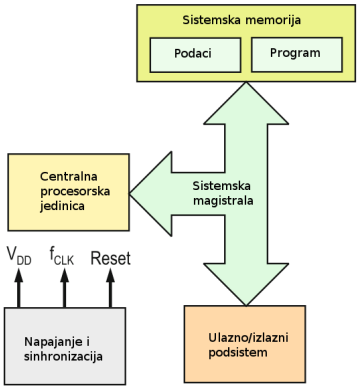
Arhitektura procesora

Minimalan skup komponenti od kojih je sastavljen neki računarski sistem, naziva se *mikroračunar*. U okviru ovog predavanja upoznaćemo se sa osnovnom strukturom mikroračunarskog sistema, nakon čega ćemo više pažnje posvetiti arhitekturi procesora. Na kraju predavanja, osvrnućemo se na AVR arhitekturu Atmel mikrokontrolera.

1 Uvod - osnovna struktura mikroračunarskog sistema

Minimalna hardverska konfiguracija mikroračunarskog sistema se sastoji od tri fundamentalne komponente: centralne procesorske jedinice (eng. Central Processing Unit - CPU), sistemske memorije i nekog oblika ulazno/izlaznog podsistema. Ove komponente su povezane višestrukim linijama grupisanim u skladu sa njihovim funkcionalnostima. Linije koje povezuju osnovne elemente mikroračunarskog sistema se nazivaju *sistemske magistrale* (eng. *system buses*). Kao dodatak, osim tri osnovna elementa mikroračunarskog sistema, dodatne komponente obezbeđuju neophodno napajanje sistema ali i vremensku sinhronizaciju neophodnu za normalnu funkcionalnost sistema. Na slici 1 može se videti osnovna arhitektura mikroračunarskog sistema.



Slika 1: *Osnovna arhitektura mikroračunarskog sistema*

Komponente mikroračunarskog sistema mogu biti implementirane na najrazličitije načine. Jedna implementacija podrazumeva da su svi elementi mikroračunarskog sistema nezavisne komponente

(čipovi), dok bi alternativno rešenje bilo integracija svih komponenti u okviru jednog čipa, 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) čini jezgro svakog mikroračunarskog sistema. CPU prima instrukcije 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. Dva tipa memorijskih elementata se najčešće mogu pronaći u okviru sistema: programska memorija i memorija podataka. Programska memorija ima ulogu da skladišti programe u formi sekvence instrukcija koje će se izvršavati. Ovi programi određuju funkcionalnost celokupnog sistema. Memorija za podatke ima ulogu da skladišti podatke koji se koriste tokom izvršavanja programa.

3. Ulazno/Izlazni podsistem se često naziva i Periferijski podsistem i on uključuje sve komponente, tj. periferije koje omogućavaju centralnom procesoru da razmenjuje informacije sa drugim komponentama sistema i spoljašnim svetom.

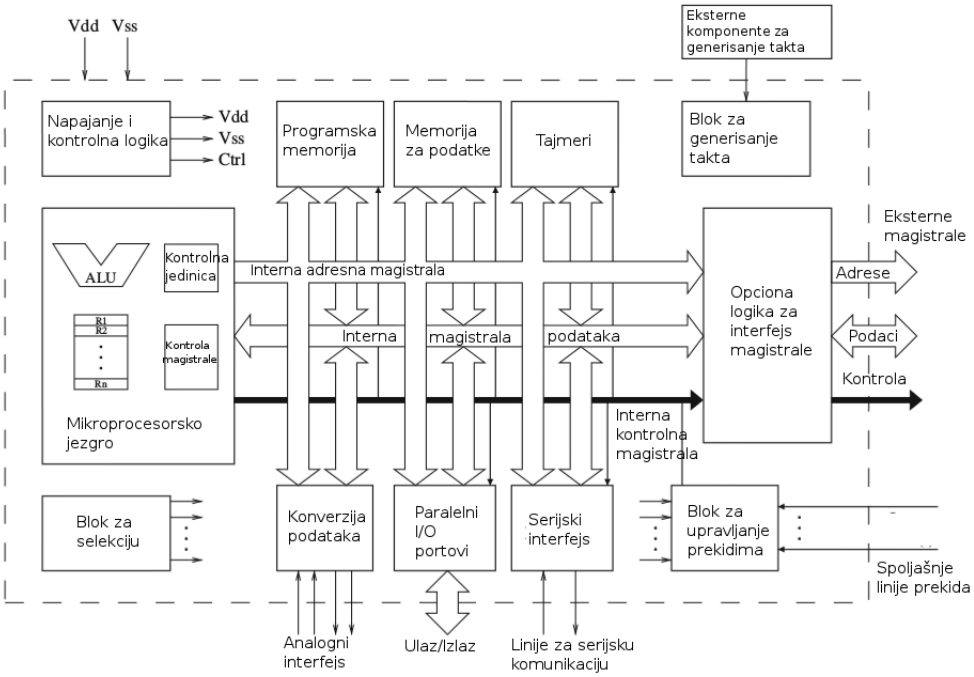
4. Sistemske magistrale su skup linija koje povezuju CPU, memoriju i ulazno/izlazni podsistem. Grupe ovih linija sistema imaju različite funkcije unutar sistema pa se one mogu podeliti na adresnu magistralu, magistralu podataka i kontrolnu magistralu.

1.1 Razlike mikrokontrolera i mikroprocesora

Mikroprocesorska jedinica (ili skraćeno mikroprocesor, eng. MPU) sadrži centralnu procesorsku jedinicu opšte namene. Da bi se dizajnirao mikroračunarski sistem baziran na mikroprocesoru, svi elementi sistema prikazani na slici 1 (magistrale, memorije i periferijske jedinice) moraju se povezati kao eksterne komponente. Osim toga, karakteristike mikroprocesora su i:

- optimizovana arhitektura u cilju preuzimanja programa i podataka iz eksterne memorije (skrivena memorija, eng. *cache*);
- mogućnost obrađivanja više instrukcija istovremeno;
- predviđanje grananja (eng. *branch prediction*);
- postojanje numeričkih ko-procesora, itd.

Najtipičniji primeri mikroračunarskog sistema baziranog na mikroprocesorima su personalni računari (eng. *Personal Computer*, PC) kao i takozvani *mainframe* računari. Najpoznatiji proizvođači mikroprocesora su Intel, Freescale, Zilog, Fujitsu, Siemens 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 10um tehnologiji, koristio je takt na 400kHz i bio je sačinjen od 2250 tranzistora. Intel-ov Xeon



Slika 2: *Arhitektura mikrokontrolera*

E7 mikroprocesor, predstavljen 2011. godine, napravljen je u 32nm tehnologiji, koristi takt od 2GHz i sadrži oko $2.6 \text{x} 10^9$ tranzistora.

Mikrokontroleri (skraćeno MCU od engleskog *Micro Controller Unit*), bazirani su na jezgru mikroprocesora, odnosno centralnoj procesorskoj jedinici (CPU), uglavnom manje složenosti u poređenju sa mikroprocesorom. Ovakvom jednom CPU je dodata memorija (kako programska tako i memorija za podatke) i nekoliko tipova periferijskih jedinica. Svi ovi elementi su integrisani u jedno integrisano kolo (čip) i nazivaju se mikrokontroler.

Ovakva struktura mikrokontrolera omogućava fleksibilnost i minimalan broj eksternih komponenti u cilju implementacije kompletnog sistema. Brojači i tajmeri, ulazno/izlazni portovi, blokovi za obradu prekida i konvertori podataka (analogno/digitalni i digitalno/analogni) uglavnom spadaju u periferijske jedinice koje su sadržane u većini savremenih mikrokontrolera. Na slici 2 prikazana je tipična arhitektura mikrokontrolera.

Iako mikrokontroleri imaju vrlo slične karakteristike kao i mikroprocesori opšte namene, znatno su 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 dostupan ogroman broj najrazličitijih mikrokontrolera, od preko hiljadu različitih proizvođača. Jedan od njih je Atmel, a njihova AVR familija u koju spada i mikrokontroler ATmega328P koji ćemo mi koristiti u okviru nastave, je najpopularnija familija mikrokontrolera današnjice.

1.2 RISC i CISC arhitektura

Mikroračunarski sistem služi za izvršavanje softvera koji je podržan od strane hardverske arhitekture sistema. Mikroračunarski sistemi se uglavnom optimizuju tako da su optimizovani ili sa stanovišta softvera, ili sa stanovišta 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);
- mala veličina programa i
- instrukcije koje se izvršavaju tipično u nekoliko sukcesivnih perioda sistemskog takta.

CISC arhitektura ima za cilj izvršavanje što više operacija u okviru svake instrukcije, u cilju generisanja jednostavnih programa (često prihvat iz memorije, izvršavanje aritmetičke ili logičke operacije i smeštanje rezultata nazad u memoriju).

RISC (*Reduced Instruction Set Computing*) mašine, sa druge strane, su dizajnirane sa fokusom na jednostavnim instrukcijama, čak i po cenu veće veličine 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.

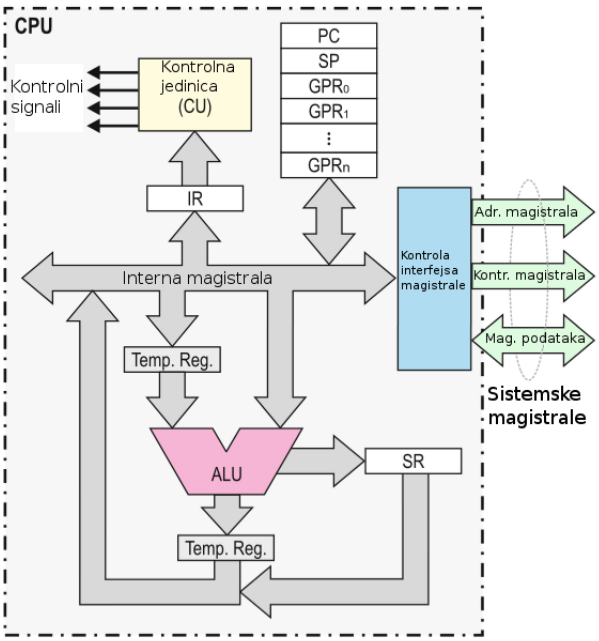
2 Centralna procesorska jedinica (CPU)

Centralna procesorska jedinica je jezgro mikroračunarskog sistema. CPU je zadužen za izvršavanje instrukcija, odnosno njegova uloga je da pretvara instrukcije u signale i akcije izvršavane na hardveru mikroračunarskog sistema. Minimalan skup komponenti koje definišu arhitekturu centralne procesorske jedinice su:

- Hardverske komponente
  - Aritmetičko-Logička jedinica (eng. *Arithmetic Logic Unit*, ALU)
  - Kontrolna jedinica (eng. *Control Unit*, CU)
  - Skup registara
  - Kontrola interfejsa magistrale
- Softverske komponente
  - Set instrukcija
  - Modovi adresiranja

Instrukcije i modovi adresiranja su determinisani i definisani specifičnostima hardverskih jedinica ALU i CU. Ovde ćemo više pažnje posvetiti hardverskoj strukturi centralne procesorske jedinice.

Slika 3 prikazuje pojednostavljeni model 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



Slika 3: *Centralna procesorska jedinica*

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.

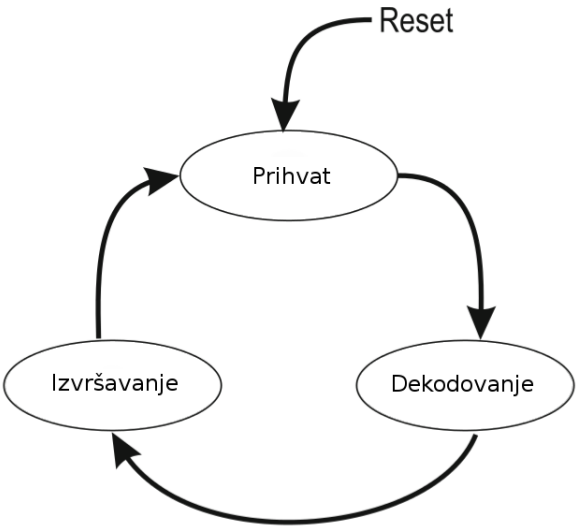
Hardverske komponente u okviru CPU koje izvršavaju operacije na podacima su ALU, interna magistrala podataka, ali i ostali funkcionalni blokovi kao što su jedinica za aritmetiku u pokretnom zarezu (eng. floating-point unit), hardverski množači itd. Hardverske komponente koje vrše operacije vezane za kontrolu sistema nazivaju se Kontrolna jedinica (eng. Control Unit). Jedinica za kontrolu magistrale, kao i komponente za sinhronizaciju se takođe najčešće posmatraju kao deo Kontrolne jedinice.

2.1 Kontrolna jedinica (CU)

Kontrolna jedinica (CU) upravlja funkcionalnošću centralne procesorske jedinice, implementirajući konačni automat (eng. *Finite State Machine* tj. FSM) koji ciklično ponavlja tri stanja: Prihvat (eng. *Fetch*), Dekodovanje (eng. *Decode*) i Izvršavanje (eng. *Execute*), kao što je prikazano na slici 4. Ova tri stanja predstavljaju elementarne operacije koje se koriste 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
- izvršavanje instrukcije.

Prihvati-dekoduj-izvrši ciklus se često u literaturi naziva i *instrukcijski ciklus* (eng. *instruction cycle* ili *CPU cycle*). Kompletan instrukcijski ciklus tipično zahteva nekoliko taktova procesora da



Slika 4: *Konačni automat Kontrolne jedinice*

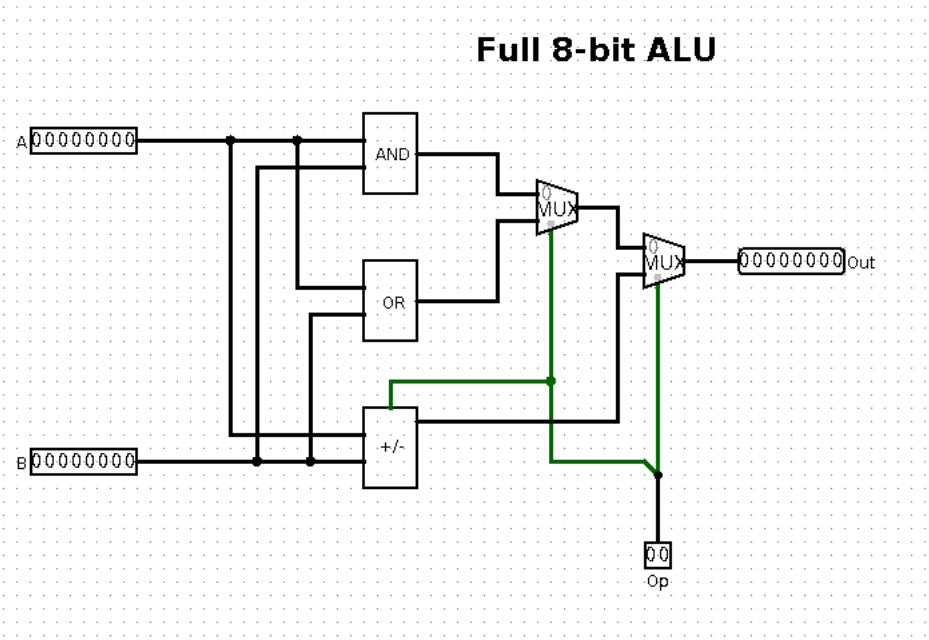
se izvrši, u zavisnosti od same instrukcije i operanada koji se u njoj koriste (konstante, varijable iz memorije ili sadržaji lokalnih registara). Neretko se dešava da CPU ima svoj interni takt koji je više učestanosti od sistemskog takta (čak i do četiri puta više). U takvim slučajevima, u literaturi 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, eng. *Program Counter*) i IR (instrukcijski registar, eng. I*nstruction Register*). Kompletan ciklus se može opisati kao:

1. Stanje Prihvata instrukcije: tokom stanja prihvatanja nove instrukcije, korišćenjem bloka kontrole interfejsa magistrale (eng. *Bus Interface Logic - BIL*), instrukcija iz memorije stiže do CPU. Programski brojač (PC) obezbeđuje adresu u memoriji sa koje se instrukcija čita. Novo pročitana 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 signali ka odgovarajućim 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 instrukciju koja će se izvršiti u narednom instrukcijskom ciklusu.

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



Slika 5: *Jednostavna aritmetičko-logička jedinica*

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 ćemo više pričati kasnije.

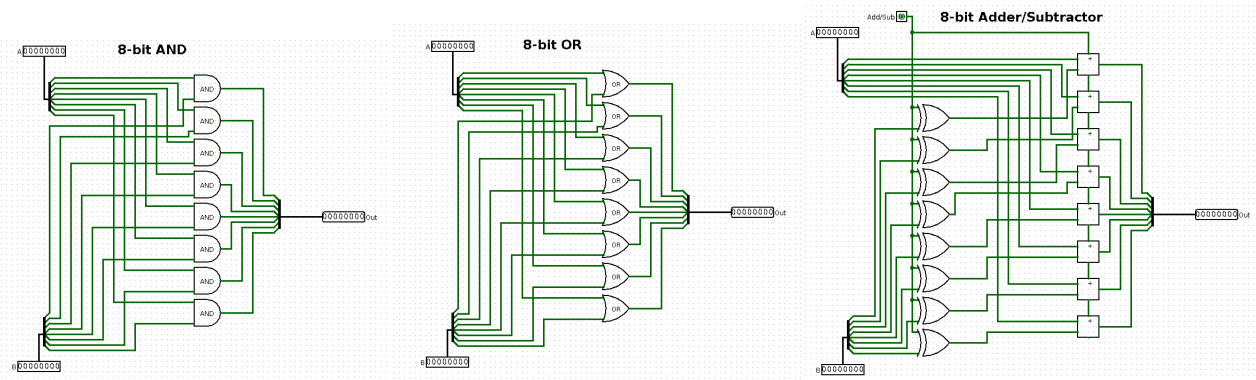
Obzirom na to da je Kontrolna jedinica realizovana kao konačan automat, neophodan je Reset signal kako bi se inicirao ciklus na početku izršavanja prve istrukcije. U cilju preuzimanja prve instrukcije iz memorije, sadržaj programskog brojača nakon reseta je uvek takav da pokazuje na prvu instrukciju koja treba da se izvršava (u slučaju mikrokontrolera, to je uglavnom adresa 0). Adresa prve instrukcije se često naziva *reset vektor*.

2.2 Aritmetičko Logička jedinica (ALU)

Aritmetičko logička jedinica je CPU komponenta 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 i već postojećih elementarnih aritmetičkih operacija) ili korišćenjem dodatnih periferijskih jedinica, kao što je na primer 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



Slika 6: *AND, OR i sabirač/oduzimač koji se koriste unutar ALU*

treba da se izvrši, prosleđuje joj promenljive nad kojima se vrši operacija, i obezbeđuje prostor gde će biti sačuvan rezultat. Kapacitet ALU je određen 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 veličina registara koji se koriste.

Primer aritmetičko logičke jedinice je prikazan na slici 5. Ova jednostavna ALU ima mogućnost izvršavanja jedne od četiri operacije: sabiranje, oduzimanje, logičko AND i logičko OR. Ulaz *op* određuje koja će se operacija izvršavati u skladu sa tabelom:

op	Operacija
00	AND
01	sabiranje
10	OR
11	oduzimanje

Slika 6 prikazuje strukturu kombinacionih mreža koje se koriste za izvršavanja aritmetičkih i logičkih operacija u okviru ALU.

2.3 Kontrola interfejsa magistrala

Kontrola interfejsa magistrala je struktura u okviru centralno procesorske jedinice koja koordinira interakciju između internih magistrala i sistemskih magistrala. Ovaj blok definiše način na koji eksterne adrese, podaci i kontrolna magistrala funkcionišu. 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 (eng. *bridges*) i hardverski moduli za arbitriranje na magistralama.

2.4 Registri

CPU registri omogućavaju privremeno smeštanje podataka, memorijskih adresa i kontrolnih informacija na način da im se može brzo i jednostavno pristupiti. Oni predstavljaju najbržu memoriju u mikroračunarskom sistemu, ali sa druge strane imaju i najmanji kapacitet. Sadržaj

registara u okviru CPU se gubi nakon nestanka 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. Najosnovniji registri koji su sadržani u okviru CPU strukture su:

- Instrukcijski registar (IR)
- Programski brojač (PC) koji se često naziva i Instrukcijski pokazivač (eng. *Instruction Pointer*-IP)
- Pokazivač steka (eng. *Stack Pointer*-SP)
- Statusni registar (eng. *Status Register*-SR)

2.4.1 Instrukcijski registar (IR)

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

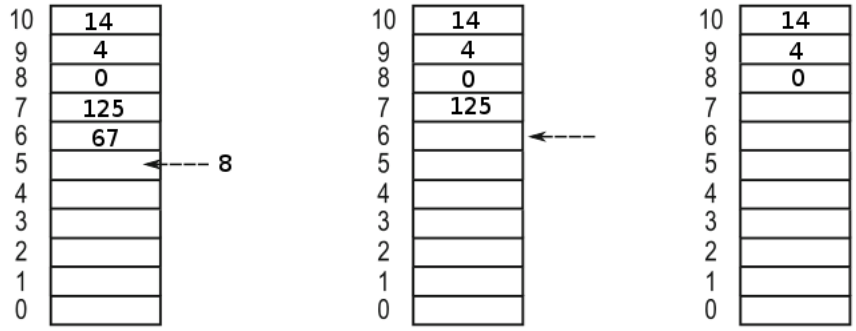
2.4.2 Programski brojač (PC)

Ovaj registar čuva adresu instrukcije koja će biti prihvaćena iz 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 iz memorije. 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 veličinom 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.

2.4.3 Pokazivač steka (SP)

Stek je specijalizovan memorijski segment koji 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



Slika 7: *Stek i pokazivač steka*

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.

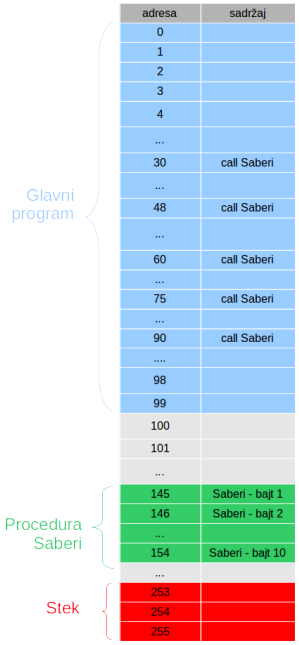
Termin stek (eng. *stack*) je preuzet iz analogije slaganja tanjira koje se vrši u prirodnom LIFO (*Last-In-First-Out*) maniru: tanjir koji je poslednji stavljen na gomilu je onaj koji mora prvi biti uzet sa gomile, u suprotnom će vrlo verovatno tanjiri pasti i razbiti se. Upravo se iz tog razloga *sadržaj*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

Međutim, u većini implementacija, popunjavanjem steka, on raste 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 podigne sa steka. SP se obično inicijalizuje na poslednju (najvišu) adresu u RAM memoriji, da bi ovakav mehanizam bio omogućen. Na primeru sa slike 7 ćemo videti kako funkcioniše stek.

Prilikom inicijalizacije, stek je prazan i pokazivač pokazuje na adresu 10. Nakon postavljanja nekoliko podataka na stek (njih 5), pokazivač steka je pomeren na 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).

Osnovna uloga steka je kontrola toka programa u slučaju pozivanja procedura (funkcija kako se one nazivaju u nekim programskim jezicima). Procedure (funkcije) su skupovi instrukcija grupisani u celine, u cilju smanjenja ukupne veličine programa. Da bismo ilustrovali problem, posmatramo sledeći primer: ukupna veličina memorije je 256 bajtova, glavni program je veličine 100 bajtova, za sabiranje dva broja potrebno je 10 bajtova i neka je tokom izvršavanja glavnog programa potrebno 5 puta sabrati dva broja (Slika 8).

Ukoliko ne bismo koristili procedure, samo za ovakvo sabiranje bi nam bilo potrebno 50 bajtova. Ako, pak, definišemo proceduru za sabiranje brojeva, i nju samo pozivamo kad god želimo da saberemo dva broja, potrebna memorija u tom slučaju je samo 10 bajtova (ako se zanemari veličina same instrukcije za poziv procedure, koja je uglavnom zanemarljiva u odnosu na veličinu same procedure). U primeru sa slike 8, glavni program počinje na adresi 0 i završava se na adresi 99 (ukupna veličina glavnog programa je 100 bajtova), dok procedura Saberi počinje na adresi 145. Stek se nalazi na dnu memorijskog prostora i popunjava se na "gore".



Slika 8: *Sadržaj memorije u primeru poziva procedure Saberi*

Problem koji ovde nastaje se odnosi na mesto povratka nakon što se izvrši procedura, jer procesor nakon što izvrši proceduru do kraja, mora da se vrati na ono mesto sa koga je procedura bila pozvana. Kao što je rečeno, ovo se dešava na 5 različitih mesta u našem programu, i procesor svaki put nakon izvršenja procedure mora da zna na koju lokaciju u memoriji mora da se vrati kako bi nastavio sa radom. Tok izvršavanja programa koji poziva proceduru Saberi prikazan je u sledećim koracima:

1. Na početku izvršavanja glavnog programa sadržaj programskog brojača je PC=0 i program počinje da se izvršava počevši sa adresom 0;

2. PC se uvećava nakon izvršenja svake instrukcije i nakon određenog vremena dođe do 30. Na adresi 30 se nalazi prvi poziv procedure Saberi. U ovom trenutku pokazivač steka ima sadržaj 255 (0xFF) što znači da je stek prazan;

3. Da bi procesor znao gde treba da se vrati nakon što se izvrši procedura Saberi, na vrh steka se stavlja adresa povratka, koja je jednaka trenutnoj vrednosti PC uvećanoj za 1. Dakle, nakon ovog koraka na lokaciju na koju pokazuje pokazivač steka se upisuje broj 31 (slika 9 levo);

4. Pokazivač steka se umanjuje za 1 kako bi pokazivao na sledeću lokaciju na koju će se upisati sledeći podatak na stek (SP=0xFE, tj SP=254);

5. U PC se upisuje adresa početka procedure Saberi, koja je jednaka 145 (0x91);

6. Pošto PC uvek pokazuje na instrukciju koja će sledeća da se izvrši, nakon ovoga će krenuti da se izvršava prva instrukcija iz procedure Saberi;

7. PC se uvećava nakon izvršenja svake instrukcije, slično kao u koraku 2. Kada dođe do kraja procedure, imaće vrednost PC=154 (0x9A);

Slika 9: *Stavljanje povratne adrese na vrh steka (levo) i preuzimanje adrese sa steka (desno)*

8. U ovom trenutku izvršavanje treba da se vrati na mesto u glavnom programu gde je bilo prekinuto pozivom procedure Saberi, a to će se desiti tako što se sa vrha steka podigne adresa koja je prethodno bila upisana i ta vrednost se upiše u PC. Da bi se ovo uradilo potrebno je najpre uvećati vrednost pokazivača steka za 1 kako bismo se vratili na poslednji upisani podatak: SP = 0xFF (slika 9 desno);

9. Vrednost sa adrese na koju pokazuje SP se upisuje u PC: PC = 31 (0x1F);

10. Nakon ovoga, stek je ponovo prazan, SP = 0xFF, a naredna instrukcija koja će se izvršavati je prva instrukcija koja sledi nakon prvog poziva procedure Saberi, tj. instrukcija koja počinje na adresi 31 (0x1F);

11. Na sličan način, kada sledeći put bude bila pozvana procedura Saberi (pozivom sa adrese 48) mikroprocesor će znati da, kada se završi procedura, treba da se vrati na adresu 49, kada se pozove sa adrese 60 da se vrati po završetku na adresu 61 itd...

Stek omogućava ugnježđavanje poziva procedura. Na primer, stek bi omogućio pozivanje neke druge procedure (npr. Oduzmi) tokom izvršavanja procedure Saberi. U tom slučaju nakon poziva procedure Oduzmi iz procedure Saberi, na steku bi se nalazile dve povratne adrese:

1. na vrhu steka bi bila adresa koja se odnosi na adresu unutar procedure Saberi gde se treba vratiti nakon izvršenja procedure Oduzmi;

2. ispod nje je druga adresa koja predstavlja adresu u okviru glavnog programa, na koju se treba vratiti kada se izvrši do kraja procedura Saberi.

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

Slika 10: *Primeri elementarnih aritmetičkih operacija*

2.4.4 Statusni registar (SR)

Statusni registar na engleskom se često naziva i *Processor Status Word* (PSW) registar ili *Flag* registar i on sadrži skup indikatorskih bita (eng. flags). Indikatorski biti su zapravo biti koji predstavljaju specifično stanje u kome se nalazi procesor.

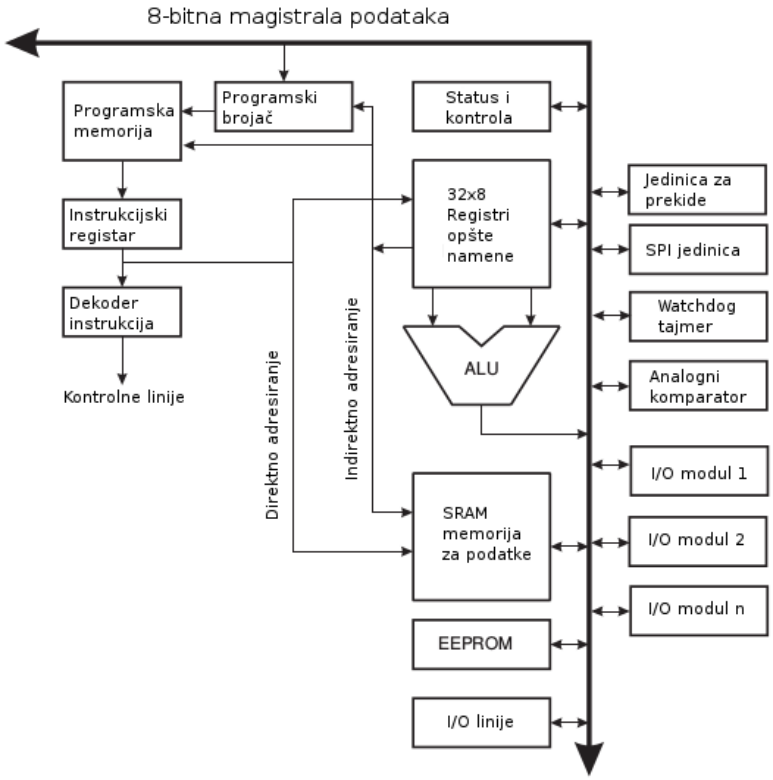
Broj statusnih bita kao i uslovi koji su signalizirani takvim statusnim bitima su najčešće zavisni od samog mikrokontrolera ili, u opštem slučaju, centralne procesorske jedinice. Većina indikatorskih bita predstavlja stanje neposredno nakon izvršavanja instrukcije koja je poslednja izvršena od strane ALU, iako, u načelu, oni mogu biti menjani direktno iz programa. Najčešći indikatorski biti koji se mogu naći u slučaju većine platformi su:

- Indikator nule (Zero flag-ZF) se postavlja na logičko 1 kada je rezultat ALU jednak nuli, u suprotnom je postavljen na 0
- Indikator prenosa (Carry flag-CF) je postavljen na 1 u slučajevima da je ALU operacija proizvela 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 (Negative flag ili Sign flag, NF) se postavlja na 1 kada je rezultat ALU operacije negativan, dok je 0 u suprotnom.
- Indikator prekoračenja (Overflow flag-VF) signalizira prekoračenje prilikom operacije sabiranja ili oduzimanja označenih brojeva (koji mogu biti kako pozitivni tako i negativni)
- Indikator prekida (Interrupt flag-IF) služi da dozvoli/zabrani zaustavljanje programa nakon pojave eksternog događaja (prekida).

Osim gore navedenih indikatorskih bita, različite familije mikrokontrolera mogu imati znatno više indikatora na raspolaganju. Na slici 10 je prikazan primer postavljanja indikatorskih bita prilikom 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



Slika 11: *AVR CPU arhitektura*

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

3 AVR CPU arhitektura

Arhitektura AVR familije mikrokontrolera je prikazana na slici 11.

U cilju maksimizovanja performansi i povećanja paralelizma, AVR familija mikrokontrolera koristi Harvard arhitekturu, sa odvojenim memorijskim prostorom i magistralama za programsku i memoriju podataka. Instrukcije iz programske memorije se izvršavaju 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 je realizovana kao fleš memorija koja može da se reprogramira direktno u sistemu (eng. In-System-Programmable).

32 8-bitna registra opšte namene sa brzim pristupom omogućavaju pristup podacima u jednom ciklusu sistemskog takta. Ovo omogućava aritmetičko-logičkoj jedinici izvršavanje 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 jednom taktu. 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 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 (eng. Global Interrupt Enable bit). Ovaj statusni bit se koristi da zabrani/dozvoli sve prekide
- T - Bit Copy Storage se koristi za operacije sa bitima (prilikom kopiranja bita, kao izvorište i odredište)
- H - Indikator polu-prenosa (eng. Half Carry bit) se koristi u nekim aritmetičkim instrukcijama, a posebno je koristan u BCD aritmetici
- S - Indikator znaka (eng. Sign bit) je u svakom trenutku jednak $N \oplus V$ (gde je $\oplus$ simbol za operaciju Ekskluzivno ILI)
- V - Indikator prekoračenja komplementa dvojke (eng. Two's Complement Overflow Flag) omogućava aritmetiku sa komplementom dvojke
- N - Indikator negativnog (eng. Negative Flag) se koristi da signalizira negativan rezultat aritmetičke ili logičke operacije
- Z - Indikator nule (eng. Zero Flag) signalizira da je rezultat poslednje aritmetičke ili logičke operacije nula
- C - Indikator prenosa (eng. Carry Flag) se koristi kao prenos u aritmetičkim i logičkim operacijama.