Mikroprocesorska elektronika

Predavanje III

Sadržaj predavanja

- Osnovna arhitektura mikroračunarskog sistema
- Centralna procesirajuća jedinica

Osnovna arhitektura mikroračunarskog sistema

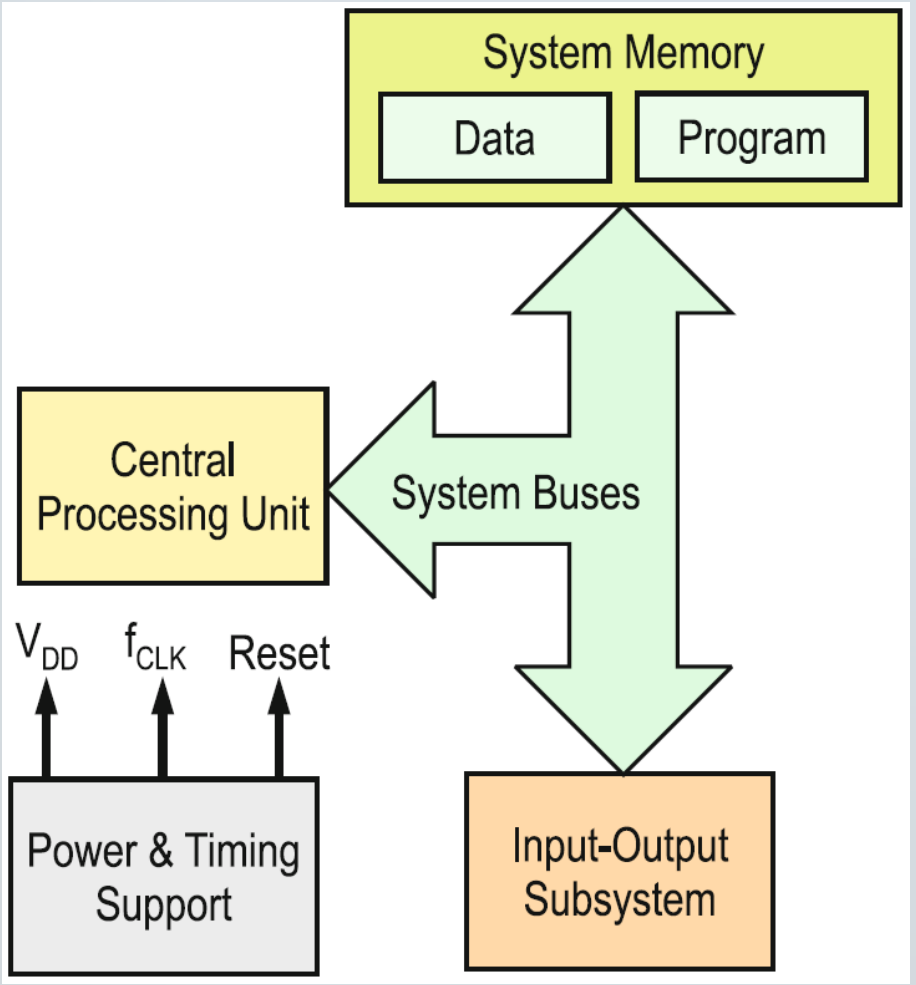
Osnovna arhitektura mikroračunarskog sistema I

- **Minimalni broj potrebnih komponenti** da bi se formirao sistem koji je u stanju da izvodi računanje čini **mikroračunar**

- **Minimalna konfiguracija mikroračunarskog sistema** sastoji se iz:
  - **Centralne procesirajuće jedinice, centralnog procesora (CPU)**
  - **Sistemske memorije**
  - **Ulazno/izlaznog interfejsa**

- Ove komponente međusobno su povezane pomoću skupa linija, grupisanih prema njihovoj funkciji, poznatih pod nazivom **sistemske magistrale**

- Dodatni broj komponenti, obezbeđuju napajanje i vremensku sinhronizaciju

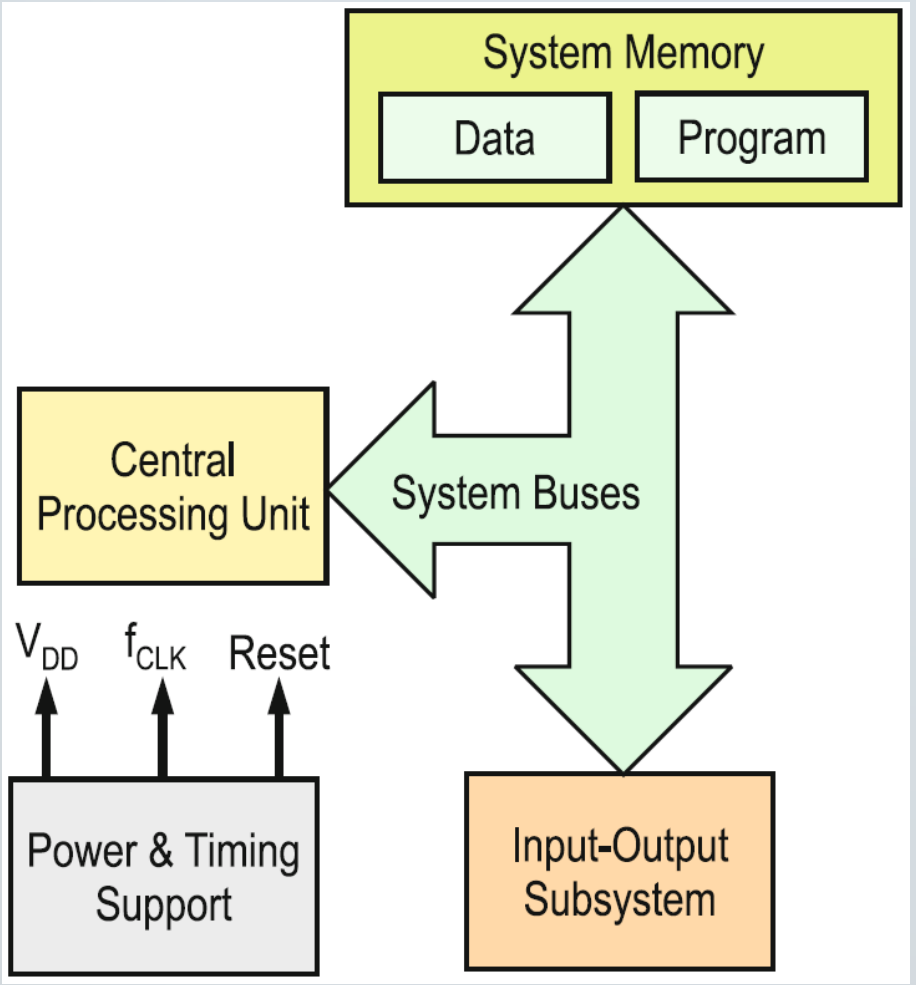


Osnovna arhitektura mikroračunarskog sistema II

- Komponente mikroračunarskog sistema mogu biti realizovane na razne načine
- Moguće ih je realizovati korišćenjem većeg broja integrisanih kola, raspoređenih na štampanoj ploči
- Mogu biti smeštene na jednom integrisanom kolu, u kom slučaju kažemo da je reč o ***mikrokontroleru***
- Danas se većina embeded sistema bazira na korišćenju mikrokontrolera

Osnovna arhitektura mikroračunarskog sistema III

- Bez obzira na način realizacije, svaka od komponenti **mikroračunarskog sistema** ima tačno definisanu funkciju:
  - **Centralna procesirajuća jedinica** – predstavlja "srce" mikroračunarskog sistema. Zadužena je za preuzimanje instrukcija iz memorije, njihovo dekodovanje i, shodno tipu instrukcije, izvođenju odgovarajućih operacija nad podacima i/ili periferijama iz ulazno/izlaznog podsistema kako bi se obezbedila željena funkcionalnost čitavog sistema
  - **Sistemska memorija** – mesto gde su smešteni programi i podaci kojima pristupa CPU. Obično postoje dva tipa memorije u sistemu: programska i memorija za podatke.
  - **Ulazno/izlazni podsistem** – sadrži sve komponente, odnosno **periferije,** koje omogućavaju da CPU razmenjuje informacije sa ostalim uređajima, sistemima i svojim okruženjem.
  - **Sistemske magistrale** – skup linija koje povezuju CPU, memoriju i ulazno/izlazni podsistem. Različite grupe linija obavljaju različite funkcije: **adresna magistrala, magistrala podataka, kontrolna magistrala**



Mikrokontroleri i mikroprocesori I

- **Mikroprocesor, MPU, u osnovi sadrži samo CPU na integrisanom kolu**

- Da bi se napravio **kompletni sistem**, sve ostale komponente (sistemska memorija, ulazno/izlazni podsistem i sistemske magistrale) moraju se implementirati **eksterno**

- Mikroprocesor tipično uključuje:

  - Visoko **optimizovanu arhitekturu** za transfer podataka od i ka sistemskoj memoriji u formi bafera i keš memorije
  - Postojanje elemenata za **ubrzavanje obrade podataka**, u obliku višestrukih funkcionalnih jedinica (fixed-point ALU, floating-point ALU, ...)
  - Sposobnost izvršavanja **više instrukcija** istovremeno
  - Sposobnost za **predikciju grananja**

Mikrokontroleri i mikroprocesori II

- Najpoznatiji sistemi bazirani na mikroprocesorima su **personalni računari (PC)**, **serveri** i **klasteri**

- Najpoznatiji proizvođači mikroprocesora su: Intel, Apple, AMD, Samsung
- Razvoj mikroprocesora prešao je dugačak put:
  - Prvi mikroprocesor kompanije Intel, 4-bitni 4004, proizveden je 1971 koristeći 10 μm tehnologiju, radio je na 400 kHz i sastojao se od 2,250 tranzistora
  - Intelov Xeon E7 mikroprocesor, 64-bitna arhitektura, proizveden 2011 koristeći 32 nm tehnologiju, radi na 2 GHz, sastoji se od $2.6 \cdot 10^9$ tranzistora
  - Intel i9 mikroprocesor, 64-bitni, 2022, 10 nm tehnologija, 5.8 GHz, do $26 \cdot 10^9$ tranzistora
  - Apple M3, 64-bitni, 2023, 3 nm tehnologija, 4.05 GHz, do $25\text{-}92 \cdot 10^9$ tranzistora
- Mikroprocesori predstavljaju **najmoćnije komponente** pomoću kojih se može **implementirati mikroračunar**

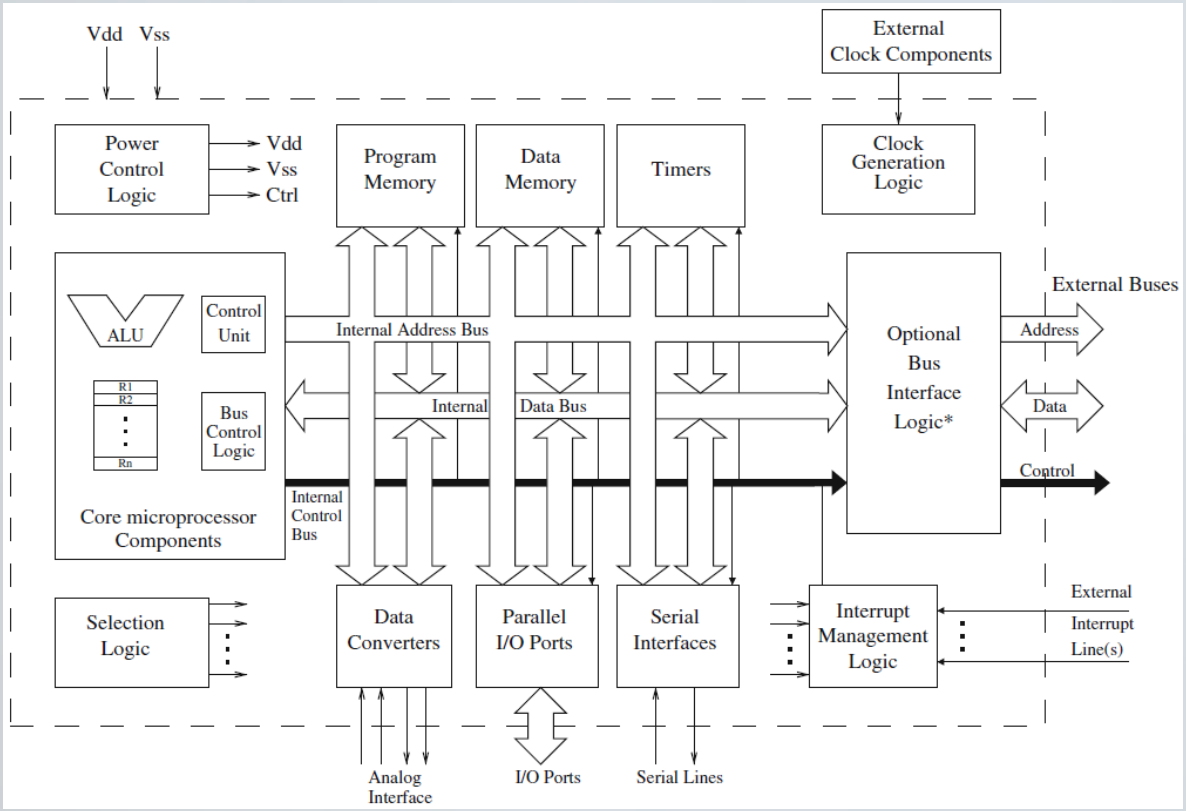
- Većina malih embeded sistema ne zahteva ovako moćne procesirajuće karakteristike

Mikrokontroleri i mikroprocesori III

- Mikrokontroler, MCU, bazira se na korišćenju **mikroprocesorskog jezgra ili centralne procesirajuće jedinice**, koje su po pravilu znatno jednostavnije od onih koje se sreću u MPU

- Ovaj osnovni CPU okružen je sa memorijom oba tipa (programska i memorija podataka) kao i sa većim ili manjim brojem periferijskih jedinica, objedinjenih na jednom integrisanom kolu

- Ovim **objedinjavanjem CPU, memorije i periferijskih jedinica u jednu, jedinstvenu celinu, nastaje mikrokontroler**



Mikrokontroleri i mikroprocesori IV

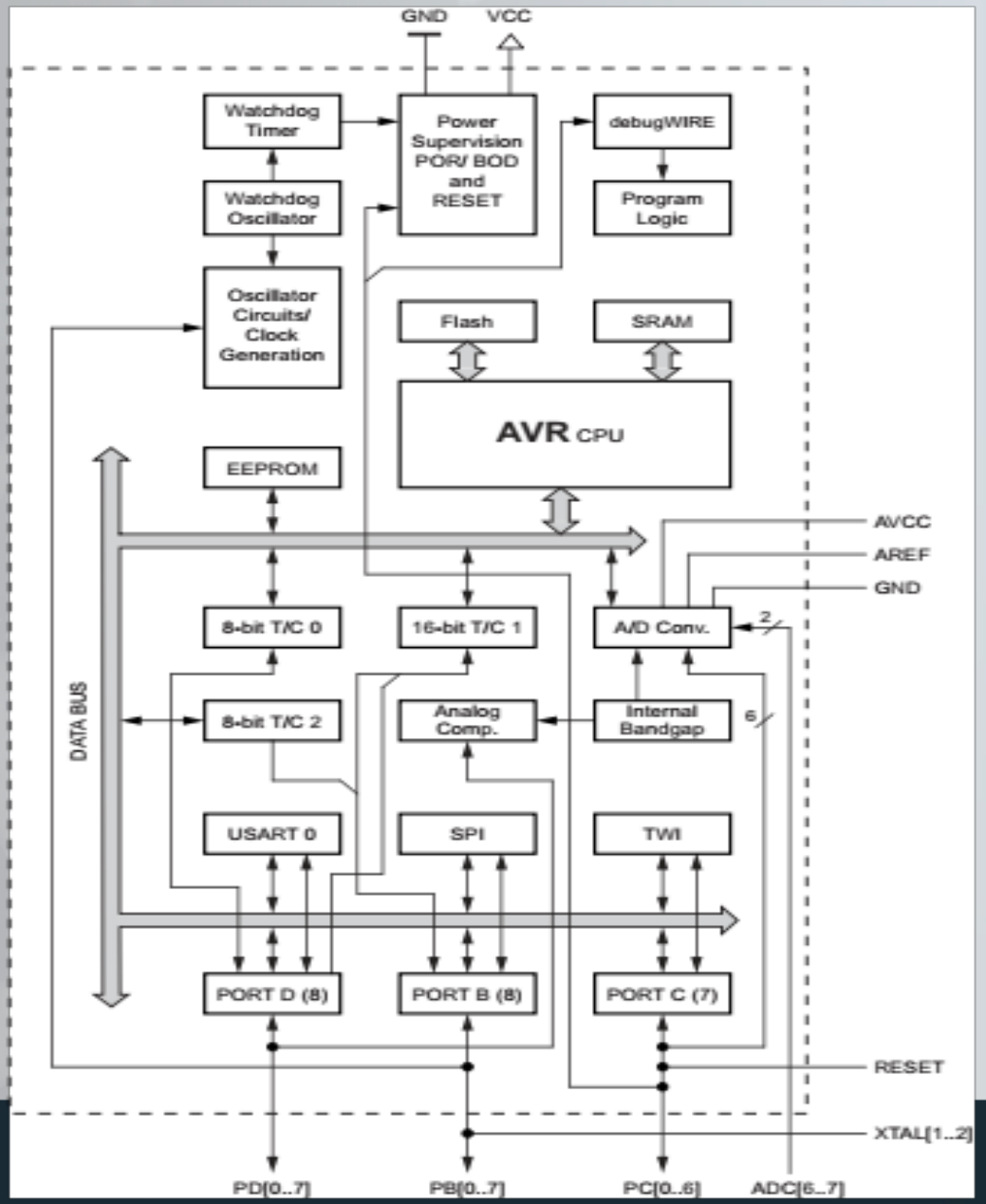
- Asortiman komponenti koje su već prisutne kao sastavni deo mikrokontrolera omogućava da se veliki broj kompletnih aplikacija realizuje samo uz pomoć mikrokontrolera, uz eventualno korišćenje minimalnog broja eksternih komponenti

- Tajmeri, brojači, kontroleri za rad sa prekidima, ulazno/izlazni portovi, A/D i D/A konvertori su najčešće **komponente** koje su prisutne unutar **mikrokontrolera**

- Mikrokontroleri se obično organizovani u **familije**. Svaka familija je organizovana oko jedne **zajedničke arhitekture** koja po pravilu obuhvata:
  - Širine programske i magistrale podataka, izražene u bitima, najčešće multiplima broja 8
  - Mikroarhitekture CPU jedinice
  - Osnovnog skupa instrukcija
  - Skupa adresirajućih modova

- Razlike između pojedinih članova familije ogledaju se u **veličini** programske i **memorije** podataka kao i u broju i vrsti periferija koje su integrisane na čip

Arhitektura mikrokontrolera ATmega328P



- Harvard arhitektura
- Flash, SRAM, EEPROM
- Direktno i indirektno adresiranje
- Dvofazna protočna obrada
- Ulazno-izlazni portovi Port B, Port B, Port D
- 3 modula za serijske komunikacione protokole USART, SPI, I^2C
- Tajmersk brojački moduli (T/C)

Centralna procesirajuća jedinica

Centralna procesirajuća jedinica I

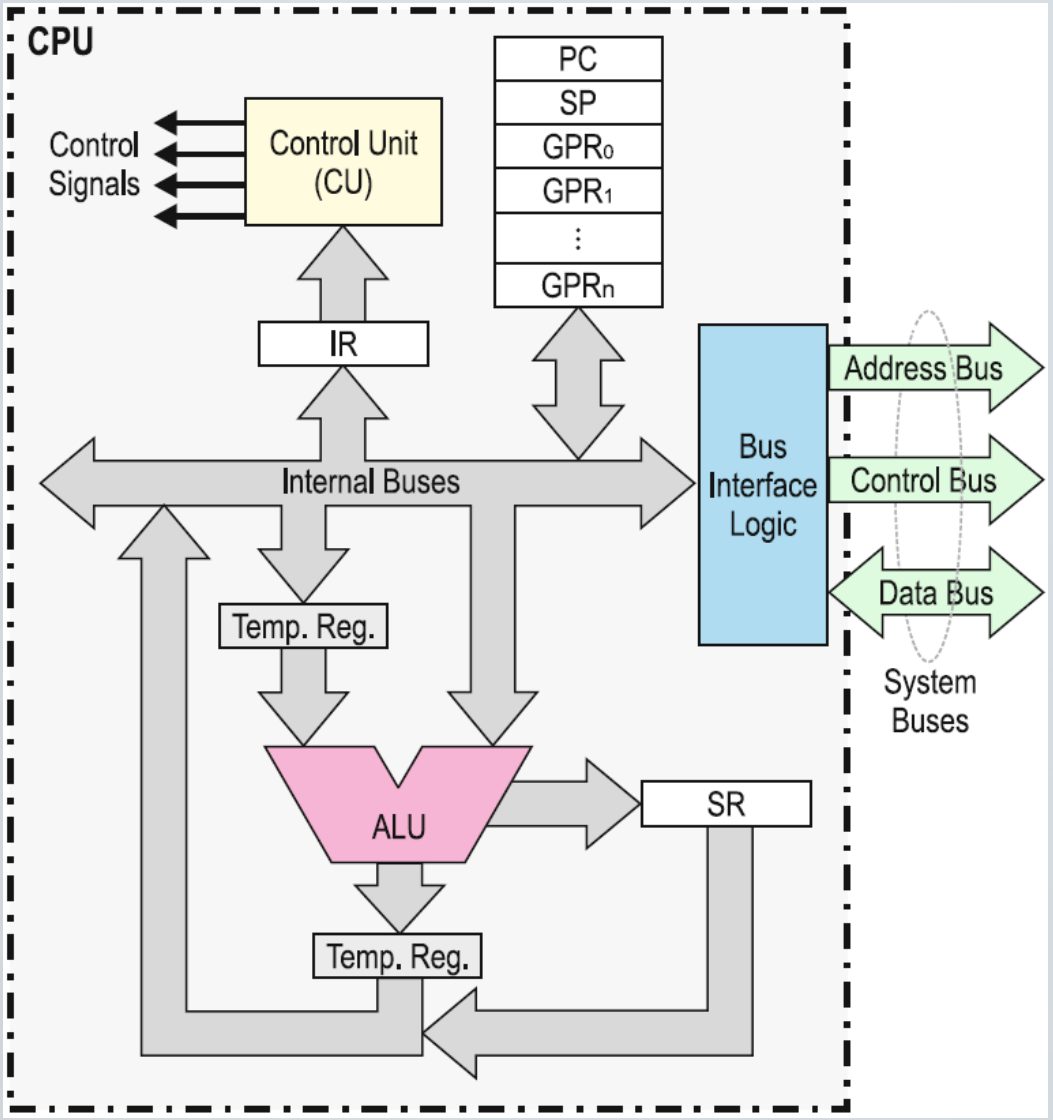
- Centralna procesirajuća jedinica (**CPU**) predstavlja **“srce” svakog embeded sistema**

- Zadužena je **za izvršavanje instrukcija programa** i u njoj se instrukcije **“transformišu”** u signale i hardverske akcije koje upravljaju radom mikroračunarskog sistema

- Minimalni skup komponenti koje definišu arhitekturu CPU uključuju:
  - **Hardverske komponente**:
    - Aritmetičko logička jedinica (ALU)
    - Upravljačka jedinica (CU)
    - Skup registara
    - Logiku za implementiranje sprežnog interfejsa (BIL)
  - **Softverske komponente**:
    - Skup instrukcija
    - Modove adresiranja

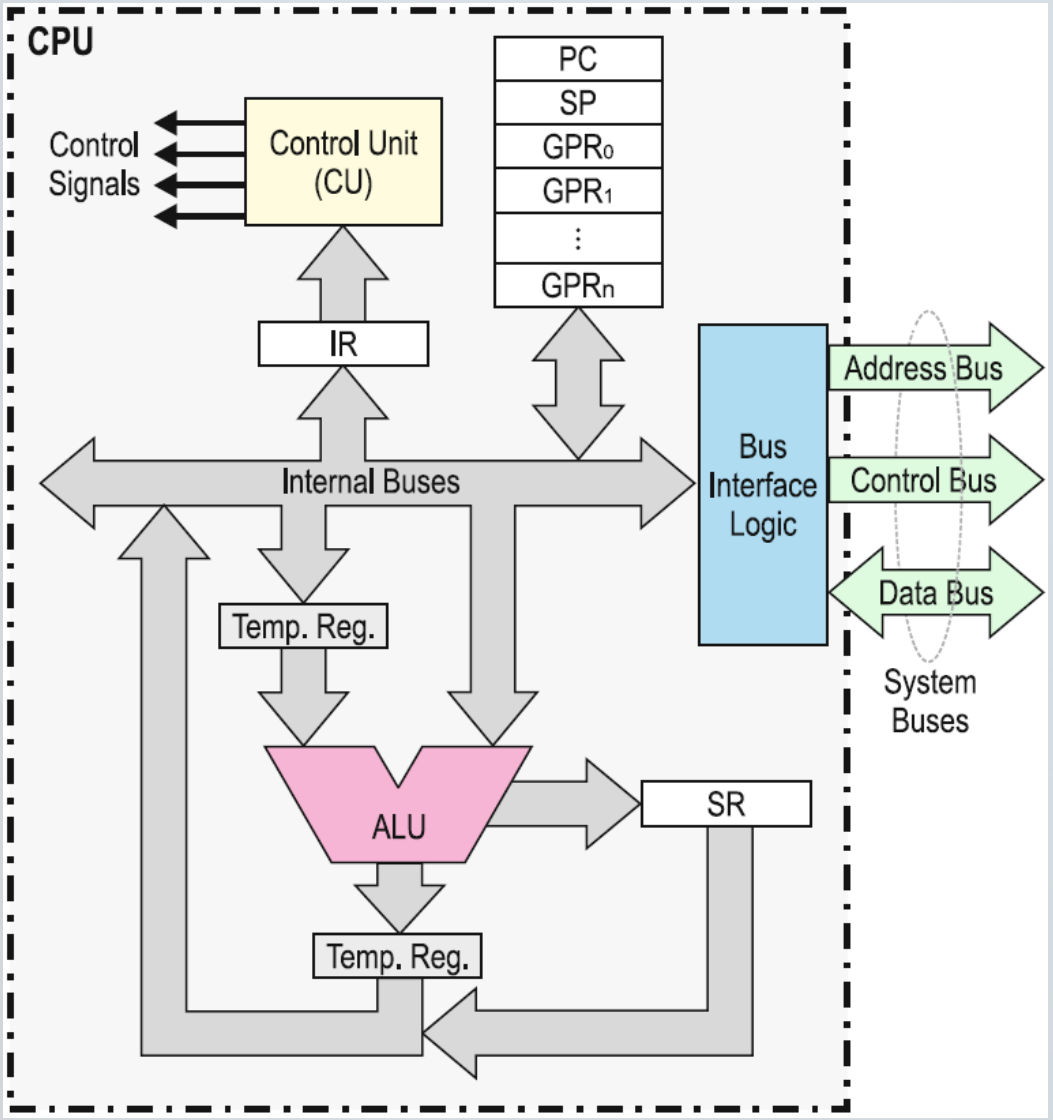
Centralna procesirajuća jedinica II

- Na slici desno prikazan je uprošćen model organizacije CPU
- Prikazane komponente omogućavaju CPU da **pristupa instrukcijama programa** i **podacima** koji se nalaze smešteni u memoriji ili u ulazno/izlaznom podsisitemu
- **Sekvenca instrukcija** koje čine program odabran je iz **skupa instrukcija posmatranog procesora**
- **Program** smešten u memoriji **diktira redosled operacija** koje je potrebno izvršiti unutar sistema
- Tokom obrade podataka, svaka od komponenti CPU ima svoju **unapred definisanu ulogu** koja je neophodna kako bi se realizovao kompletan sistem



Centralna procesirajuća jedinica III

- Skup hardverskih komponenti unutar CPU koje izvršavaju manipulaciju čini podsistem za obradu podataka (*datapath*) procesora
- **Podsistem za obradu podataka** uključuje:
  - ALU
  - Unutrašnje registre za smeštanje podataka
  - Unutrašnje magistrale podataka
  - Ostale funkcionalne jedinice, kao što su jedinice za obradu podataka predstavljenih u pokretnom zarezu, hardverske delitelje, itd.
- Hardverske komponente koje kontrolišu rad procesora čine njegov **upravljački podsistema** (*control path*)
- Upravljački podsistem uključuje:
  - Upravljačku jedinicu (CU)
  - BIL logiku
  - Komponente zadužene za vremensku sinhronizaciju



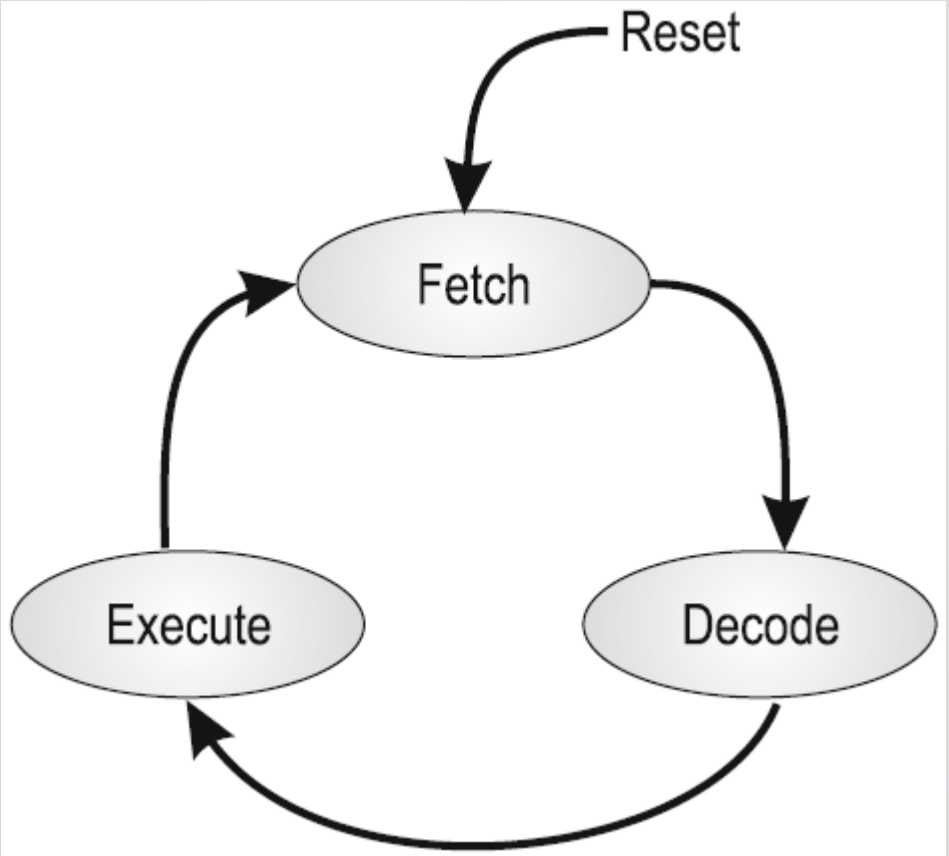
Upravljačka jedinica I

- **Upravljačka jedinica (CU)** upravlja radom CPU

- Ona je realizovana kao **konačni automat** koji neprekidno prolazi kroz sledeći skup stanja:
  1. Fazu prihvatanja instrukcije (*instruction fetch, IF*)
  2. Fazu dekodovanja instrukcije (*instruction decode, ID*)
  3. Fazu izvršavanja instrukcije (*instruction execute, IE*)

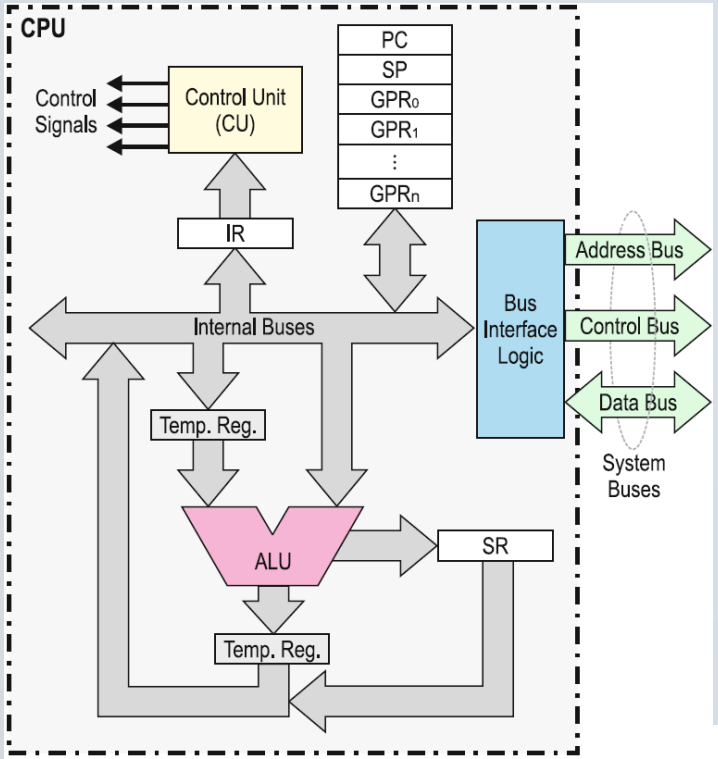
- Ovaj proces prihvatanja, dekodovanja i izvršavanja poznat je i pod nazivom ciklus instrukcije ili **CPU ciklus**

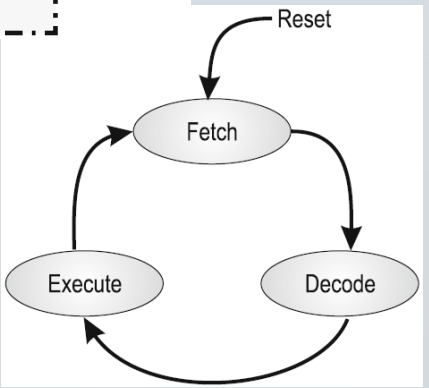
- Kompletan ciklus obično traje **nekoliko taktova**, u zavisnosti od tipa instrukcije i korišćenog načina adresiranja



Upravljačka jedinica II

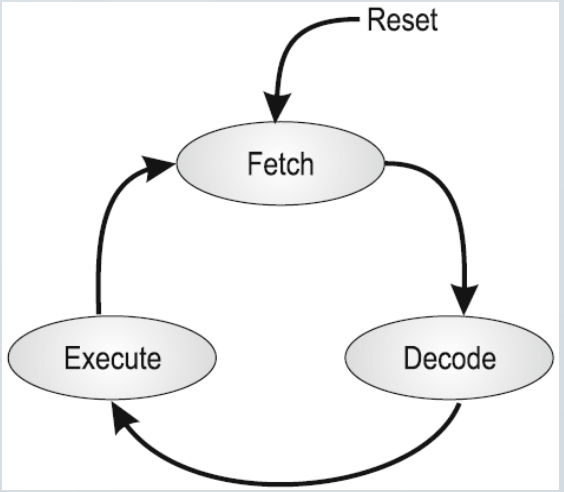
- Operacije koje se izvršavaju u svakom od stanja CPU ciklusa su:
  - **Faza prihvata**: U ovoj fazi se kod nove instrukcije iz programske memorije prenosi u CPU koristeći BIL modul. **Programski brojač (PC)** obezbeđuje adresu naredne instrukcije koju je potrebno prihvatiti iz memorije. Kod prihvaćene instrukcije smešta se u registru instrukcija (IR)
  - **Faza dekodovanja**: U ovoj fazi vrši se dekodovanje koda prihvaćene instrukcije kako bi CU mogla znati koju instrukciju treba da izvrši. Dekodovana informacija se koristi od strane CU da generiše potrebnu sekvencu upravljačkih signala kako bi se obavile akcije koje su definisane instrukcijom
  - **Faza izvršavanja**: U ovoj fazi CU aktivira neophodne CPU funkcionalne jedinice (ALU) u cilju izvršavanja akcija definisanih tekućom instrukcijom. Na kraju ove faze, vrednost PC registra se inkrementira kako bi pokazivala na adresu naredne instrukcije koju je potrebno izvršiti.





Upravljačka jedinica III

- Nakon završetka faze izvršavanja instrukcije, CU aktivira BIL modul kako bi, koristeći informaciju smeštenu u PC registru, započeo prihvat sledeće instrukcije, započinjući **novi CPU ciklus**
- CPU ciklus može zahtevati **dodatne taktove**, u zavisnosti od **adresnog moda** koji koristi instrukcija koja se trenutno izvršava
- Na primer, u fazi dekodovanja instrukcije može se pojaviti potreba za prihvatom **dodatnih podataka smeštenih u memoriji** što će zahtevati dodatne taktove
- Obzirom da je CU realizovana **kao konačni automat**, mora postojati reset ulaz kako bi se CU uvela u pravo početno stanje
- **PC registar** je takođe realizovan na takav način da se njegov sadržaj nakon reseta postavlja na vrednost adrese memorijske lokacije u kojoj je smeštena **prva instrukcija programa**
- Adresa prve instrukcije u memoriji naziva se ***reset vektor***

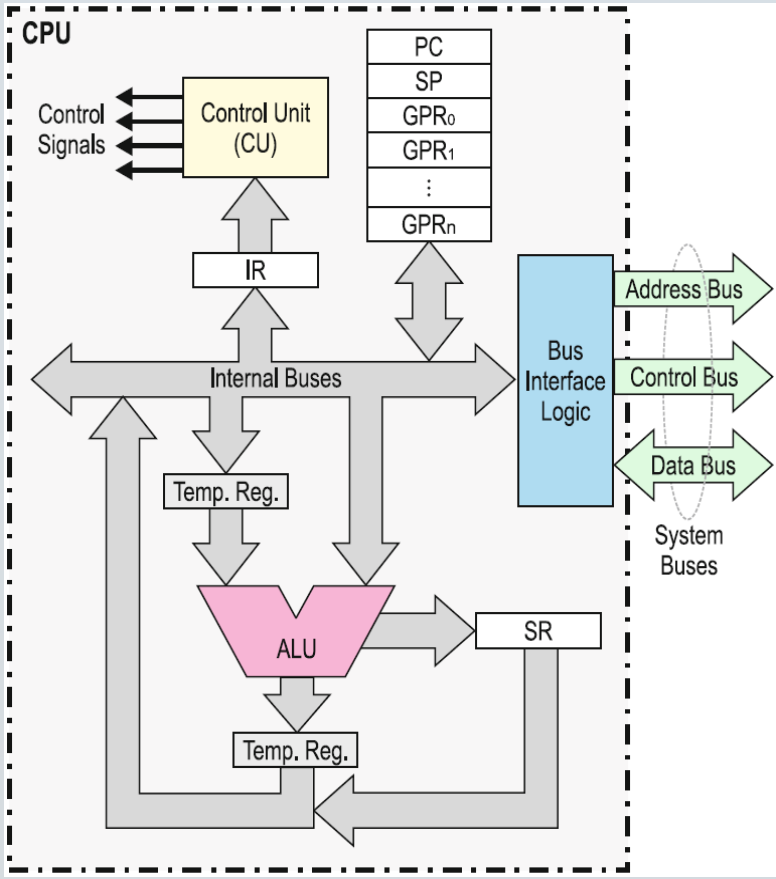


Aritmetičko logička jedinica

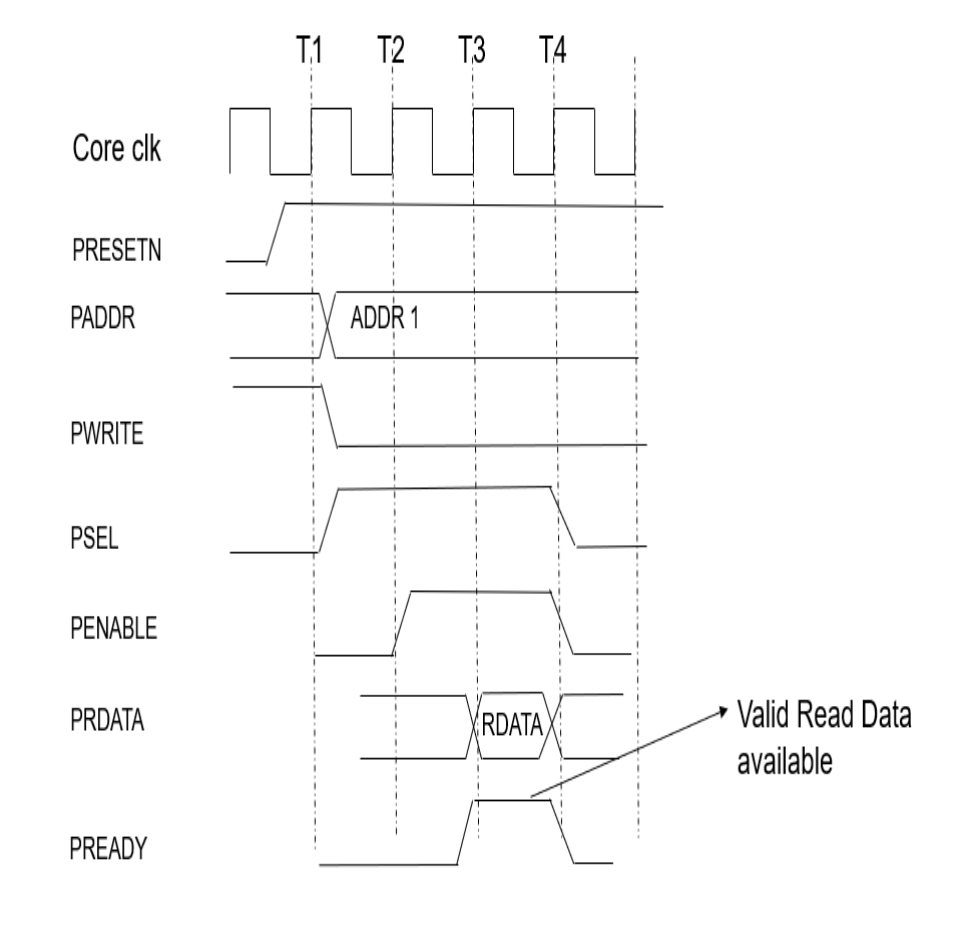
- **Aritmetičko logička jedinica (ALU)** je CPU komponenta u kojoj se obavljaju sve aritmetičke i logičke operacije koje sistem mora da podržava
- **Osnovne aritmetičke** operacije uključuju sabiranje, oduzimanje, komplementiranje i podržane su od strane većine ALU jedinica
- **Složenije ALU jedinice** mogu pružati hardversku podršku za složenije operacije kao što su množenje, deljenje
- **Logičke operacije** koje ALU jedinica može da izvede uključuju bitske I, ILI, NE i XOR operacije, kao i operacije pomeranja i rotiranja sadržaja registra
- **CU jedinica upravlja radom ALU** jedinice preko skupa upravljačkih signala pomoću kojih specificira koju je operaciju potrebno izvesti nad trenutnim ulaznim podacima
- **Širina ulaznih operanada ALU jedinice** obično se koristi za iskazivanje računske moći posmatranog procesora: 8-bitna, 16-bitna, 32-bitna, 64-bitna

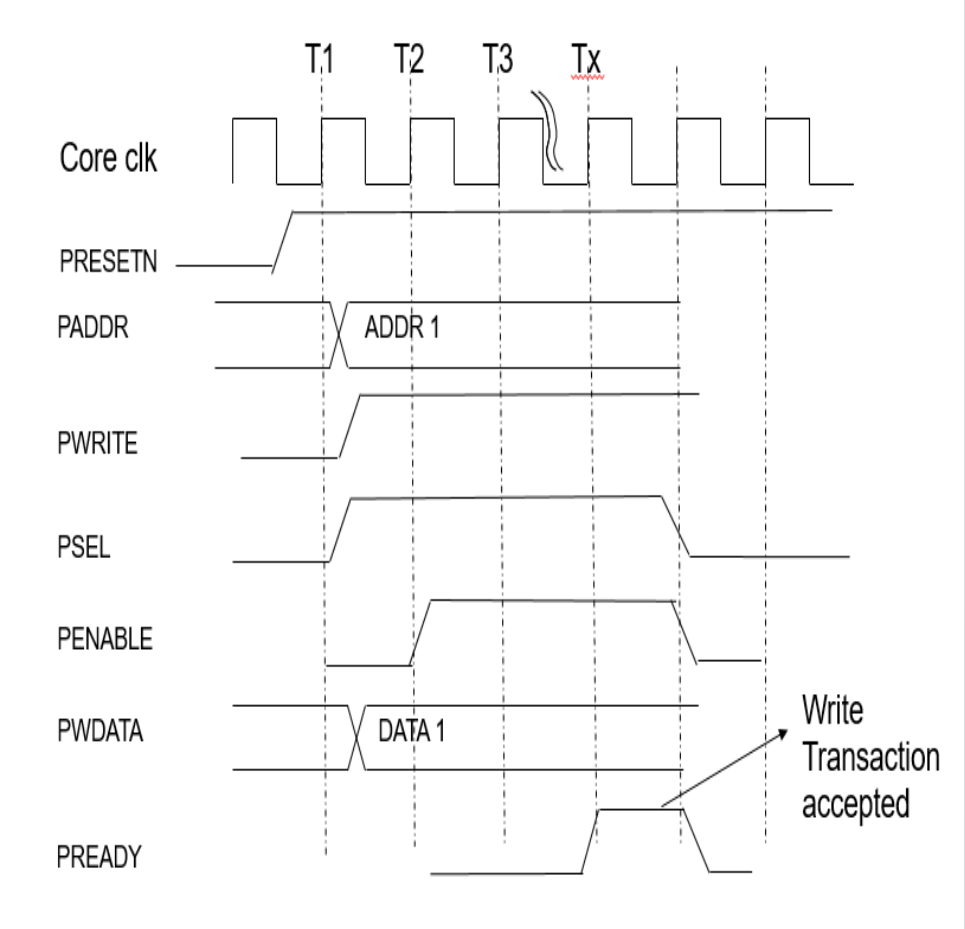
Logika za implementiranje sprežnog interfejsa

- **BIL modul** obezbeđuje nesmetani tok instrukcija i podataka od i ka procesoru
- BIL modul koordinira interakcijom između unutrašnjih magistrala procesora i sistemskih magistrala
- BIL modul određuje način na koji rade adresna, kontrolna i magistrala podataka
- U **malim embeded sistemima** BIL logika je u potpunosti sadržana unutar procesora i potpuno je transparentna za dizajnera
- U **distribuiranim embeded sistemima** BIL logika može zahtevati posebne jedinice koje obezbeđuju vezu između procesora i sistemskih magistrala, kao što su: kontroleri magistrala (bus controller), premošćivači magistrala (bus bridge), arbiteri magistrala (bus arbiter)



Primer APB protokola





web.eecs.umich.edu/~prabal/teaching/eecs373-f12/readings/ARM_AMBA3_APB.pdf

Registri opšte namene- GPR

- Unutrašnji registri procesora obezbeđuju privremeni smeštaj za podatke, adrese i upravljačke informacije

- Predstavljaju **najbrže memorijske jedinice** u embeded sistemu, ali su istovremeno vrlo malog kapaciteta

- Sadržaj unutrašnjih registara gubi se prilikom isključivanja napajanja

- Unutrašnji registri mogu se podeliti u dve velike grupe:
  - **Registri opšte namene (GPR)**: Registri koji nemaju neku specifičnu funkciju unutar procesora. Mogu se koristiti za čuvanje podataka, vrednosti promenljivih, adresa, itd., po volji programera. U zavisnosti od arhitekture, CPU može da sadrži od jednog do nekoliko desetina GPR registara.
  - **Registri posebne namene (SFR)**: Registri koji imaju tačno definisanu namenu unutar procesora. Većina procesora sadrži barem sledeće SFR registre: **registra instrukcija (IR), programski brojač (PC), pokazivač steka (SP), statusni registar (SR)**

Registar instrukcija

- Registar instrukcija (IR) čuva **kod naredne instrukcije** koju je potrebno izvršiti pomoću procesora
- Proces prebacivanja koda instrukcije iz programske memorije u IR registar zove se **faza prihvata instrukcije** (*instruction fetch*)
- U jednostavnim procesorima, IR može da čuva kod samo jedne instrukcije
- U složenijim procesorima postoji veći broj IR registara, koji omogućavaju istovremeno izvršavanje većeg broja instrukcija na većem broju funkcionalnih jedinica

Programski brojač

- **Programski brojač (PC)** sadrži **adresu naredne instrukcije** koju je potrebno prebaciti iz memoriju u procesor radi njenog izvršavanja
- Ponekad se naziva i **pokazivač instrukcija** (*instruction pointer, IP*)
- Svaki put kada se tekuća instrukcija dekoduje i izvrši, upravljačka jedinica (CU) ažurira sadržaj PC registra kako bi on pokazivao na memorijsku lokaciju u kojoj je smeštena naredna instrukcija koju je potrebno izvršiti
- **Ažuriranje sadržaja PC registra** najčešće podrazumeva njegovo inkrementovanje za unapred određenu konstantnu vrednost, ali u slučaju izvršavanja programskih skokova ili poziva potprograma sadržaj PC registra se umesto inkrementovanja menja sa potpuno novom vrednošću adrese
- Širina PC registra obično određuje maksimalnu veličinu memorije koju procesor može da adresira
- Sadržaj PC registra obično ne može da se direktno menja od strane programera

Pokazivač steka

- **Pokazivač steka** (***stack pointer*, SP**) pokazuje na “vrh” stek memorije
- Stek je **specijalizovani memorijski segment** koji služi za privremeno čuvanje podataka, kod koga se podacima može pristupati samo u strogo kontrolisanoj sekvenci
- Stek je nezamenjiv prilikom **implementacije potprograma** i prekidnih **potprograma**
- Većina procesora **nema hardverski definisan stek** već, kroz odgovarajući skup specijalizovanih instrukcija, omogućuje rad sa korisnički definisanim stekom unutar memorije podataka

Statusni registar I

- **Statusni registar (SR)**, poznat i pod imenom procesorska statusna reč (*Processor Status Word*, PSW) ili fleg registar (*Flag Register*) sadrži skup **indikatorskih bita, flegova, kao i dodatnih bita** koji se odnose na ili **kontrolišu status procesora**

- Pod flegom podrazumevamo jedan bit čija vrednost odslikava prisustvo ili odsustvo odgovajaćeg stanja

- Broj indikatorskih bita zavisi od složenosti procesora

- Većina indikatorskih bita odslikava stanje procesora neposredno nakon izvršavanja neke operacije na ALU jedinici, iako u opštem slučaju korisnik može da manipuliše njihovim sadržajem putem programa

Statusni registar II

- Indikatorski bitovi koji se najčešće mogu naći unutar procesora su:
  - **Indikator nule** (*Zero Flag*, ZF): Ima vrednost jedan ako je rezultat poslednje ALU operacije bio jednak nuli, u protivnom ima vrednost nula.
  - **Indikator prenosa** (*Carry Flag*, CF): Ovaj indikator se setuje svaki put kada se prilikom izvršavanja ALU operacije pojavi prenos, na primer kod operacije sabiranja. Međutim, neke ne-aritmetičke operacije, kao operacija pomeranja, mogu takođe uticati na postavljanje ovog indikatorskog bita.
  - **Indikator znaka** (*Sign Flag*, SF): Ovaj indikator se postavlja ako je rezultat poslednje ALU operacije bio negativan broj, u protivnom ima vrednost nula. Ovaj indikator zapravo reflektuje vrednost bita najveće značajnosti (MSB) u rezultatu ALU operacije.
  - **Indikator prekoračenja opsega** (*Overflow Flag*, OF): Ovaj indikator signalizira nastanak prekoračenja opsega prilikom operacija sabiranja i oduzimanja označenih brojeva.
  - **Indikator prekida** (*Interrupt Flag*, IF): Ovaj indikator nije asociran sa ALU jedinicom, već je pod programskom kontrolom. Ukoliko je postavljen, tada su prekidi u sistemu dozvoljeni.

Primer

- Posmatrajmo sledeće operacije sabiranja označenih 8-bitnih brojeva unutar ALU:

01001010+	10110100+	10011010+	11001010+
01111001=	01001100=	10111001=	00011011=
0 11000011	1 00000000	1 01010011	0 11100101

- Odrediti vrednost ZF, CF, SF i OF flegova nakon izvršavanja ovih operacija
  - ZF se postavlja ako je rezulatat operacije jednak nuli, ne uzimajući u obzir bit prenosa
  - CF direktno korespondira sa vrednošću bita prenosa
  - SF odgovara vrednosti MSB bita rezultata
  - OF se postavlja ako rezultat sabiranja dva broja istog znaka ima drugi znak

Primer

- Posmatrajmo sledeće operacije sabiranja označenih 8-bitnih brojeva unutar ALU:

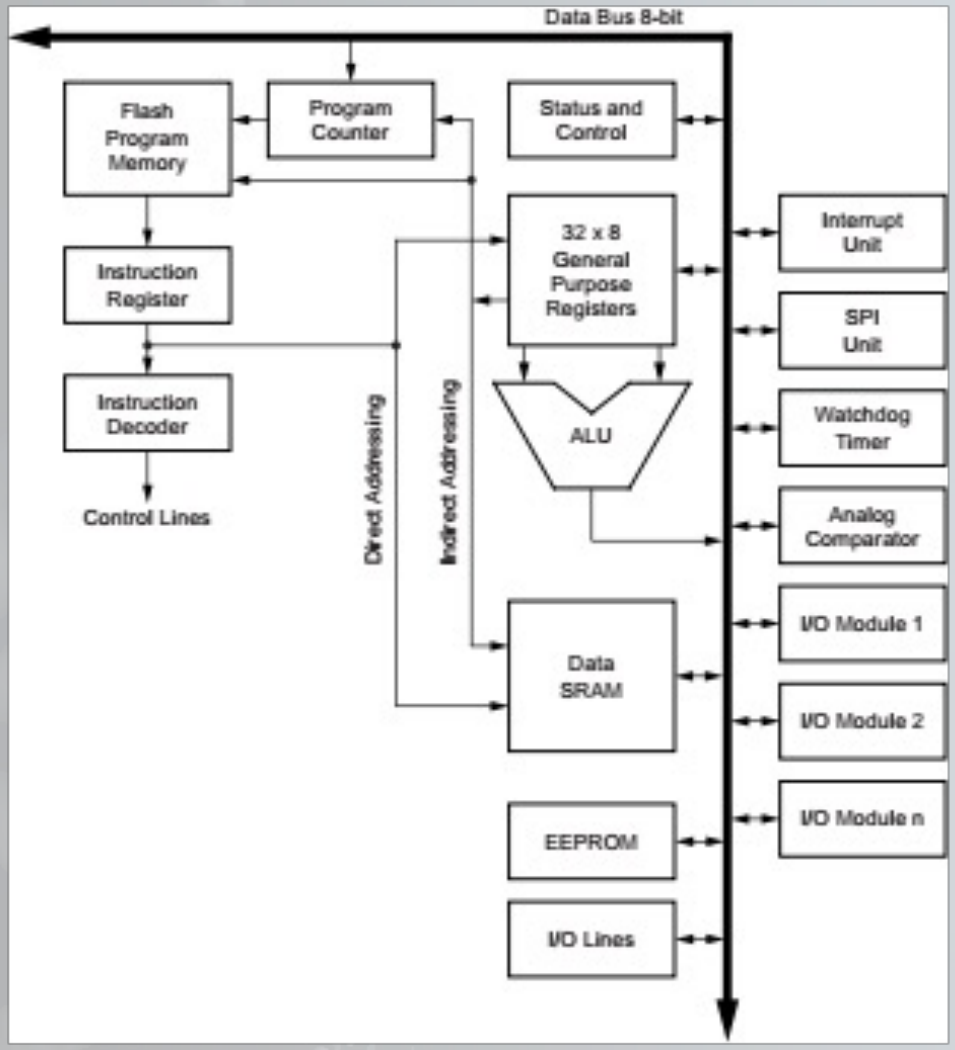
01001010+	10110100+	10011010+	11001010+
01111001=	01001100=	10111001=	00011011=
0 11000011	1 00000000	1 01010011	0 11100101

- Odrediti vrednost ZF, CF, SF i OF flegova nakon izvršavanja ovih operacija

  - ZF se postavlja ako je rezulatat operacije jednak nuli, ne uzimajući u obzir bit prenosa
  - CF direktno korespondira sa vrednošću bita prenosa
  - SF odgovara vrednosti MSB bita rezultata
  - OF se postavlja ako rezultat sabiranja dva broja istog znaka ima drugi znak

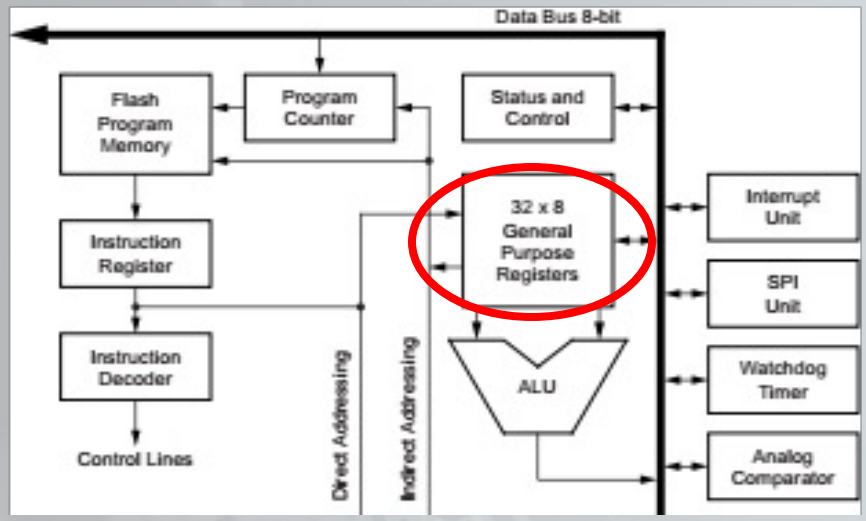
  - 4Ah+79h=C3h: ZF=0, CF=0, SF=1, OF=1.
  - B4h+4Ch=100h: ZF=1, CF=1, SF=0, OF=0.
  - 9Ah+B9h=153h: ZF=0, CF=1, SF=0, OF=1.
  - CAh+1Bh=E5h: ZF=0, CF=0, SF=1, OF=0.

Blok dijagram AVR arhitekture



- AVR CPU Core
- Odvojene memorije i magistrale za program i podatke (Harvard arhitektura)
- U toku izvršavanje jedne instrukcije, sledeća instrukcija je prihvaćena (pre-fetched) iz programske memorije
- Koncept omogućava da se insturkcije izvrše u svakom klok ciklusu
- Programska memorija je (in-system) reprogramabilna fleš memorija

Blok dijagram AVR arhitekture

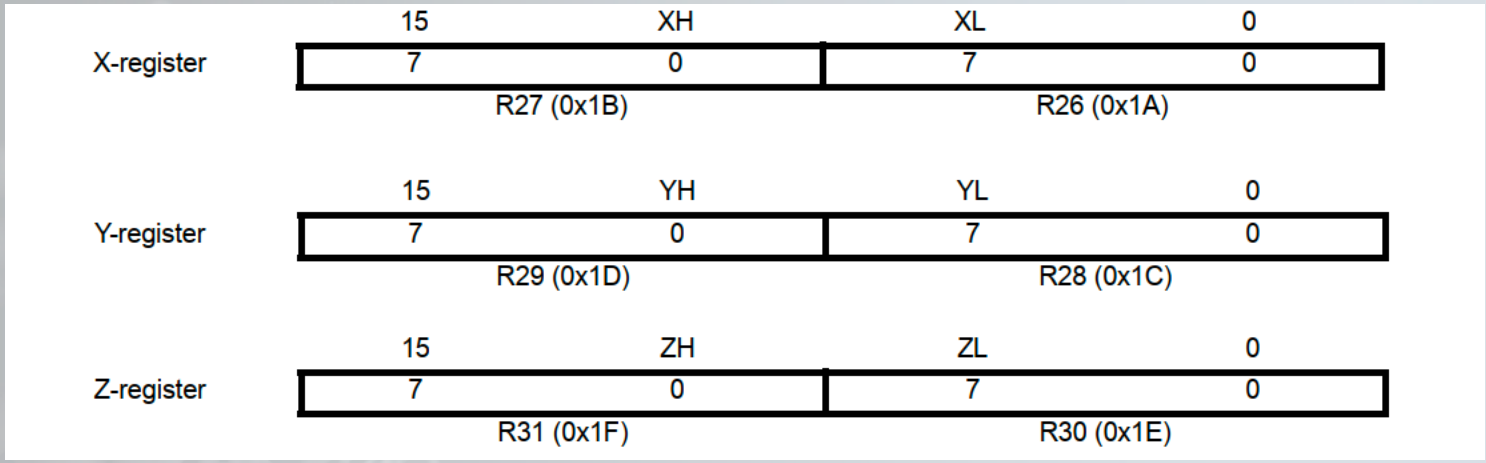


7	0	Addr.	
R0		0x00	
R1		0x01	
R2		0x02	
...			
R13		0x0D	
R14		0x0E	
R15		0x0F	
R16		0x10	
R17		0x11	
...			
R26		0x1A	X-register Low Byte
R27		0x1B	X-register High Byte
R28		0x1C	Y-register Low Byte
R29		0x1D	Y-register High Byte
R30		0x1E	Z-register Low Byte
R31		0x1F	Z-register High Byte

General Purpose Registers

- AVR CPU General Purpose Working Registers
- 32x8 GPR (R0-R31)
- Adrese 0x00-0x1F
- Tipične ALU operacije jednotaktne (single-cycle)
- 6 od 32 registra (poslednjih 6 adresa 0x1A-0x1F) mogu da se koriste kao 3 (X,Y,Z) specijalizovana 16-bitna registarska pokazivač za indirektno adresiranje
- Low Byte X registra (0x1A), High Byte X registra (0x1B),…

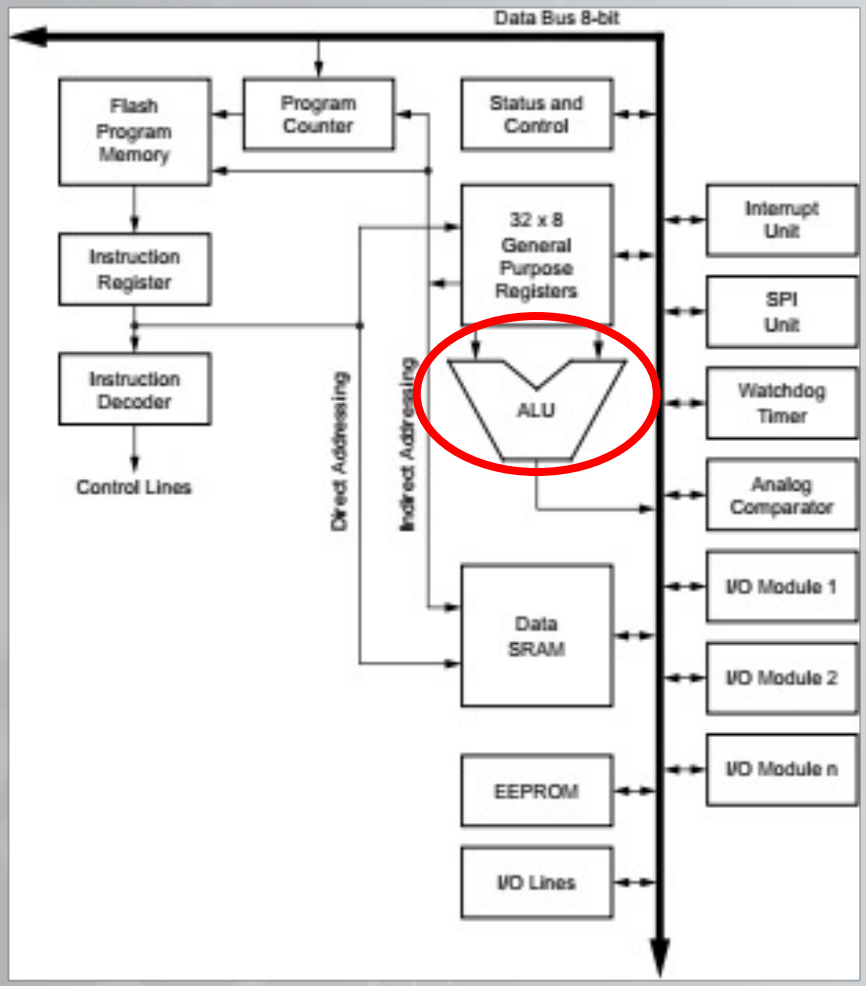
AVR GPR registri



7 0	Addr.	
R0	0x00	
R1	0x01	
R2	0x02	
...		
R13	0x0D	
R14	0x0E	
R15	0x0F	
R16	0x10	
R17	0x11	
...		
R26	0x1A	X-register Low Byte
R27	0x1B	X-register High Byte
R28	0x1C	Y-register Low Byte
R29	0x1D	Y-register High Byte
R30	0x1E	Z-register Low Byte
R31	0x1F	Z-register High Byte

- Registri R26..R31 imaju dodatne funkcije
- 16-bitni pokazivači adresa za indirektno adresiranje
- 3 indirektna adresna registra X, Y i Z
- U zavisnosti od adresnog moda ovi registri mogu imati funkciju fiksnog pomeraja, automatskog inkrementiranja i automatskog dekrementiranja

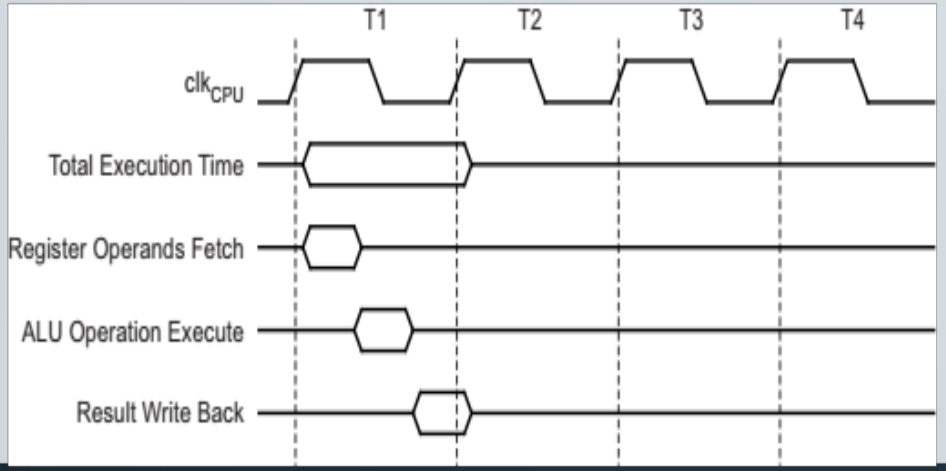
Blok dijagram AVR arhitekture



ALU jedinica

- Podržava aritmetičke i logičke operacije između registara ili između konstante i registra
- **Jednoregistarske operacije** mogu biti izvršene u ALU
- Nakon aritmetičke operacije statusni registar se osvežava na osnovu rezultata izvršene operacije
- ALU operacije u tri kategorije: aritmetičke, logičke i operacije nad bitovima (bit-functions)

Single Cycle ALU Operation



AVR statusni registar SREG

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Statusni registar

- Sadrži informacije o rezultatu poslednje izvršene aritmetičke instrukcije
- Može da se koristi za izmenu toka programa u cilju izvršavanja uslovnih operacija
- Osvežava se nakon svih ALU operacija
- Čuvanje sadržaja SR-a pri ulasku u prekidnu rutinu i vraćanje sadržaja nakon povratka iz prekida mora se realizovati softverski

AVR statusni registar SREG

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – I: Global Interrupt Enable

- Setuje se za dozvolu prekida
- Individualna dozvola prekida se ostvaruje zasebnim kontrolnim registrima
- Ukoliko je obrisan, ni jedan prekid nije dozvoljen
- I-bit se hardverski briše nakon pojave prekida i setuje se RETI instrukcijom da omogući sledeće prekide
- Može biti setova i obrisan sa aplikacijama koje sadrže SEI i CLI instrukcije

AVR statusni registar SREG

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 6 – T: Bit Copy Storage

- Za instrukcije kopiranja bita BLD (Bit LoaD) i čuvanja bita BST (Bit STore) koristi se T-bit kao izvor ili **odredište** za bit nad kojim se izvršava operacija
- Bit iz registra u registarskom fajlu može biti kopiran u T instrukcijom BST, i bit u T može biti kopiran u bit u registru registarskog fajla korišćenjem BLD instrukcije
- Privremeno čuva rezultat operacije na nivou bita
- Za kopiranje specifičnog bita iz GPR registra u T-bit SREG

AVR statusni registar SREG

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 5 – H: Half Carry Flag

- Indikacija poluprenosa (engl. half carry) u nekin aritmetičkim operacijama
- Indikacija da se desio prenos (engl. carry) izmedju bita 3 i bita 4 (granica izmedju nižeg i višeg nibla)

Bit 4 – S: Sign Bit S=N⊕V

- S-bit- ekskluzivno ili između negativnog flega N i flega V (fleg za prekoračenje u okviru aritmetike komplementa dvojke)

Bit 3– V: Two's Complement Overflow Flag

- V fleg podržava arimetiku komplementa dvojke, indikacija prekoračenja opsega (overflow)

AVR statusni registar SREG

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 2 – N: Negative Flag

- Indikacija negativnog rezultata aritmetičkih ili logičkih operacija

Bit 1 – Z: Zero Flag

- Indikacija nultog rezultata aritmetičkih ili logičkih operacija

Bit 0 – C: Carry Flag

- Indikacija prenosa (engl. carry) u okviru aritmetičkih ili logičkih operacija

AVR pokazivač steka

- Stek se koristi za čuvanje **privremenih podataka**, **lokalnih promenljivih** za čuvanje **povratnih adresa** nakon prekida i poziva potprograma
- Implementira se od više ka nižoj memorijskoj lokaciji
- Pokazivača steka uvek pokazuje na vrh steka
- Pokazivač steka pokazuje na podatke SRAM lokacije dodeljenih steku
- PUSH komanda dekrementira pokazivač steka
- Stek u SRAMu mora biti definisam programom pre poziva potprograma ili dozvole prekida
- Inicijalno pokazivač steka ima vrednost jednako poslednjoj adresi internog SRAM-a i pokazivač steka mora biti postavljen da pokazuje iznad početka SRAM-a

AVR pokazivač steka

- AVR pokazivač steka je implementiran u vidu dva 8-bitna registra u I/O modulu, broj bita koji će se koristiti zavisi od implementacije (SPL, SPH)

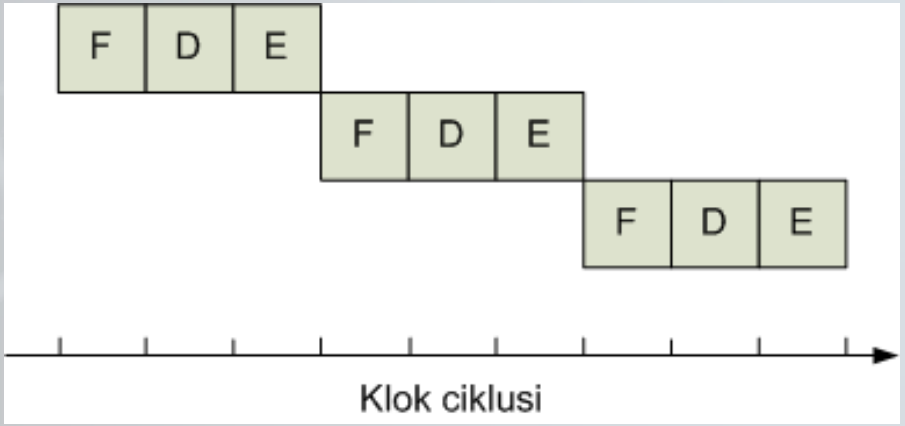
- U zavisnosti od memorijskog prostora sa podacima, poneka može biti dovoljan samo SPL deo, u tom slučaju SPH registar neće biti prisutan.

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	**SP15**	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
0x3D (0x5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	
	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

Napredne arhitekture procesora

- Osnovna arhitektura procesora koju smo do sada razmatrali bazira se na **sekvencijalnom izvršavanju CPU ciklusa**, koji se sastoji iz tri faze: faze prihvata (F), faze dekodovanja (D) i faze izvršavanja (E)

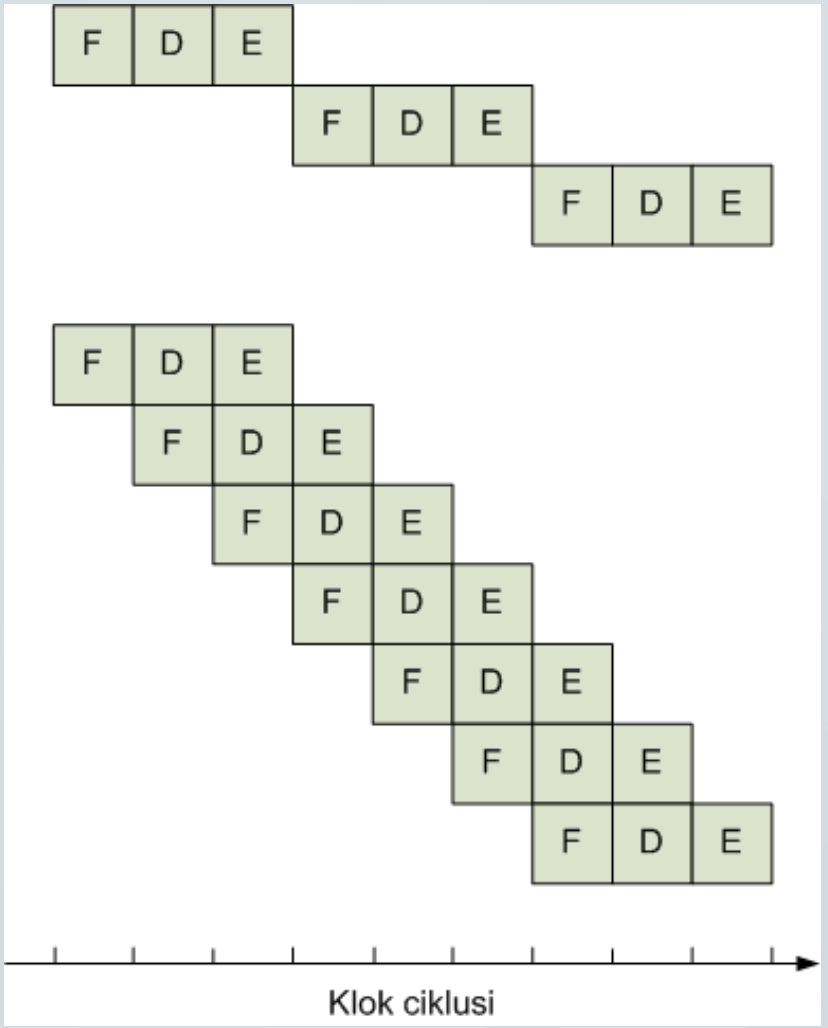
- Tek nakon što tekuća instrukcija prođe kroz sve tri faze (F, D, E) može se započeti sa izvršavanjem sledeće instrukcije



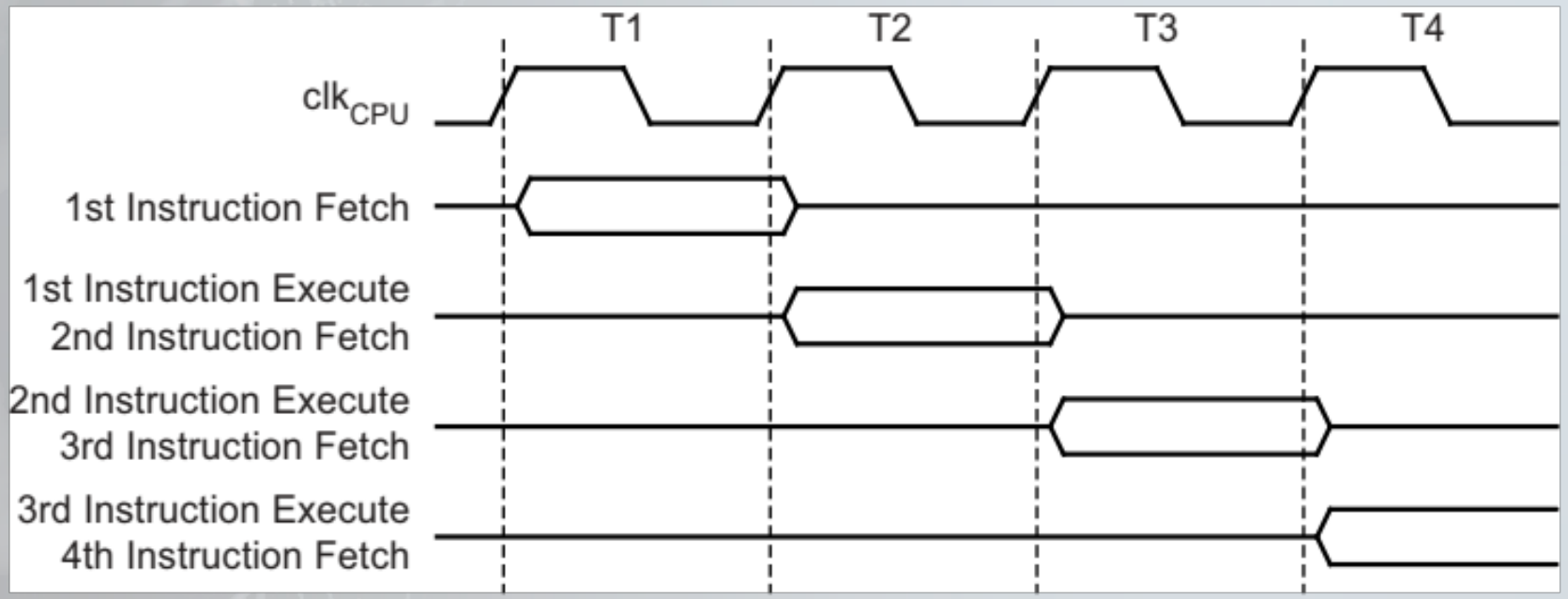
- Tokom prethodnih decenija mnogo napora uloženo je u osmišljavanje arhitektura koje će ubrzati vreme izvršavanja instrukcija

Procesori sa protočnom obradom I

- Uvođenjem **protočne obrade (*pipelining*)** moguće je izvršavanje većeg broja instrukcija istovremeno
- Prilikom protočne obrade, izvršavanje individualnih instrukcija je preklopljeno, tako što se **svaka instrukcija** nalazi u **drugoj fazi izvršavanja**
- Ovakav način izvršavanja, pod idealnim uslovima, koji podrazumevaju uvek pun lanac obrade (*pipeline*), omogućava da se u **svakom taktu kompletira izvršavanje jedne instrukcije**
- **Procesor sa *N* faza protočne obrade** je u idealnom slučaju u stanju da isti program izvrši u *N* puta brže od procesora koji ne koristi protočnu obradu
- U praksi, broj faza protočne obrade vrlo lako može biti veći od 10



ATmega328P


T1
T2
T3
T4
clk$_{CPU}$
1st Instruction Fetch
1st Instruction Execute
2nd Instruction Fetch
2nd Instruction Execute
3rd Instruction Fetch
3rd Instruction Execute
4th Instruction Fetch

Procesori sa protočnom obradom II

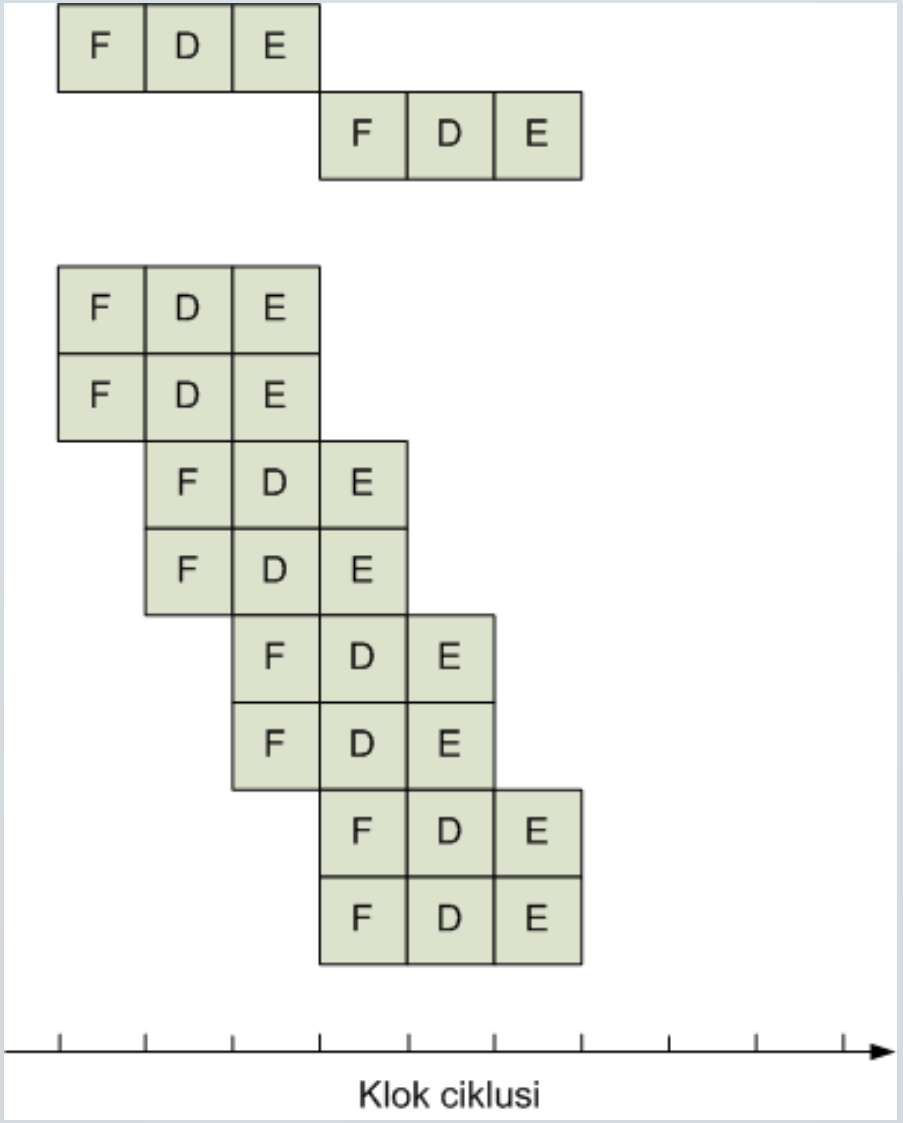
- Uvođenjem protočne obrade, procesorski resursi su mnogo efikasnije iskorišćeni

- Međutim, tehnika protočne obrade ima i izvesne nedostatke:
  - Zahteva uvođenje bafer registara između pojedinih faza procesiranja instrukcija. Ovi registri će uneti dodatno kašnjenje i smanjiti brzinu rada procesora

  - Teorijsko ubrzanje od *N* puta moguće je jedino ako je pipeline pun u svakom trenutku. Ovo nije uvek moguće obezbediti, na primer u slučaju izvršavanja instrukcija **uslovnog grananja**. Pošto će se adresa naredne instrukcije koju je potrebno izvršiti znati tek nakon kompletiranja instrukcije uslovnog grananja, a mi u pipeline u svakom taktu moramo uvoditi novu instrukciju, može se desiti da su instrukcije koje se nalaze u pipelinu nakon instrukcije uslovnog grananja pogrešne i da je umesto njih potrebno izvršavati neki drugi niz instrukcija (koji pripada drugoj grani). Postoje dva načina da se ovaj problem reši: zaustavljanje pipelin-a do trenutka u kojem znamo koju narednu instrukciju je potrebno izvršiti (**pipeline stalling**); pražnjenje pipeline-a (**pipeline flushing**), kada se iz pipeline-a uklanjaju sve instrukcije koje nije potrebno izvršavati. Bilo koja od metoda da se koristi, ona će dovesti do degradacija performansi.

Superskalarni procesori I

- Kod superskalarnih procesora postoji barem dva lanca protočne obrade
- Veći broj lanaca protočne obrade omogućava da se u svakom taktu **više od jedne instrukcije** nalazi u svakoj od faza izvršavanja
- Teoretski, superskalarni procesor sa ***K* lan**aca protočne obrade, od kojih se svaki sastoji od *N* faza obrade u stanju je da ***K*·*N* puta brže izvršava** instrukcije od običnog procesora
- Međutim, u praksi se vrlo retko sreću programi kod kojih je u svakom trenutku moguće u paraleli izvršiti *K* instrukcija
- Ovo je stoga što su sukcesivne instrukcije vrlo retko međusobno nezavisne jedna od druge, odnosno, za izvršavanje jedne instrukcije potrebne su informacije koje se generišu tokom izvršavanje druge



Superskalarni procesori II

- Superskalarni procesori su vrlo komplesna instegrisana kola, ne samo zbog toga što poseduju veliku redundantnost hardverskih resursa (*K* potpuno identičnih lanaca protočne obrade), već i zbog toga što moraju da poseduju **dodatnu logiku** za određivanje postojanja međuzavisnosti između instrukcija koje je potrebno izvršiti

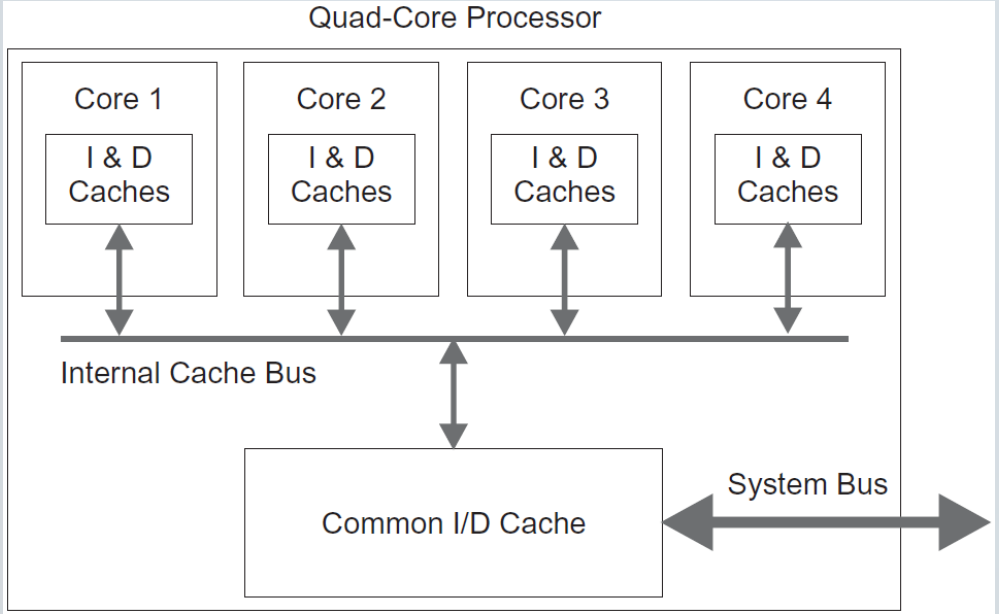
- Ukoliko se u dizajn superskalarnog procesora uključi i logika za izvršavanje instrukcija van reda (*out-of-order instruction execution*) **njihova hardverska složenost dodatno raste**

- Usled toga što stvarne performanse superskalarnog procesora mogu u velikoj meri varirati između najboljeg i najlošijeg vremena izvršavanja programa, vrlo je teško **projektovati determinstički embeded sistem** baziran na superskalarnom procesoru

- Superskalarni procesori se uglavnom koriste u radnim stanicama, u aplikacijama koje **ne zahtevaju rad u realnom vermenu**

Višejezgarni procesori

- Usled napretka tehnologije izrade integrisanih kola danas je moguće **na jednom integrisanom kolu fabrikovati više od jednog procesora**
- Svaki od procesora može da bude realizovan kao procesor sa **protočnom obradom ili superskalarni procesor**
- **Ubrzanje** rada sistema ovog puta se ostvaruje **paralelnim izvršavanjem** nekoliko slabo zavisnih programa
- Ovo je vrlo česta situacija prilikom projektovanja emebeded sistema, u kojima imamo veći broj zadataka koje je potrebno obavljati, a **zadaci su međusobno slabo povezani**
- Slaba povezanost znači da je količina informacija koju zadaci razmenjuju između sebe vrlo mala



Teorijska pitanja P3

Predavanja 3- Organizacija mikroračunarskog sistema 1

Spisak teorijskih pitanja uz Predavanja 3

1. Osnovna arhitektura mikroračunarskog sistema.

2. Objasniti glavne razlike između mikroprocesora i mikrokontrolera.

3. Centralna procesirajuća jedinica - navesti glavne komponente i ukratko ih objasniti.

4. Upravljačka jedinica.

5. Aritmetičko-logička jedinica.

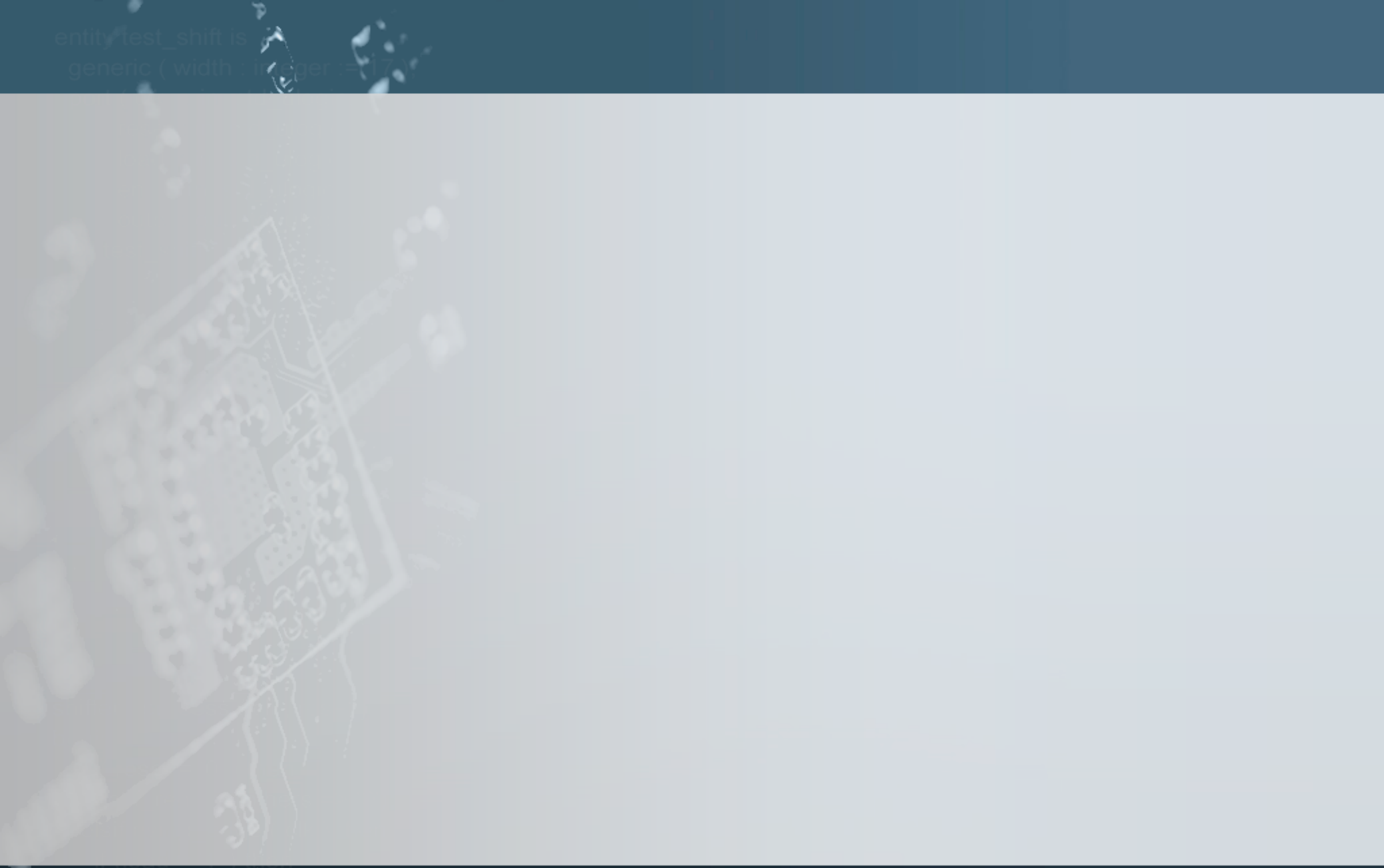
6. Logika za implementaciju sprežnog interfejsa.

7. Unutrašnji registri procesora. Navesti i objasniti najvažnije unutrašnje registre.

Predavanja 3- Organizacija mikroračunarskog sistema 1

Spisak teorijskih pitanja uz Predavanja 3

8. AVR arhitektura CPU sa memorijom.
9. AVR arhitektura - unutrašnji registri procesora.
10. AVR arhitektura- ALU jedinica.
11. Napredne arhitekture procesora - procesori sa protočnom obradom
12. AVR mikrokontroleri- protočna obrada.
13. Napredne arhitekture procesora - superskalarni procesori
14. Napredne arhitekture procesora - višejezgarni procesori


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