# Arhitektura mikrokontrolera

**Mikrokontroler je mali računar koji se proizvodi kao jedinstveno integrisano kolo koje sadrži procesorsko jezgro, memoriju i programabilne ulazno/izlazne periferije. Kada je u pitanju memorija, tipičan mikrokontroler sadrži programsku memoriju u obliku neke varijante EPROM-a (najčešće FLASH) i operativnu memoriju u vidu RAM-a malog kapaciteta. Ovde će biti reči o osnovoj strukturi mikrokontrolera, nakon čega će više pažnje biti posvećeno arhitekturi procesora, pri čemu će fokus biti na arhitekturi procesorskog jezgra Atmel AVR mikrokontrolera.**

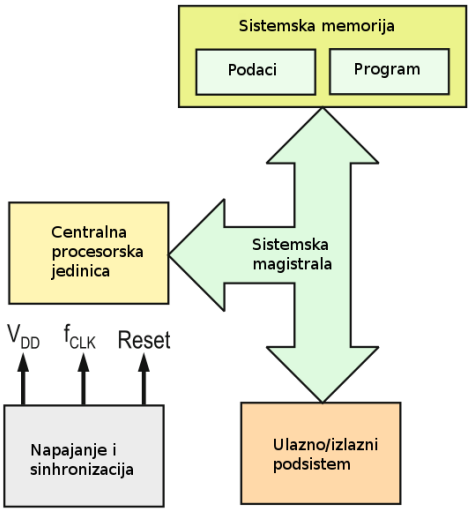
## Uvod - osnovna struktura mikroračunarskog sistema

Tipičan mikroračunarski sistem sadrži tri fundamentalne komponente: centralnu procesorsku jedinicu (engl. *Central Processing Unit - CPU*), sistemsku memoriju i neki oblik ulazno/izlaznog podsistema. Ove komponente povezane su višestrukim linijama grupisanim u skladu sa njihovim funkcionalnostima. Linije koje povezuju osnovne elemente mikroračunarskog sistema nazivaju se *sistemskim magistralama* (engl. *system bus*). Kao dodatak, osim tri osnovna elementa mikroračunarskog sistema, dodatne komponente obezbeđuju neophodne uslove za funkcionisanje sistema, kao što je naponska stabilizacija, generisanje signala takta i sl. Na slici 1 prikazana je osnovna arhitektura mikroračunarskog sistema.

Komponente mikroračunarskog sistema mogu biti međusobno povezane u funkcionalnu celinu na više različitih načina. Tipična implementacija podrazumeva da su svi elementi mikroračunarskog sistema nezavisne komponente (integrisana kola, engl. *chip*), dok bi alternativno rešenje bilo integracija svih komponenti u okviru jednog čipa, u strukturi koja se naziva *Mikrokontroler*. Nezavisno od toga kako je realizovan mikroračunarski sistem, svaka od komponenti tog sistema ima specifičnu funkciju i namenu:

1. Centralna procesorska jedinica (CPU) predstavlja ključni deo svakog mikroračunarskog sistema. CPU čita instukcije iz programske memorije, dekoduje ih i izvršava ih. Osim toga, CPU je zadužen i za adekvatno korišćenje periferijskih jedinica koje sačinjavaju ulazno/izlazni podsistem.

2. Sistemska memorija je zadužena za smeštanje programa i podataka, koji se koriste od strane centralne procesorske jedinice. U okviru mikrokontrolera, obično su prisutna dva tipa




**Slika 1:** *Osnovna arhitektura mikroračunarskog sistema*

memorije: programska memorija i memorija podataka. Programska memorija ima ulogu skladištenja programa u formi sekvence mašinskih instrukcija koju procesor izvršava. Program određuje funkcionalnost celokupnog sistema. Memorija za podatke ima ulogu čuvanja podataka koji se obrađuju tokom izvršavanja programa.

3. Ulazno/Izlazni podsistem uključuje sve komponente, tj. periferijske uređaje koji omogućavaju centralnom procesoru da razmenjuje informacije sa drugim komponentama sistema i spoljašnim svetom.

4. Sistemske magistrale predstavljaju skup linija koje povezuju CPU, memoriju i ulazno/izlazni podsistem. Postoje grupe linija koje obavljaju različite funkcije unutar sistema: adresna magistrala, magistrala podataka i kontrolna magistrala.

## Razlike mikrokontrolera i mikroprocesora

Mikroprocesorka jedinica (skraćeno mikroprocesor, engl. *MPU*) sadrži centralnu procesorsku jedinicu opšte namene. Prilikom projektovanja mikroračunarskog sistema baziranog na mikroprocesoru, svi elementi sistema prikazani na slici 1 (magistrale, memorije i periferijske jedinice) moraju biti povezane kao eksterne komponente. Osim toga, u naprednije karakteristike mikroprocesora opšte namene mogu spadati još i:

- Optimizovana arhitektura u cilju preuzimanja programa i podataka iz eksterne memorije (skrivena memorija, engl. *cache*);
- Mogućnost protočne obrade više instrukcija istovremeno (engl. *pipelining*);
- Predviđanje grananja (engl. *branch prediction*);
- Postojanje numeričkih ko-procesora, itd.

Tipičan primer mikroračunarskog sistema baziranog na mikroprocesoru opšte namene je personalni računar (engl. *Personal Computer - PC*) kao i takozvani *mainframe* računari. Najpoznatiji proizvođači mikroprocesora su Intel, AMD, Freescale, Zilog, NXP, Texas Instruments i mnogi drugi. Dizajn mikroprocesora je znatno napredovao u poređenju sa prvobitnim modelima koji su se pojavili početkom 70-tih godina prošlog veka. Intel-ov prvobitni 4004 iz 1971. godine je bio napravljen u 10$\mu$m tehnologiji, koristio je takt na 400kHz i bio je sačinjen od 2250 tranzistora. Intel-ov Xeon E7 mikroprocesor, predstavljen 2011. godine, napravljen je u 32nm tehnologiji, koristi takt od 2GHz i sadrži oko 2.6x10$^9$ tranzistora[1].

Mikrokontroleri (engl. *Micro Controller Unit - MCU*), bazirani su na jezgru mikroprocesora, odnosno centralnoj procesorskoj jedinici (CPU) koja je po pravilu manje složenosti u poređenju sa mikroprocesorom opšte namene. Ovakvoj CPU jedinici je pridodata memorija (kako programska tako i memorija za podatke) i više različitih tipova periferijskih jedinica u koje obično spadaju:

- Grupe digitalnih ulazno/izlaznih priključaka opšte namene (portovi)
- Brojači i tajmeri
- Podsistem za generisanje i obradu prekida
- Interfejsi za serijsku komunikaciju
- Komparatori, A/D i D/A konvertori

Ovakva struktura omogućava fleksibilnost i implementaciju kompletnog sistema korišćenjem minimalnog broja eksternih komponenti. Na slici 2 prikazana je tipična arhitektura mikrokontrolera. Iako mikrokontroleri imaju slične karakteristike kao i mikroprocesori opšte namene, po pravilu su znatno manje kompleksnosti i više su prilagođeni aplikaciji za koju su namenjeni. Mikrokontroleri su uglavnom grupisani u kategorije (familije) koje karakterišu zajedničke karakteristike (struktura registara, set instrukcija, modovi adresiranja, itd). Na tržištu je danas dostupno mnoštvo mikrokontrolerskih familija različitih proizvođača. Među njima se po značaju i zastupljenosti na tržištu posebno izdvajaju PIC kontroleri (proizvođač Microchip), AVR kontroleri (Atmel), ARM (razni proizvođači) i kontroleri zasnovani na 8051 arhitekturi (Intel).

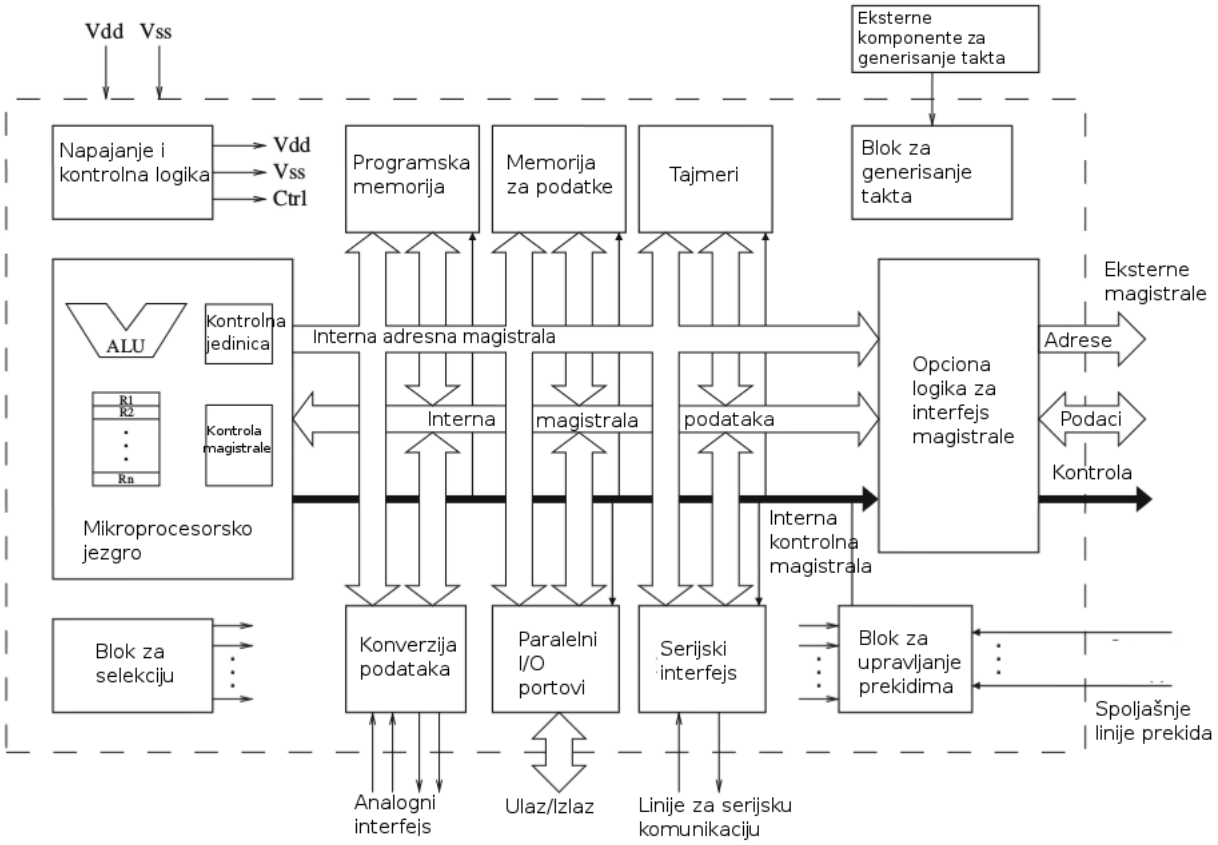
## RISC i CISC arhitektura

Mikroračunarski sistem izvršava softver koji je podržan od strane hardverske arhitekture sistema. Mikroračunarski sistemi se uglavnom optimizuju po pitanju softvera, ili hardvera. U skladu sa tim, postoje dve standardne arhitekture mikroračunara: CISC i RISC.

CISC (*Complex Instruction Set Computing*) mašine karakteriše:

- Promenljiva dužina instrukcijske reči (različit broj bita se koristi za kodovanje instrukcija)
- Manja dužina programa u odnosu na RISC arhitekture
- Instrukcije koje se izvršavaju tipično u nekoliko sukcesivnih perioda sistemskog takta.

[1]Tokom prethodnih 40-ak godina, složenost mikroprocesora se povećavala u skladu sa Murovim zakonom: broj komponenti (tranzistora) po jedinici površine integrisanog kola se duplira približno svake dve godine.




**Slika 2:** *Arhitektura mikrokontrolera*

CISC arhitektura izvršava više mikrooperacija u okviru svake instrukcije, npr. prihvat koda instrukcije iz programske memorije, izvršavanje aritmetičke ili logičke operacije i smeštanje rezultata nazad u memoriju.

RISC (*Reduced Instruction Set Computing*) mašine, sa druge strane, dizajnirane su sa idejom da proces izvršenja instrukcija bude što jednostavniji, čak i po cenu veće dužine programa koji će se izvršavati. Ovakav pristup pojednostavljuje strukturu hardvera. Kod RISC arhitekture, izvršavanje svake pojedinačne instrukcije je znatno skraćeno u poređenju sa CISC arhitekturama.

## Centralna procesorska jedinica (CPU)

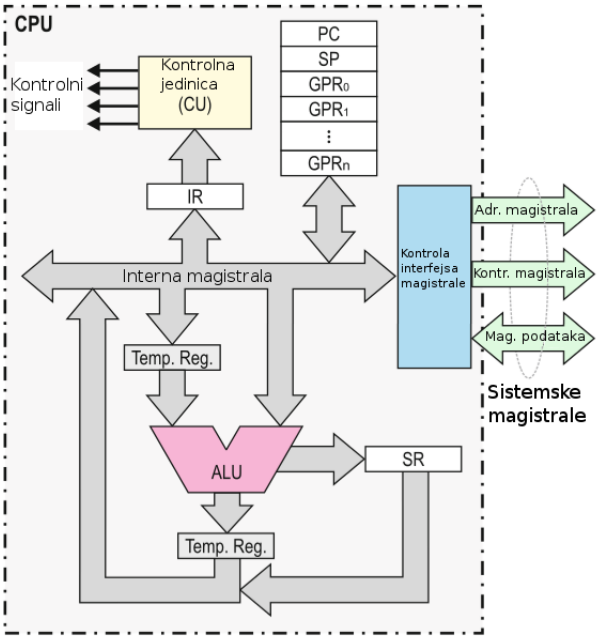
Centralna procesorska jedinica predstavlja najvažniji deo mikroračunarskog sistema. CPU je zadužen za izvršavanje instrukcija, odnosno njegova uloga je generisanje odgovarajuće sekvence signala koja proizvodi akciju izvršavanu na hardveru mikroračunarskog sistema. Minimalan skup komponenti koje sačinjavaju arhitekturu centralne procesorske jedinice su:

- Hardverske komponente
  - Aritmetičko-Logička jedinica (engl. *Arithmetic Logic Unit - ALU*)
  - Kontrolna jedinica (engl. *Control Unit - CU*)

  - Skup registara
  - Kontrola interfejsa magistrale

- Softverske komponente
  - Set instrukcija
  - Modovi adresiranja

Instrukcije i modovi adresiranja su određeni i definisani specifičnostima hardverskih jedinica ALU i CU. U nastavku će biti nešto više reči o hardverskoj strukturi centralne procesorske jedinice.

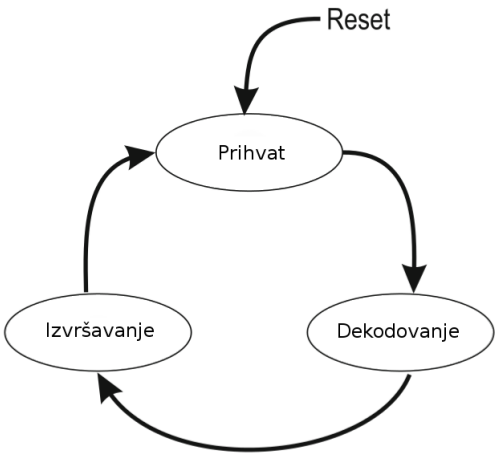



**Slika 3:** *Centralna procesorska jedinica*

Slika 3 prikazuje pojednostavljeni dijagram centralne procesorske jedinice sa njenim internim hardverskim blokovima. Ovi blokovi omogućavaju da CPU pristupa programu i podacima koji su smešteni u memoriji i/ili periferijskim jedinicama. Sam program se sastoji od sekvence pojedinačnih instrukcija koje su sadržane u setu podržanih instrukcija datog CPU. Program smešten u memoriji određuje sekvencu operacija koje će se izvršiti na sistemu. Prilikom obrade podataka, svaka komponenta centralne procesorske jedinice igra važnu ulogu i zadužena je tačno za jednu vrstu operacija koje ne mogu biti izvršene od stane drugih komponenata, alternativno.

Hardverske komponente u okviru CPU koje izvršavaju operacije na podacima su ALU, interna magistrala podataka, ali i ostali funkcionalni blokovi kao što su jedinica za aritmetiku u pokretnom zarezu (engl. floating-point unit), hardverski množači itd.

Hardverske komponente koje vrše operacije vezane za kontrolu sistema ulaze u sastav kontrolne jedinice (engl. *Control Unit*). Jedinica za kontrolu magistrale, kao i komponente za sinhronizaciju se takođe najčešće posmatraju kao deo kontrolne jedinice.




**Slika 4:** *Konačni automat kontrolne jedinice*

## Kontrolna jedinica (CU)

Kontrolna jedinica (CU) upravlja funkcionalnošću centralne procesorske jedinice, implementirajući konačni automat (engl. *Finite State Machine - FSM*) koji ciklično ponavlja tri stanja: Prihvat (engl. *Fetch*), Dekodovanje (engl. *Decode*) i Izvršavanje (engl. Execute), što je prikazano na slici 4. Ova tri stanja predstavljaju elementarne operacije koje se obavljaju prilikom izvršavanja svake pojedinačne instrukcije i odnose se na:

- Prihvat instrukcije iz memorije
- Dekodovanje instrukcije nakon čega CPU "zna" koja instrukcija treba da se izvrši
- Izvršenje instrukcije.

Prihvati-dekoduj-izvrši ciklus se često u literaturi naziva i instrukcijski ciklus. Kompletan instrukcijski ciklus tipično zahteva nekoliko taktova procesora da se izvrši, u zavisnosti od same instrukcije i operanada koji se u njoj koriste (konstante, varijable iz memorije ili sadržaji lokalnih registara). Neretko se dešava da CPU ima svoj interni takt koji je više učestanosti od sistemskog takta (čak i do četiri puta više). U takvim slučajevima, u literaturi se može videti da je za izvršenje instrukcije dovoljan jedan sistemski takt, što je naravno tačno, ali treba imati u vidu da se zapravo instrukcija izvrši nakon četiri (ili više) internih taktova. Nekoliko blokova učestvuje u instrukcijskom ciklusu, među kojima specijalnu ulogu imaju registri PC (programski brojač, engl. *Program Counter*) i IR (instrukcijski registar, engl. *Instruction Register*). Tokom kompletnog ciklusa procesor prolazi kroz sledeća stanja:

1. **Stanje prihvata instrukcije:** Tokom stanja prihvata nove instrukcije, korišćenjem bloka kontrole interfejsa magistrale (engl. *Bus Interface Logic - BIL*), instrukcija se iz memorije učitava u CPU. Programski brojač (PC) obezbeđuje adresu u memoriji sa koje se instrukcija čita. Adresirana instrukcija se čita korišćenjem magistrale podataka i smešta se u instrukcijski registar (IR).

2. **Stanje dekodovanja instrukcije:** Nakon prihvata instrukcije, CU prelazi u stanje dekodovanja, u kome se značenje instrukcije "dešifruje". Dekodovana informacija se koristi kako bi se slali odgovarajući upravljački signali ka CPU komponentama u cilju izvršavanja aktivnosti predviđenih samom dekodovanom instrukcijom.

3. **Stanje izvršavanja instrukcije:** U ovom stanju, CU šalje komande odgovarajućim funkcionalnim jedinicama u okviru CPU u cilju izvršavanja aktivnosti određenih instrukcijom. Na kraju ove faze izvršavanja, sadržaj programskog brojača se inkrementira, kako bi pokazivao na adresu sledeće instrukcije u programskoj memoriji, koja će se izvršiti tokom narednog instrukcijskog ciklusa.

Nakon izvršne faze, CU šalje komande bloku kontrole interfejsa magistrale, kako bi ona koristeći sadržaj programskog brojača preuzela narednu instrukciju iz memorije, čime se inicira naredni instrukcijski ciklus (novi prihvat instrukcije).

Ciklus može obuhvatati među-cikluse slične instrukcijskom ciklusu, u slučaju kada je tokom dekodovanja instrukcije potrebno preuzeti iz memorije dodatne podatke (npr. za instrukciju koja sabira dva broja, brojeve koje treba sabrati). Ovo takođe zavisi i od moda adresiranja koji se koristi, ali o ovome će biti više reči kasnije.

Obzirom na to da je kontrolna jedinica realizovana kao konačni automat, neophodan je Reset signal kako bi se procesor doveo u početno stanje, koje prethodi izvršenju prve istrukcije. U cilju preuzimanja prve instrukcije iz memorije, sadržaj programskog brojača nakon reseta je uvek takav da pokazuje na prvu instrukciju koja treba da se izvršava (u slučaju mikrokontrolera, to je uglavnom adresa 0). Adresa prve instrukcije često se naziva *reset vektor*.

## Aritmetičko Logička jedinica (ALU)

Aritmetičko logička jedinica je komponenta CPU zadužena za sve aritmetičke i logičke operacije koje treba izvršiti u datom mikroračunarskom sistemu. Osnovne aritmetičke operacije kao što su sabiranje, oduzimanje i komplement (predstava negativnih brojeva), podržane su od strane svih ALU. Neke složenije, sa druge strane, uključuju i hardverske komponente za kompleksnije operacije, kao što su množenje i deljenje. Ipak, u većini slučajeva ove operacije se vrše ili softverski (korišćenjem nekog od algoritama koji koriste već postojeće elementarne aritmetičke operacije), ili korišćenjem dodatnih periferijskih jedinica, kao što je npr. hardverski množač.

Logičke operacije koje se tipično izvršavaju od strane ALU su operacije koje rade sa pojedinačnim bitima u okviru bajta ili reči (I, ILI, NE, EX-ILI). Takođe, u ove operacije spadaju i pomeranje i rotiranje. Ove operacije su izuzetno značajne jer omogućavaju promenu određenih bita u registrima, bez uticaja na ostale bite.

Kontrolna jedinica upravlja aritmetičko-logičkom jedinicom tako što specificira koja operacija treba da se izvrši, prosleđuje joj operande nad kojima se vrši operacija, i obezbeđuje prostor gde će biti sačuvan rezultat. Kapacitet ALU određen je arhitekturom mikroprocesora: na primer, za 16-bitni mikroprocesor ALU ima mogućnosti izvršavanja operacija na 16-bitnim podacima. Ovo uveliko određuje i samu strukturu mikroprocesora, jer podrazumeva da je širina magistrale podataka takođe 16, kao i širina registara koji se koriste.

## Kontrola interfejsa magistrala

Kontrola interfejsa magistrala je struktura u okviru centralne procesorske jedinice koja koordinira interakciju između internih magistrala i sistemskih magistrala. Ovaj blok definiše način na koji funkcioniše prenos podataka posredstvom sistemskih magistrala. U slučaju jednostavnih sistema ovaj blok je sadržan u okviru CPU, dok je u slučaju složenijih sistema najčešće potrebno dodavanje eksternih modula koji su zaduženi da obezbede ovu funkcionalnost. Primeri ovih modula su periferije za kontrolu magistrala, mostovi (engl. *bridges*) i hardverski moduli za arbitriranje na magistralama.

## Registri

Registri procesora omogućavaju privremeno smeštanje podataka, memorijskih adresa i kontrolnih informacija tako da im se može brzo i jednostavno pristupiti. Oni predstavljaju najbržu memoriju u mikroračunarskom sistemu, koja sa druge strane ima i najmanji kapacitet. Sadržaj registara u okviru CPU se gubi nakon nestanka napona napajanja. U načelu, registri se mogu podeliti u dve grupe: registri opšte namene i specijalizovani registri.

Registri opšte namene su registri koji nisu "vezani" za specifične funkcije procesora i mogu da se koriste za smeštanje podataka, promenljivih ili pokazivača na adrese, po potrebi. U skladu sa ovim, često se u literaturi klasifikuju kao adresni ili registri za podatke. U zavisnosti od arhitekture procesora, CPU može sadržati do nekoliko desetina registara opšte namene.

Registri specijalne namene su registri koji su zaduženi za specifične funkcije u radu CPU. Navažniji registri koji su sadržani u okviru CPU strukture su:

- Instrukcijski registar (IR)
- Programski brojač (engl. *Program Counter - PC*) koji se često naziva i instrukcijski pokazivač (engl. *Instruction Pointer - IP*)
- Pokazivač steka (engl. *Stack Pointer - SP*)
- Statusni registar (SREG), koji sadrži indikatore ("zastavice", engl. *flags*)

### Instrukcijski registar (IR)

Instrukcijski registar čuva instrukciju koja se trenutno dekoduje i izvršava od strane CPU. Akcija prenosa instrukcije iz memorije u IR naziva se prihvat instrukcije.

### Programski brojač (PC)

Ovaj registar čuva adresu instrukcije koja će biti učitana iz programske memorije od strane centralne procesorske jedinice. Često se naziva i instrukcijski pokazivač. Svaki put kada se instrukcija prihvata i dekoduje, kontrolna jedinica inkrementira vrednost PC registra, kako bi pokazivao na narednu instrukciju koja će se izvršavati tokom sledećeg instrukcijskog ciklusa. Ovakvo ponašanje se može promeniti tokom izvršavanja programa, na više načina (npr. instrukcijama grananja kada se sadržaj programskog brojača zamenjuje novom adresom na koju treba skočiti). Obzirom da PC sadrži adresu, njegova širina mora biti usklađena sa kapacitetom programske memorije.

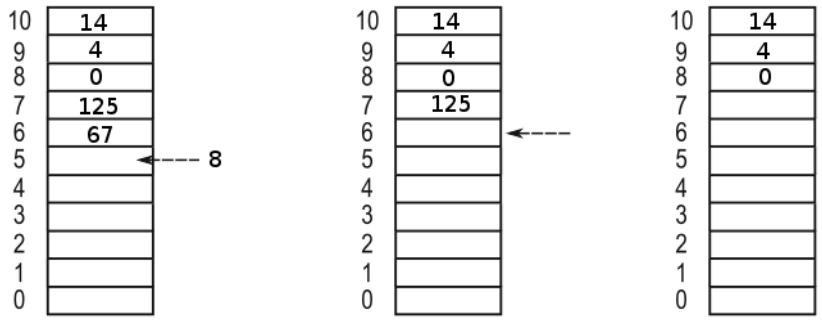
Programski brojač nije predviđen da mu se sadržaj menja direktno iz programa koji se izvršava. Ovog pravila se pridržavaju tradicionalne arhitekture, jer ne omogućavaju da PC bude dostupan kao operand instrukcija. Novije RISC arhitekture su postale malo fleksibilnije u tom smislu u pokušaju da pojednostave programiranje. Ipak, i u slučaju novijih RISC arhitektura, ova fleksibilnost treba biti korišćena oprezno kako se ne bi narušio korektan tok izvršavanja programa.

### Pokazivač steka (SP)

Stek je specijalizovana memorijska struktura koja se koristi za privremeno smeštanje podataka u specifičnom redosledu. Sama operacija smeštanja i prihvatanja podataka u skladu sa ovim redosledom upravljana je od strane CPU korišćenjem pokazivača steka. Nekolicina mikrokontrolera koristi fiksnu poziciju steka u memorijskom prostoru, dok je u većini slučajeva dozvoljeno korisniku da definiše poziciju steka u okviru RAM sekcije (ukoliko to ne uradi korisnik, automatski će se dodeliti prostor tokom procesa kompajliranja). Sadržaj pokazivača steka se odnosi na vrh stek memorije. Ova pozicija govori CPU gde se nalazi poslednje smešteni podatak. Operacija smeštanja podatka na stek se u literaturi najčešće naziva **PUSH** operacija, dok se čitanje podatka sa steka naziva **POP** operacija. Svaki put prilikom korišćenja steka, sadržaj pokazivača steka se menja.

Termin stek (engl. *stack*) je preuzet iz analogije slaganja tanjira koje se vrši u prirodnom LIFO (Last-In-First-Out) maniru: tanjir koji je poslednji stavljen na gomilu je onaj koji mora prvi biti uzet sa gomile, u suprotnom postoji realna opasnost da će tanjiri pasti i razbiti se. Upravo se iz tog razloga termin *sadržaj* steka upravo odnosi na podatak na *vrhu* steka.

Međutim, u većini implementacija, stek se popunjava na "dole" umesto na "gore". To zapravo znači da se vrednost SP smanjuje svaki put kada se na stek stavi novi podatak, a povećava se kada se podatak preuzme sa steka. SP se obično inicijalizuje na poslednju (najvišu) adresu u RAM memoriji, da bi ovakav mehanizam bio omogućen. Na primeru sa slike 5 biće ilustrovano funkcionisanje steka.




**Slika 5:** *Stek i pokazivač steka*

Prilikom inicijalizacije, stek je prazan i pokazivač pokazuje na adresu 10. Nakon postavljanja nekoliko podataka na stek (njih 5), pokazivač steka je pomeren na adresu 5. Ukoliko bi se još jedan podatak stavljao na stek, on bi bio postavljen na adresu 5 (broj 8 sa slike). Ipak, ukoliko se čita podatak sa steka (operacijom *POP*), pokazivač steka se pomera na 6, jer je poslednje pročitan podatak uklonjen sa steka (broj 67). Nakon još jednog uklanjanja, pokazivač steka se povećava na 7, nakon što se podatak sa vrha steka uklanja (broj 125).

### Statusni registar

Statusni registar u literaturi se često naziva i *Processor Status Word - PSW* registar ili *Flags* registar. On sadrži skup indikatorskih bita (engl. *flags*). Indikatorski biti su biti koji se automatski setuju/resetuju u zavisnosti od rezultata aritmetičke ili logičke instrukcije. Broj statusnih bita, kao i stanje koje je signalizirano takvim statusnim bitima su najčešće zavisni od samog mikrokontrolera, ili u opštem slučaju, centralne procesorske jedinice. Većina indikatorskih bita predstavlja stanje neposredno nakon izvršavanja instrukcije koja je poslednja izvršena od strane ALU, iako, u načelu, oni mogu biti menjani direktno iz programa. Najčešći indikatorski biti koji su implementirani na većini platformi su:

- Indikator nule (engl. *Zero Flag*) se postavlja na logičku 1 kada je rezultat operacije jednak nuli, u suprotnom se postavlja na 0.
- Indikator prenosa (engl. *Carry Flag*) se postavlja na 1 u slučajevima da je operacija dala rezultat sa prenosom (npr. sabiranjem dva 8-bitna broja je dobijen 9-bitni broj). Postoje i instrukcije koje imaju uticaj na ovaj indikator, a nisu vezane za elementarne aritmetičke operacije.
- Indikator znaka (engl. *Negative Flag, Sign Flag*) se postavlja na 1 kada je rezultat operacije negativan, dok je 0 u suprotnom.
- Indikator prekoračenja (engl. *Overflow Flag*) signalizira prekoračenje prilikom operacije sabiranja ili oduzimanja označenih brojeva (koji mogu biti kako pozitivni tako i negativni).

Osim gore navedenih indikatorskih bita, različite familije mikrokontrolera mogu imati znatno više indikatora na raspolaganju. Na slici 6 prikazan je primer postavljanja indikatorskih bita prilikom izvršenja aritmetičkih operacija.

$$\begin{array}{r} 01001010+ \\ 01111001= \\ \hline 0\ 11000011 \end{array} \qquad \begin{array}{r} 10110100+ \\ 01001100= \\ \hline 1\ 00000000 \end{array} \qquad \begin{array}{r} 10011010+ \\ 10111001= \\ \hline 1\ 01010011 \end{array} \qquad \begin{array}{r} 11001010+ \\ 00011011= \\ \hline 0\ 11100101 \end{array}$$

**Slika 6:** *Primeri elementarnih aritmetičkih operacija*

U gornjem primeru vrednosti statusnih bita su:

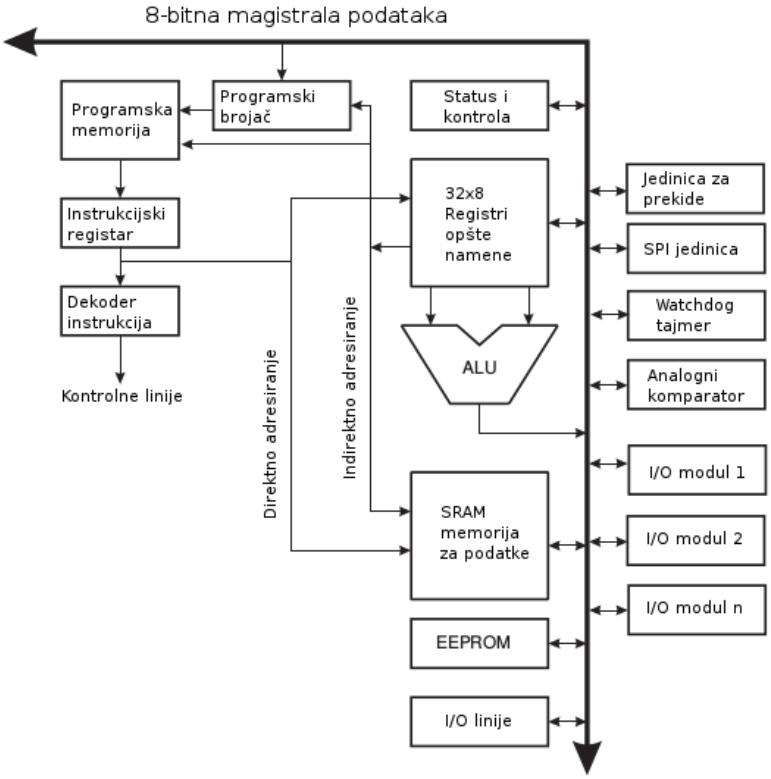
- 4Ah + 79h = C3h : C = 0, N = 1, Z = 0 i V = 1
- B4h + 4Ch = 100h : C = 1, N = 0, Z = 1 i V = 0
- 9Ah + B9h = 153h : C = 1, N = 0, Z = 0 i V = 1
- CAh + 1Bh = E5h : C = 0, N = 1, Z = 0 i V = 0

Treba obratiti pažnju na to da se prekoračenje ne dešava u slučajevima kada su oba operanda prilikom sabiranja različitog znaka, tj. kada im se razlikuju MSB (biti najvećeg značaja). N i V indikatorski biti se uglavnom odnose na označene brojeve u aritmetičkim operacijama oduzimanja i sabiranja. Indikator nule uvek pokazuje, nezavisno od operacije, da li je rezultat 0 ili ne.

## AVR arhitektura

Arhitektura AVR familije mikrokontrolera prikazana je na slici 7.

Budući da je namenjena upotrebi u *Embedded* sistemima, AVR familija mikrokontrolera koristi Harvard arhitekturu sa odvojenim memorijskim prostorima za programske instrukcije i podatake (promenljive). Instrukcije iz programske memorije izvršavaju se u protočnoj obradi prvog nivoa: dok se jedna instrukcija izvršava, naredna instrukcija se prihvata iz programske memorije. Ovaj koncept omogućava da se u svakom taktu izvrši jedna instrukcija. Programska memorija realizovana kao FLASH je memorija koja može da se reprogramira direktno u sistemu (engl. *In-System-Programmable*).




**Slika 7:** *AVR CPU arhitektura*

32 osmobitna registra opšte namene sa brzim pristupom (R0-R31) omogućavaju pristup podacima u jednom ciklusu sistemskog takta. Ovo omogućava aritmetičko-logičkoj jedinici izvršavanje operacija takođe u jednom taktu. ALU tipično preuzima dva operanda iz registara opšte namene, izvršava potrebnu operaciju i rezultat smešta nazad u registre opšte namene, sve u istoj periodi takta. ALU, osim toga, omogućava aritmetičke i logičke operacije između registara ili između registra i konstante. Takođe, operacije sa jednim registrom se mogu izvršiti od strane aritmetičko-logičke jedinice. Nakon svake aritmetičke operacije, statusni registar se ažurira kako bi adekvatno prikazao informacije u vezi sa rezultatom operacije.

Tokom prekidnih rutina i poziva pod-procedura, povratna adresa programskog brojača (PC) se smešta na stek. Stek se nalazi u okviru SRAM memorije za podatke, te je samim tim njegov

kapacitet ograničen samo kapacitetom SRAM memorije. Svi korisnički programi moraju inicijalizovati pokazivač steka (SP) tokom reset rutine (pre nego što se prva pod-procedura ili prekidna rutina izvrše).

Statusni registar (SREG) sadrži informacije u vezi sa rezultatom poslednje izvršene aritmetičke instrukcije. Ova informacija se može iskoristiti da promeni tok programa korišćenjem kondicionalnih operacija. Sadržaj statusnog registra se ne čuva automatski prilikom ulaska u prekidnu rutinu (niti se restaurira nakon završetka iste), te ovo mora da se obavlja iz korisničkog programa. Statusni biti koji sačinjavaju statusni registar u AVR arhitekturi su:

- I - Bit za globalnu dozvolu prekida (engl. *Global Interrupt Enable Flag*). Postavljanje ovog bita na 0 dovodi do globalne zabrane svih prekida. Ukoliko je I=1, kontrola dozvole prekida je prepuštena individualnim bitima dozvole u okviru kontrolnih registara periferijskih jedinica.
- T - (engl. *Bit Copy Storage Flag*) koristi se za operacije sa bitima (prilikom kopiranja bita, kao izvorište i odredište).
- H - Indikator polu-prenosa (engl. *Half Carry Flag*) se koristi u nekim aritmetičkim instrukcijama, a posebno je koristan u BCD aritmetici
- S - Indikator znaka (engl. *Sign Flag*) je u svakom trenutku jednak $N \oplus V$ (gde je $\oplus$ simbol za operaciju Ekskluzivno ILI)
- V - Indikator prekoračenja komplementa dvojke (engl. *Two's Complement Overflow Flag*) omogućava aritmetiku sa komplementom dvojke.
- N - Indikator negativnog znaka (engl. *Negative Flag*) se koristi da signalizira negativan rezultat aritmetičke operacije.
- Z - Indikator nule (engl. *Zero Flag*) signalizira da je rezultat poslednje aritmetičke ili logičke operacije nula.
- C - Indikator prenosa (engl. *Carry Flag*) se koristi kao indikator prenosa (odnosno deveti bit) u aritmetičkim i logičkim operacijama.