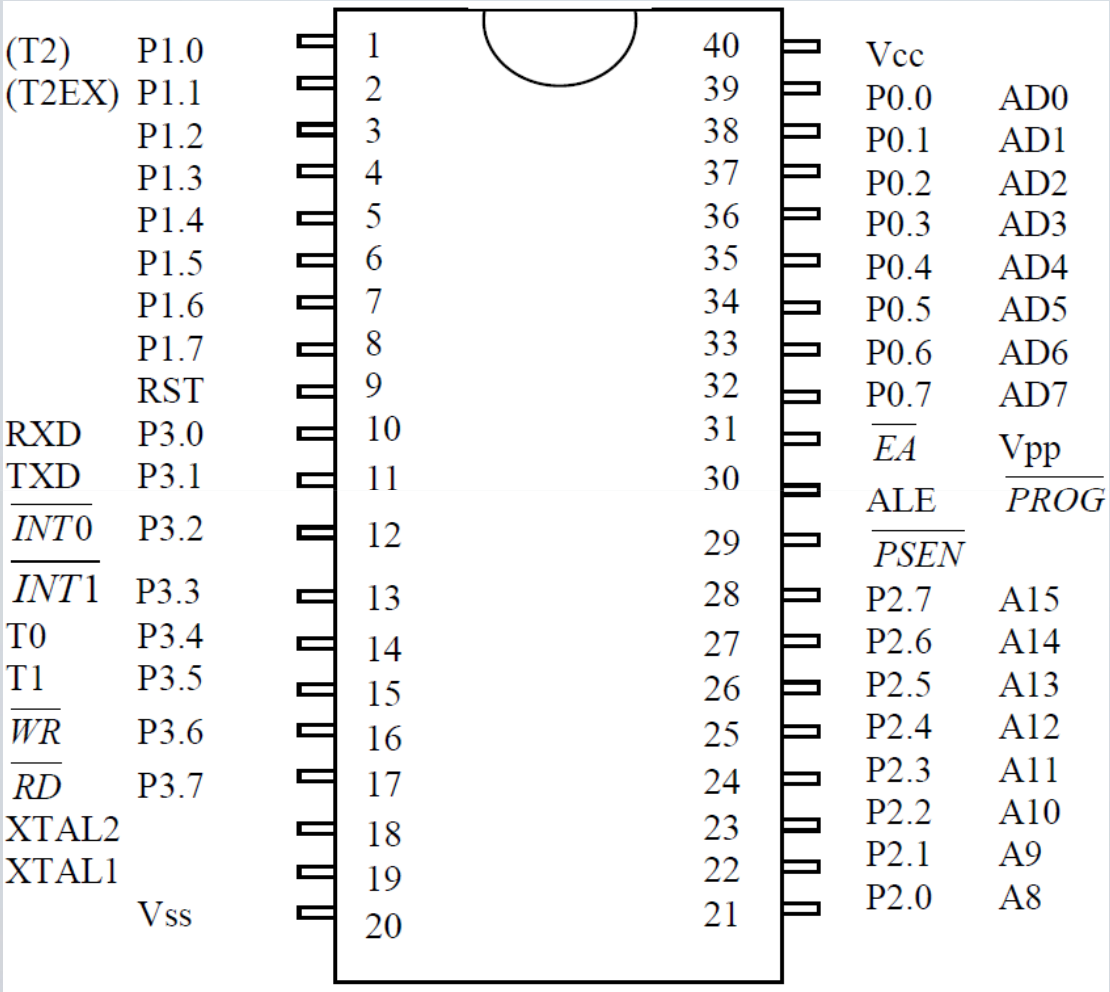
Interfejs II

- *Nožice 10 do 17 - **Port 3**:*
  - Ako se koristi kao univerzalni ulaz ili izlaz, po svemu je sličan portu 1, ali na svakoj nožici ima još po neku specijalnu funkciju:
    - 10 (**P3.0**) **RXD** - Serijski ulaz za asinhronu komunikaciju (mod 1, 2 i 3) ili serijski izlaz za sinhronu komunikaciju (mod 0)
    - 11 (**P3.1**) **TXD** - Serijski izlaz za asinhronu komunikaciju (mod 1, 2 i 3) ili taktni (clock) izlaz sa sinhronu komunikaciju (mod 0)
    - 12 (**P3.2**) **INT0** - Ulaz za prekid (interapt) 0
    - 13 (**P3.3**) **INT1** - Ulaz za prekid (interapt) 1
    - 14 (**P3.4**) **T0** - Ulaz spoljnjeg takta za brojač 0
    - 15 (**P3.5**) **TI** - Ulaz spoljnjeg takta za brojač 1
    - 16 (**P3.6**) **WR** - Signal za upis u spoljnu memoriju
    - 17 (**P3.7**) **RD** - Signal za čitanje iz spoljne memorije



Interfejs III

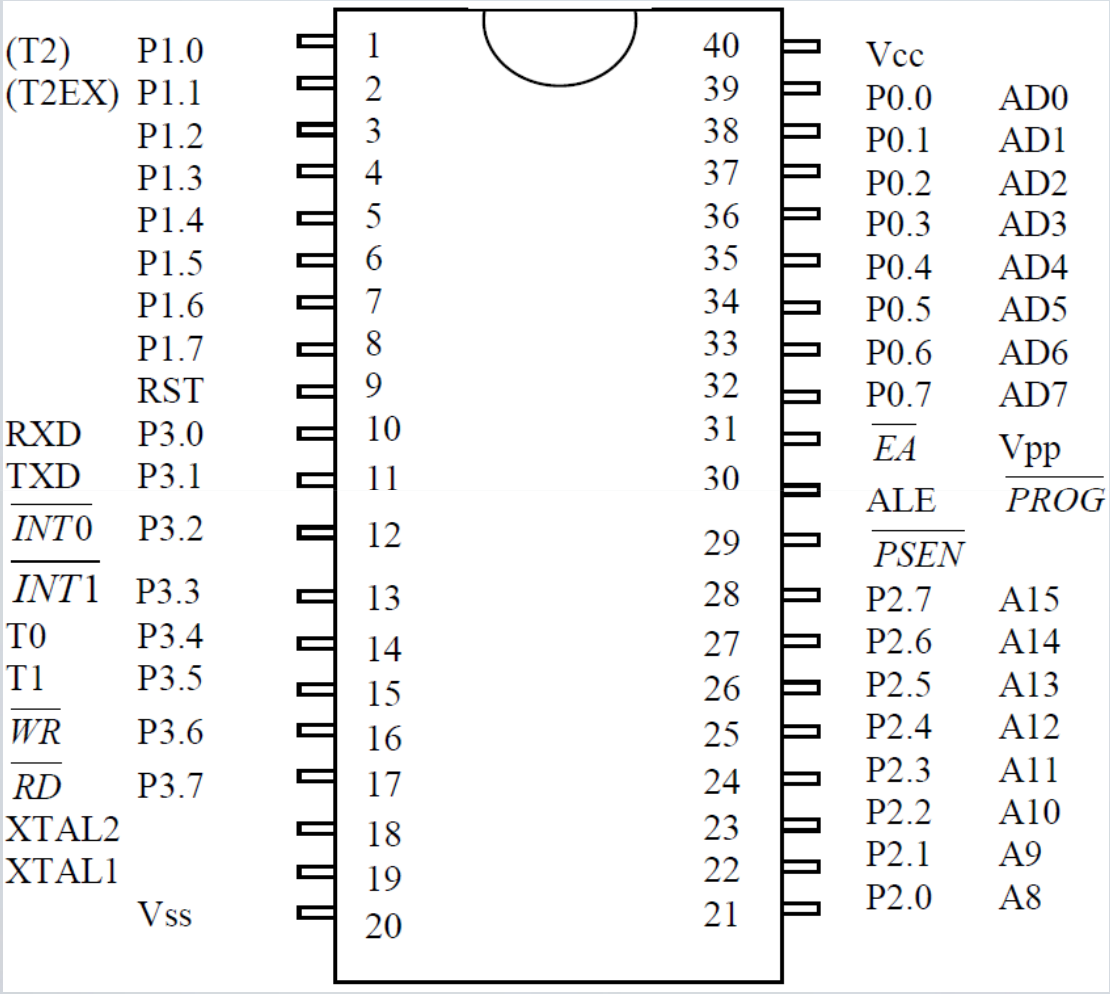
- *Nožice 18 i 19 - **X2** i **X1**:*
  - Izlaz i ulaz internog oscilatora. Ako se koristi kvarc-kristal za stabilizaciju učestanosti oscilatora (što je najčešći slučaj), on se vezuje za ove dve nožice, s tim što na svaku nožicu (prema masi) treba dodati još po jedan kondenzator od 20-40pF.
  - Ovo je potrebno da bi se sprečilo oscilovanje na nekom višem harmoniku.
  - Opseg učestanosti je od 1 do 12 MHz, a izranuju se i mikrokontroleri koji rade i na znatno višim frekvencijama.
- *Nožica 20 – Masa*
- *Nožice 21 do 28 - **Port 2** ili adrese **A8** do **A15**:*
  - Ako se koristi mikrokontroler sa internim ROM-om i nema spoljnjeg ROM-a ili RAM-a, mogu se koristiti sve linije ovog porta kao univerzalni ulazi ili izlazi.
  - Ako se koristi spoljna memorija, onda su ovo visoki adresni izlazi, od **A8** do **A15**. U tom slučaju, čak i ako se koriste samo neke adrese, preostale nožice ovog porta ne mogu da se koriste kao ulazi ili izlazi.



Interfejs IV

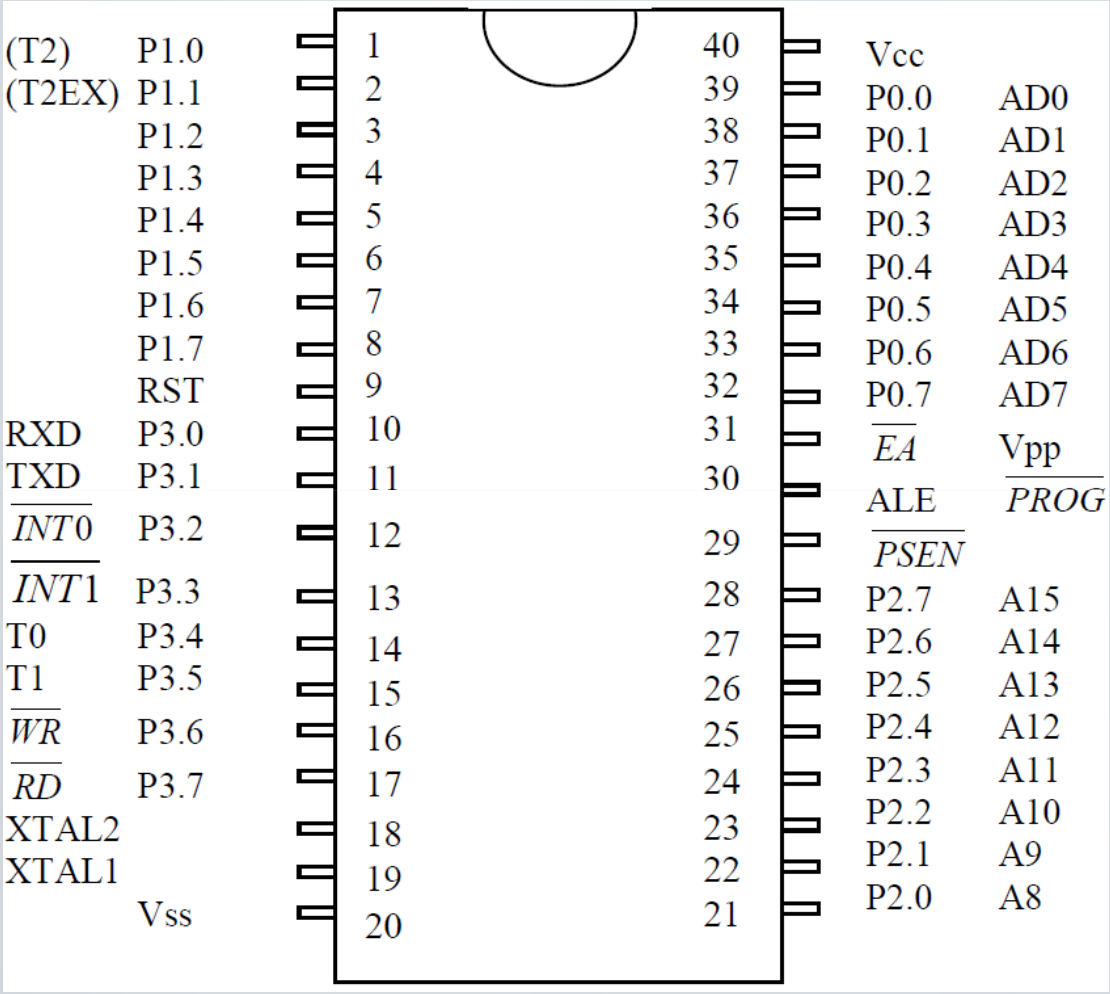
- *Nožica 29 - **PSEN**: Program Select Enable (aktiviranje spoljašnjeg ROM-a):*
  - Normalno se ovaj izlaz spaja sa **CS** ili **OE** ulazom na spoljnom EPROM-u, jer ga mikrokontroler aktivira (dovodi na nizak nivo) svaki put kad čita bajt iz spoljašnje programske memorije (za kontrolu spoljašnjeg RAM-a se koriste druge nožice).

- *Nožica 30 - **ALE**: Address Latch Enable (Upis u adresni registar):*
  - Da bi sve željene funkcije spakovao u standardno kućište od samo 40 nožica, Intel je morao da pribegne multipleksiranju nekih signala.
  - Tako je port **P0** dobio dve funkcije: izlazne adrese **A0**-**A7** i ulaz/izlaz podataka **D0**-**D7**.
  - Pre svakog očitavanja programa iz spoljne memorije ili prozivanja RAM-a mikrokontroler na **P0** proslenuje niži bajt adresnog registra i aktivira izlaz **ALE**.
  - Spoljni registar na visok nivo **ALE** memoriše stanje **P0**, a izlazi ovog registra se koriste kao **A0**-**A7**. U drugom delu mašinskog ciklusa mikrokontrolera **P0** se koristi kao magistrala podataka (*Data Bus*).



Interfejs V

- *Nožica 31 - **EA**: External Access (Pristup spoljašnjem ROM-u):*
  - Ako je ovaj izlaz nizak, mikrokontroler će sve instrukcije čitati iz spoljnog ROM-a, bez obzira da li ima interni ROM
  - Ako je ovaj izlaz visok, prvih 4 KB (8051, 8751) ili 8 KB (8052, 8752) će čitati iz internog, a sve ostalo do kraja adresnog prostora iz eksternog ROM-a.
- *Nožice 32 do 39 - **Port 0**, Adrese **A0-A7** ili magistrala podataka **D0-D7**:*
  - Slično portu **P2**, i port **P0** može da se koristi kao univerzalni ulaz i izlaz samo ako se ne koristi spoljna memorija.
  - Ako se koristi, tada je **P0** adresni izlaz za **A0**-**A7** kad je **ALE** visok, a magistrala podataka (*Data Bus, **D0-D7***) kada je **ALE** nizak.
- *Nožica 40 - Napajanje +5V*



Skup instrukcija

Skup instrukcija I

- Set instrukcija za mikrokontroler 8051 obuhvata ukupno 111 instrukcija, od kojih je:
  - 49 jednobajtnih,
  - 45 dvobajtnih i
  - 17 trobajtnih.

- Ako se uzmu u obzir i varijante istih instrukcija (na primer ista instrukcija primenjena na različite registre iz registarske banke), ukupno postoji 255 instrukcija grupisanih u sledeće grupe:
  - Instrukcije prenosa podataka
  - Aritmetičke instrukcije
  - Logičke instrukcije
  - 1-bitne instrukcije
  - Instrukcije kontrole toka izvršavanja programa

Skup instrukcija II

- Mikrokontroler 8051 podržava sledeće modove adresiranja:
  - **Direktno** – adresa operanda smeštena je u drugom bajtu instrukcije
  - **Neposredno** – vrednost operanda koju je potrebno koristiti smeštena je u samoj instrukciji
  - **Registarsko** – jedan operand smešten je u neki od R0-R7 registara
  - **Indirektno** – adresa operanda smeštena je u neki od R0-R7 registara
  - **Predekrement/postinkrement** – koristi se za rad sa stekom

8051– instrukcije prenosa podataka

Sintaksa	Opis	Adresni mod			
		DIR	IND	REG	IMM
MOV A, <src>	Prebaci u A vrednost operanda <src>	X	X	X	X
MOV <dest>, A	Prebaci sadržaj A u operand <dest>	X	X	X	
MOV <dest>, <src>	Prebaci vrednost operanda <src> u operand <dest>	X	X	X	X
MOV DPTR, #data16	Prebaci vrednost numeričke konstante #data16 u registar DPTR				X
PUSH <src>	Upiši na vrh steka vrednost operanda <src>	X			
POP <dest>	Prebaci podatak sa vrha steka u operand <dest>	X			
XCH A, <byte>	Zameni vrednosti A i operanda <byte>	X	X	X	
XCHD A, @Ri	Zameni donjih 4 bita A i memorijske lokacije čija se adresa nalazi u Ri		X		
MOVX A, @Ri	Prebaci podatak iz spoljašnje RAM memorije, sa adrese koja se nalazi u Ri, u A		X		
MOVX @Ri, A	Prebaci podatak iz A u spoljašnju RAM memoriju na adresu koja se nalazi u Ri		X		
MOVX A, @DPTR	Prebaci podatak iz spoljašnje RAM memorije, sa adrese koja se nalazi u DPTR, u A		X		
MOVX @DPTR, A	Prebaci podatak iz A u spoljašnju RAM memoriju na adresu koja se nalazi u DPTR		X		
MOVC A, @A+DPTR	Prebaci podatak iz programske memorije, sa adrese (A+DPTR), u A		X		
MOVC A, @A+PC	Prebaci podatak iz programske memorije, sa adrese (A+PC), u A		X		

8051– aritmetičke instrukcije

Sintaksa	Opis	Adresni mod			
		DIR	IND	REG	IMM
ADD A, <byte>	Saberi sadržaj A sa vrednošću operanda <byte>	X	X	X	X
ADDC A, <byte>	Saberi sadržaj A sa vrednošću operanda <byte> i vrednošću C bita iz PSW	X	X	X	X
SUBB A, <byte>	Oduzmi od A vrednost operanda <byte> i vrednost C bita iz PSW	X	X	X	X
INC A	Uvećaj vrednost A za 1				
INC <byte>	Uvećak vrednost operanda <byte> za 1	X	X	X	
INC DPTR	Uvećaj vrednost DPTR za 1				
DEC A	Umanji vrednost A za 1				
DEC <byte>	Umanji vrednost operanda <byte> za 1	X	X	X	
MUL AB	Pomnoži A i B. Donji bajt rezultata smesti u A, a gornji bajt rezultat u B.				
DIV AB	Podeli A sa B. Rezultat delenja smesti u A, a ostatak u B.				
DA A	Ažuriraj sadržaj A tako da sadrži korektan BCD kod nakon sabiranja dva BCD broja				

8051– logičke instrukcije

Sintaksa	Opis	Adresni mod			
		DIR	IND	REG	IMM
ANL A, <byte>	Bitsko logičko I između A i <byte>, rezultat se smešta u A	X	X	X	X
ANL <byte>, A	Bitsko logičko I između <byte> i A, rezultat se smešta u <byte>	X			
ANL <byte>, #data	Bitsko logičko I između <byte> i konstante #data, rezultat se smešta u <byte>	X			
ORL A, <byte>	Bitsko logičko ILI između A i <byte>, rezultat se smešta u A	X	X	X	X
ORL <byte>, A	Bitsko logičko ILI između <byte> i A, rezultat se smešta u <byte>	X			
ORL <byte>, #data	Bitsko logičko ILI između <byte> i konstante #data, rezultat se smešta u <byte>	X			
XRL A, <byte>	Bitsko logičko EX-ILI između A i <byte>, rezultat se smešta u A	X	X	X	X
XRL <byte>, A	Bitsko logičko EX-ILI između <byte> i A, rezultat se smešta u <byte>	X			
XRL <byte>, #data	Bitsko logičko EX-ILI između <byte> i konstante #data, rezultat se smešta u <byte>	X			
CLR A	Obriši sadržaj A (postavi na vrednost 0x00)				
CPL A	Komplementiraj sadržaj A (bitsko logičko NE nad sadržajem A)				
RL A	Rotiraj sadržaj A za jedno mesto u levo, sadržaj MSB bita se gubi				
RLC A	Rotiraj sadržaj A za jedno mesto u levo, sadržaj MSB bita odlazi u C bit iz PWS				
RR A	Rotiraj sadržaj A za jedno mesto u desno, sadržaj LSB bita se gubi				
RRC A	Rotiraj sadržaj A za jedno mesto u desno, sadržaj LSB bita odlazi u C bit iz PWS				
SWAP A	Zameni gornji i donji nibl unutar A				

8051– 1-bitne instrukcije

Sintaksa	Opis	Adresni mod			
		DIR	IND	REG	IMM
ANL C, bit	Logičko I između C bita PSW registra i bita **bit**				
ANL C, /bit	Logičko I između C bita PSW registra i negirane vrednosti bita **bit**				
ORL C, bit	Logičko ILI između C bita PSW registra i bita **bit**				
ORL C, /bit	Logičko ILI između C bita PSW registra i negirane vrednosti bita **bit**				
MOV C, bit	Prebaci vrednost bita **bit** u C bit PSW registra				
MOV bit, C	Prebaci vrednosti C bita iz PSW registra u **bit**				
CLR C	Postavi vrednost C bita iz PSW registra na 0				
CLR bit	Postavi vrednost bita **bit** na 0				
SETB C	Postavi vrednost C bita iz PSW registra na 1				
SETB bit	Postavi vrednost bita **bit** na 1				
CPL C	Komplementiraj vrednost C bita iz PSW registra (Logička negacija bita C)				
CPL bit	Komplementiraj vrednost bita **bit** (Logička negacija bita **bit**)				
JC rel	Prenesi tok izvršavanja programa na adresu **rel** ako je bit C=1				
JNC rel	Prenesi tok izvršavanja programa na adresu **rel** ako je bit C=0				
JB bit, rel	Prenesi tok izvršavanja programa na adresu **rel** ako je **bit**=1				
JNB bit, rel	Prenesi tok izvršavanja programa na adresu **rel** ako je **bit**=0				
JBC bit, rel	Prenesi tok izvršavanja programa na adresu **rel** i postavi **bit** na 0 ako je **bit**=1				

8051–instrukcije kontrole toka izvršavanja programa I

Sintaksa	Opis	Adresni mod			
		DIR	IND	REG	IMM
AJMP addr	Prenesi tok izvršavanja na adresu **addr**. Dozvoljen je skok na adrese koje se nalaze unutar istog 2KB programskog segmenta.				
LJMP addr	Prenesi tok izvršavanja na adresu **addr**. Dozvoljen je skok na bilo koju adresu u okviru programskog adresnog prostora od 64KB.				
SJMP addr	Prenesi tok izvršavanja na adresu **addr**. Adresa skoka mora da bude u opsegu od -128 do 127 od trenutne adrese. Adresa skoka ne mora se nalaziti u okviru istog 2K segmenta.				
JMP @A+DPTR	Prenesi tok izvršavanja na adresu **A+DPTR**				
ACALL addr	Pozovi podprogram čija je prva instrukcija smeštena na adresu **addr**. Početna adresa podprograma mora se nalaziti u istom 2K programskom segmentu u kojem se nalazi i adresa poziva.				
LCALL addr	Pozovi podprogram čija je prva instrukcija smeštena na adresu **addr**. Početna adresa podprograma može se nalaziti bilo gde unutar 64 KB programskog adresnog prostora.				
RET	Povratak iz podprograma. Prenesi tok izvršavanja programa na instrukciju koja sledi neposredno nakon instrukcije poziva podprograma.				
RETI	Povratak iz prekidnog podprograma. Prenesi tok izvršavanja programa na instrukciju koja sledi neposredno nakon instrukcije tokom čijeg izvršavanja se desio zahtev za prekidom.				
NOP	No operatio. Prenesi tok izvršavanja programa na sledeću instrukciju koja sledi neposredno nakon NOP instrukcije.				

8051–instrukcije kontrole toka izvršavanja programa II

Sintaksa	Opis	Adresni mod			
		DIR	IND	REG	IMM
JZ rel	Prenesi tok izvršavanja programa na adresu **rel** ako je A=0				
JNZ rel	Prenesi tok izvršavanja programa na adresu **rel** ako je A≠0				
DJNZ <byte>, rel	Dekrementuj sadržaj operanda <byte> i prenesi tok izvršavanja programa na adresu **rel** ako je novi sadržaj operanda <byte> različit od 0				
CJNE A, <byte>, rel	Uporedi sadržaj A i <byte> i prenesi tok izvršavanja programa na adresu **rel** ako su oni različiti				
CJNE <byte>, #data, rel	Uporedi sadržaj <byte> i numeričke konstante #data i prenesi tok izvršavanja programa na adresu **rel** ako su oni različiti				

Uticaj instrukcija na statusne bite iz PSW registra

Instrukcija	C	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	-
DIV	0	X	-
DA	X	-	-
RRC	X	-	-
RLC	X	-	-
SETB C	1	-	-
CLR C	0	-	-
CPL C	X	-	-
ANL C, bit	X	-	-
ANL C, /bit	X	-	-
ORL C, bit	X	-	-
ORL C, /bit	X	-	-
MOV C, bit	X	-	-
CJNE	X	-	-

Asembler za mikrokontroler 8051

Direktive asemblera A51 I

- **EQU** direktiva dodeljuje numeričku vrednost navedenom simbolu.
- Primer:

temperatura **EQU** 20

- Numerička vrednost koja je ovom direktivom dodeljena simbolu temperatura ne može se redefinisati u programu
- Ova direktiva se koristi da bi kod bio čitljiviji, jer kada se u programu naiđe na simbole temperatura ili brzina, zna se da taj simbol označava vrednost temperature ili brzine
- Ako bi bile napisane samo brojčane vrednosti, program bi bio znatno nerazumljiviji

Direktive asemblera A51 II

- **SET** direktiva radi isto što i **EQU** direktiva, ali se može redefinisati neograničen broj puta tokom izvršavanja programa

- Primer:

Brzina **SET** 25

…

Brzina **SET** 100

…

Brzina **SET** Brzina+10

Direktive asemblera A51 III

- **BIT** direktiva dodeljuje adresu bita navedenom simbolu

- Primer:

```
RELE BIT P1.0
LED BIT P3.4
RAMWR BIT WR
IZLAZ BIT P2.0
```

Direktive asemblera A51 IV

- **ORG** direktiva definiše tekuću vrednost brojača lokacija
- Primer:

ORG 2000

- Nakon gore navedene direktive adresa sledeće instrukcije ili komande DB (ili DW) imati vrednost 2000

Direktive asemblera A51 V

- **DS** direktiva rezerviše prostor u memoriji izražen u bajtovima

- Primer:

```
                    ORG 100 ; počni od adrese 100
                    DS 7    ; rezerviši 7 bajtova
NovaLok:                    ; nastavi program
```

- U gornjem primeru labela NovaLok će imati vrednost 107, i na nju se može skočiti ako želimo da nastavimo rad na adresi 107

Direktive asemblera A51 VI

- **DB** direktiva upisuje navedenu vrednost bajta u programsku memoriju
- Ako se navodi više vrednosti, one se odvajaju zarezima
- Ako se navodi ASCII niz, stavlja se pod jednostruke navodnike
- Primer:

DB 22,33,'Alarm',0

- **DW** je isto kao **DB**, ali se upisuje 16-bitna (dvo-bajtna) vrednost
- Za razliku od nekih drugih procesora (na primer 8086), prvo se upisuje visoki, pa niski bajt

Direktive asemblera A51 VII

- **CSEG** označava da se naredni segment odnosi na programsku memoriju.
- **XSEG** označava da se naredni segment odnosi na spoljni RAM.
- **DSEG** označava da se naredni segment odnosi na interni RAM.
- **ISEG** označava da se naredni segment odnosi na gornji deo internog RAM-a, koji se može adresirati isključivo indirektno pomoću registara R0 i R1 (lokacije 80h-FFh, interna memorija koja nije implementirana u mikrokontroleru 8051)

Primer asemblerskog programa za mikrokontroler 8051

```
; osnovne aritmeticke instrukcije
dseg
            org 20h         ; preskocimo registarske banke
rez:        ds 1            ; rezervisemo 1 bajt
            jedinica equ 1h ;definisemo konstantu
cseg
            org 0
            ajmp start
start:
            mov a, #0feh
            add a, #jedinica
            add a, #1       ; a <- 0 ; c <-1
            addc a, 0       ; a <- 1
            setb c          ; c <- 1
            subb a, #1      ; a <- ff ; c <-1
            inc a
            mov r7, #5
            mov a, r7
            mov b, #3
            mul ab          ; a <- 5*3 = f
            mov b, #2
            div ab          ; a <- div(f/2) = 7 ; b <- mod(f/2) = 1
            mov rez, a
            sjmp start      ; kod mikrokontrolera uvek imamo glavnu petlju
end
```


entity test_shift is
generic (width : integer := 7
shift_reg <= unsigned (inp);
elsif (en = '1') then