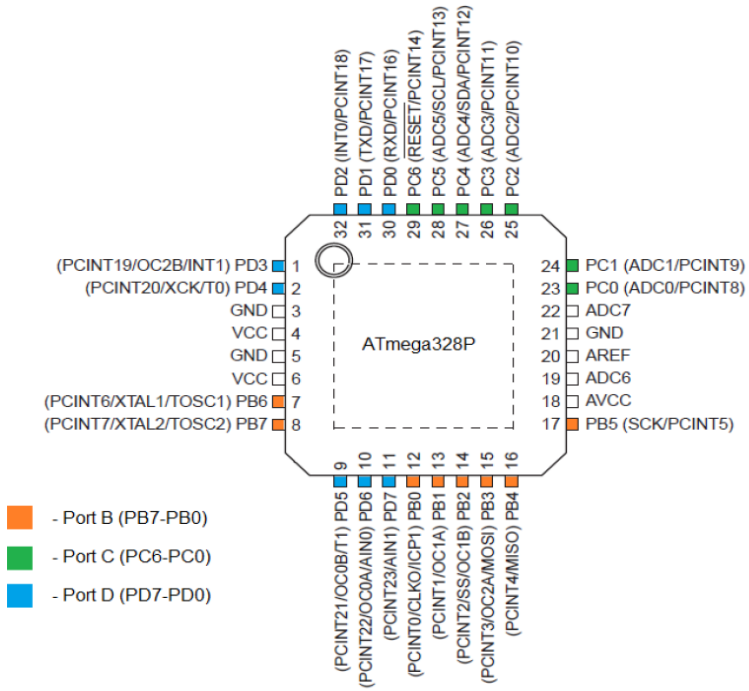
# Portovi i sistem prekida

# Portovi mikrokontrolera

• Najveći broj nožica (pinova) na kućištu mikrokontrolera predstavlja ulazno-izlazne priključke namenjene povezivanju mikrokontrolera sa periferijskim uređajima.

• Svaki od ovih pinova može da služi kao digitalni ulaz/izlaz opšte namene (engl. **GPIO** = ***General Purpose Input/Output*** ). Pored toga, pinovima mogu biti dodeljene i alternativne funkcionalnosti kao što su linije za slanje/prijem serijskog porta, ulazi A/D konvertora, PWM izlazi i sl. Na slici su obeleženi raspored i alternativne funkcije pinova mikrokontrolera ATmega328p.

• Grupe ulazno-izlaznih pinova nazivaju se **portovi**. U okviru jednog porta obično se nalazi do 8 ulazno-izlaznih priključaka.



# Upravljanje i očitavanje stanja portova

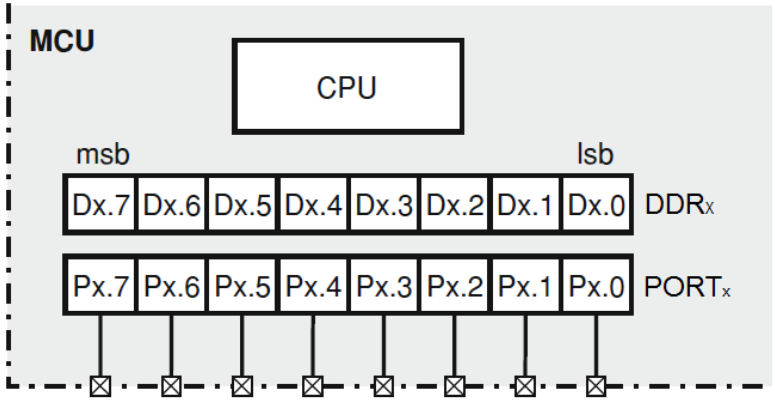
• U opštem slučaju, za upravljanje i očitavanje stanja porta **x** koriste se tri konfiguraciona registra:

- **DDR**x (engl. *Data Direction Register*) - Određuje smerove pinova porta x
- **PORT**x - Određuje stanja onih pinova porta x koji se koriste kao digitalni izlazi
- **PIN**x - Služi za očitavanje stanja onih pinova porta x koji se koriste kao digitalni ulaz

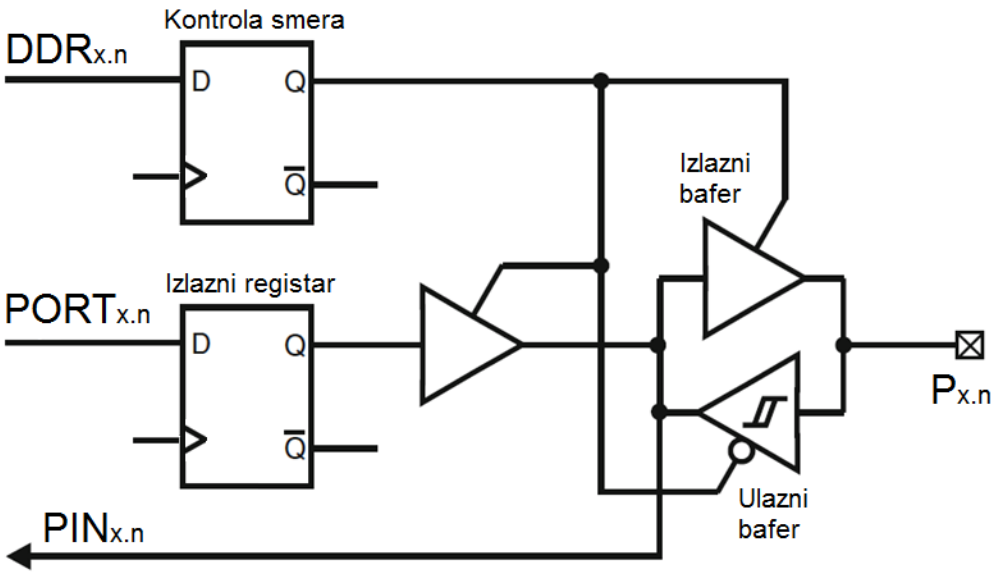
• $D_{x.n}$ (bit na poziciji n u okviru registra $DDR_x$) vrši selekciju smera pina, na sledeći način:

$$DDR_{x.n} = \begin{cases} 1, pin\ P_{x.n}\ je\ konfigurisan\ kao\ izlaz \\ 0, pin\ P_{x.n}\ je\ konfigurisan\ kao\ ulaz \end{cases}$$

• Kada je $DDR_{x.n}$ = 1, stanje pina određeno je bitom $PORT_{x.n}$ (bit na poziciji n u okviru registra $PORT_x$)
• Kada je $DDR_{x.n}$ = 0, stanje pina se očitava preko bita $PIN_{x.n}$ (bit na poziciji n u okviru registra $PIN_x$)



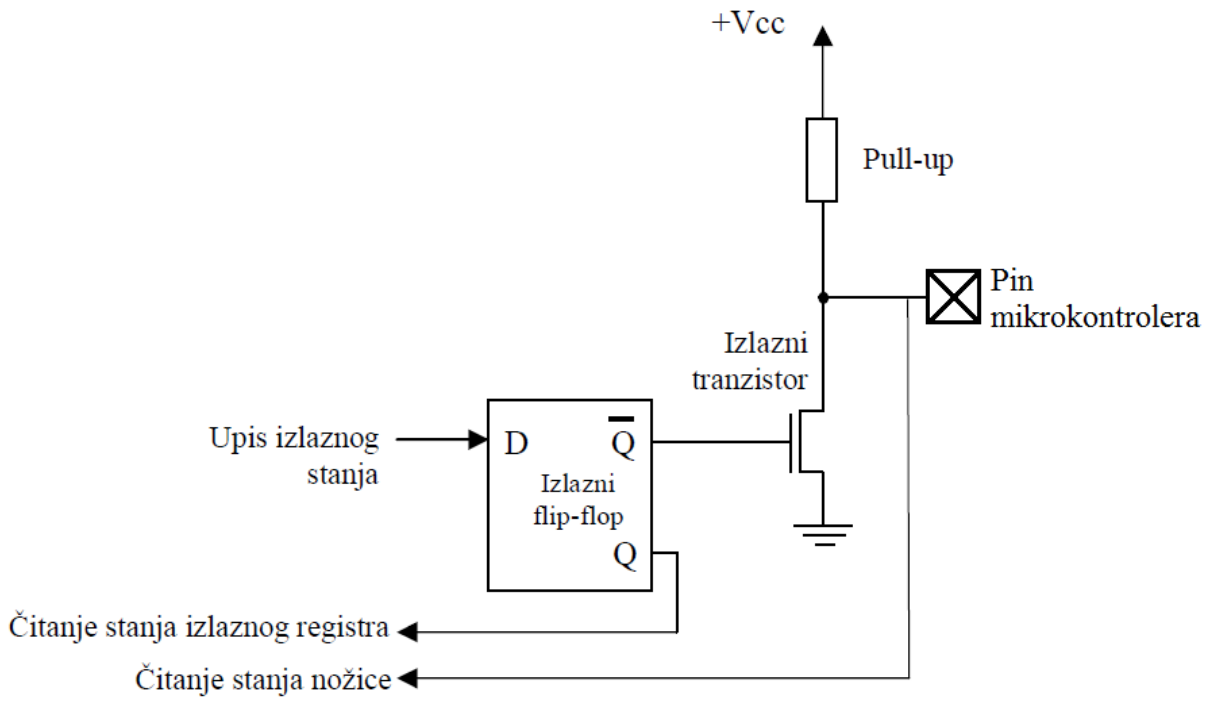
# Interna struktura GPIO pinova




Uobičajena struktura GPIO pina mikrokontrolera

- U primeru prikazanom na slici je prikazana interna struktura kontrolne logike **pina Px.n** (pin sa oznakom ‘n’ u okviru porta sa oznakom ‘x’)
- Flipflop koji pripada registru za kontrolu smera ($DDR_{x.n}$ na slici) određuje da li će pin biti korišćen kao ulaz, ili kao izlaz.
- Ukoliko se pin koristi kao izlaz, njegovo stanje određeno je stanjem flipflopa koji pripada izlaznom registru ($PORT_{x.n}$)

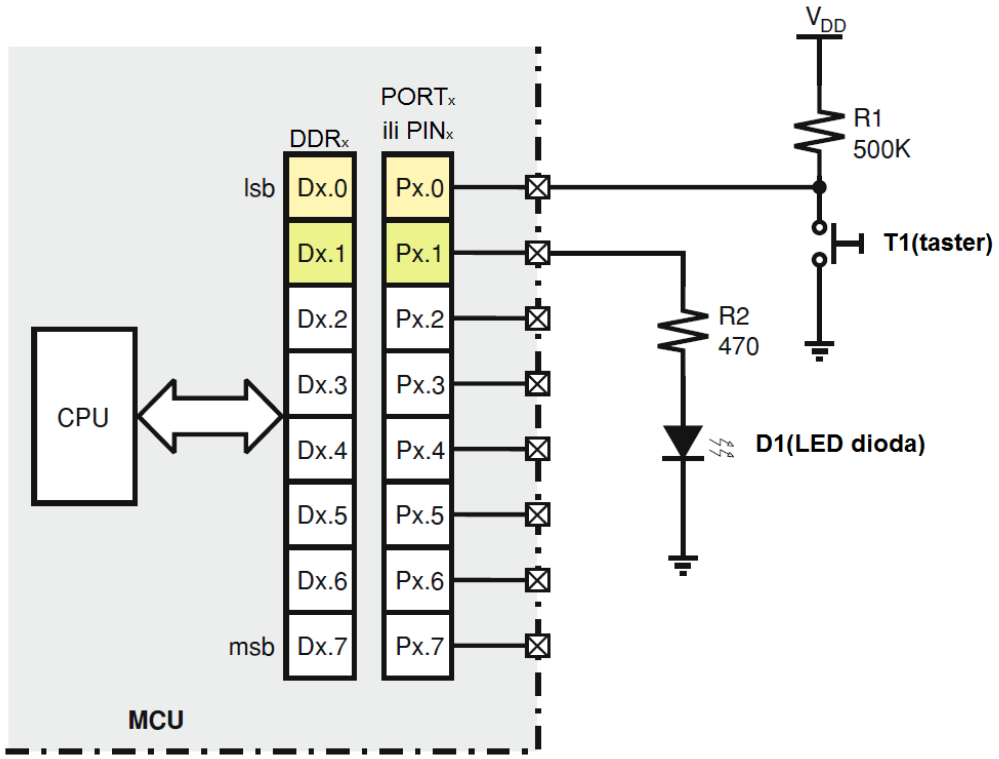
# GPIO pinovi sa open-collector (open-drain) logikom




- Pinovi mikrokontrolera sa *open-collector* logikom odlikuju se "jakom" nulom i "slabom" jedinicom: ukoliko je u odgovarajući bit **izlaznog registra upisana nula**, uključuje se izlazni tranzistor. U suprotnom, izlazni tranzistor je isključen, pa je izlazno stanje određeno pull-up otpornikom.
- Ukoliko se pin koristi kao **ulaz**, u izlazni flip flop je neophodno upisati logičku jedinicu, da bi izlazni **tranzistor bio isključen**. Na ovaj način, eksterna logika upravlja stanjem pina i može ga po potrebi oboriti na 0.

# Povezivanje portova mikrokontrolera sa tasterima i LED diodama

- U primeru prikazanom na slici, vidi se način povezivanja jednog **ulaznog uređaja (taster)** i jednog izlaznog uređaja (LED dioda) na pinove koji pripadaju istom portu (port X)



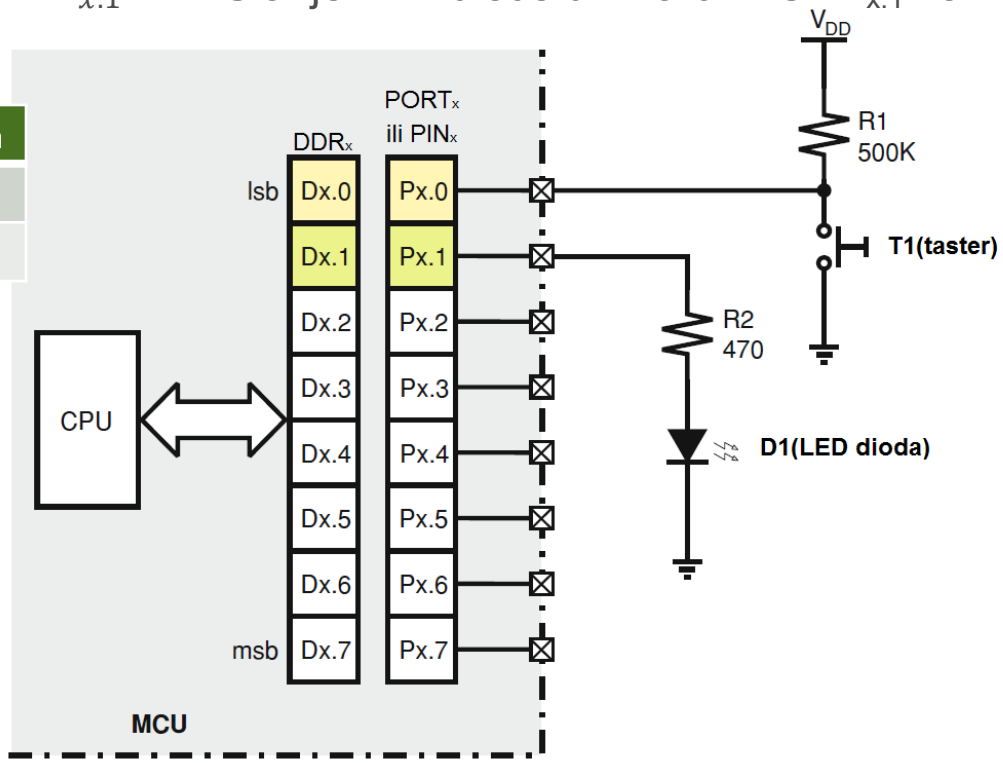
# Povezivanje portova mikrokontrolera sa tasterima i LED diodama

- U primeru prikazanom na slici, vidi se način povezivanja jednog **ulaznog uređaja (taster)** i jednog izlaznog uređaja (LED dioda) na pinove koji pripadaju istom portu (port X)
- Pin $P_{x.0}$ je ulazni ⇒ $DDR_{x.0} = 0$. Stanje tastera se čita preko bita $PIN_{x.0}$ na sledeći način:

| $PIN_{x.0}$ | Taster |
|---|---|
| 0 | Pritisnut |
| 1 | Pušten |

- Pin $P_{x.1}$ je izlazni ⇒ $DDR_{x.1} = 1$. Stanje LED diode diktira bit $PORT_{x.1}$ na sledeći način:

| $PORT_{x.1}$ | LED dioda |
|---|---|
| 0 | Isključena |
| 1 | Uključena |



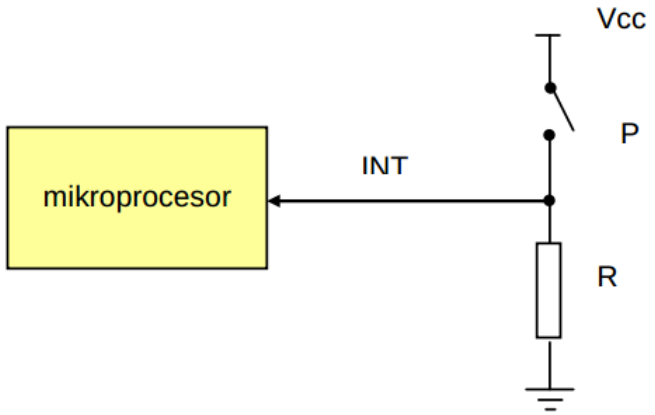
# Sistemi prekida

# Sistem prekida – osnovna ideja

- Prekidi predstavljaju mehanizam pomoću kojeg mikroprocesor, ili mikrokontroler može da **odreaguje na pojedine događaje** koji zahtevaju hitnu obradu. Ovo je pogotovo od značaja u vremenski kritičnim aplikacijama, gde vremena odziva moraju biti strikno ispoštovana.
- Podsistem prekida omogućava mikroprocesoru da pod dejstvom spoljnog ili unutrašnjeg dogadaja, prekine izvršavanje tekućeg programa i pređe na izvršavanje specijalizovanog potprograma namenjenog obradi tog dogadaja.
- U opštem slučaju, **postoje 2 vrste prekida**:
    - Hardverski (eksterni) - Izazvani od strane periferijskog uređaja sa kojim je procesor povezan. Tipično, procesor se obaveštava o zahtevu za prekidom putem specijalizovanog **elektronskog signala (INT, engl. Interrupt ).** Primeri događaja koji izazivaju zahtev za prekidom: pritisak tastera na tastaturi, pomeranje miša i sl.
    - Softverski (interni) - Izazvan specifičnim stanjem procesora (npr. deljenjem nulom), ili specijalizovanom instrukcijom koja dovodi do pojave prekida (engl. trap).
- Svakom izvoru prekida dodeljuje se zaseban potprogram za obradu. Broj hardverskih prekida ograničen je brojem linija za zahteve za prekidom (engl. **IRQ = Interrupt Request** ).
- Prekidi su uobičajeno korišćena tehnika u multitaskingu, pogotovo u aplikacijama koje rade u realnom vremenu.

# Načini reagovanja na eksterni događaj

**PRIMER**: Neka je mikroprocesor povezan sa prekidačem, kao što je prikazano na slici. Potrebno je momentalno detektovati uključenje prekidača i u tom slučaju izvršiti odgovarajuću proceduru.

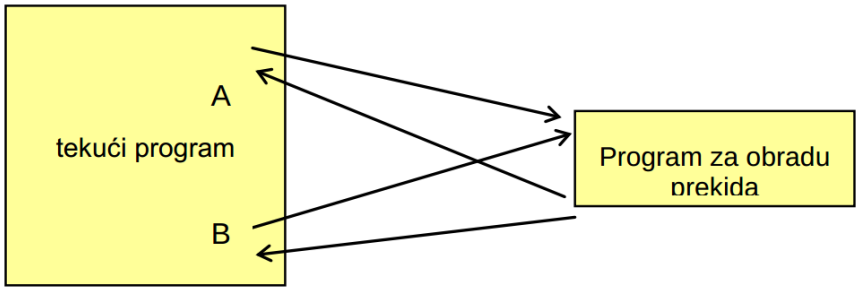



- Dva tipična pristupa rešavanju navedenog problema su:

1. Program se "vrti" u petlji tokom koje kontinualno proverava stanje ulazne linije i u slučaju detekcije logičke jedinice poziva specijalizovanu proceduru. Tokom izvršenja petlje, procesor nije u mogućnosti da obavlja neki drugi zadatak, što može znatno da umanji efikasnost sistema. Ovakav pristup se u programerskom žargonu naziva "prozivanje" (engl. polling ).
2. Pritisak tastera detektuje specijalizovana hardverska jedinica - **kontroler prekida**. U međuvremenu, procesor izvršava neki zadatak manje hitnosti. Kada se pojavi zahtev za prekidom, procesor automatski prekida **trenutnu aktivnost, odrađuje potprogram za obradu prekida,** nakon čega nastavlja sa izvršenjem programa od istog mesta gde se nalazio u trenutku pojave prekida.

# Potprogram za obradu prekida

- Potprogram za obradu prekida je deo programskog koda koji se automatski poziva po nastanku zahteva za prekidom. Naziva se još i prekidna rutina (engl. *Interrupt Service Routine* (ISR), *Interrupt Handler* ).
- Program koji procesor izvršava onda kada nema prekida, u ovom kontekstu se naziva **tekući program**. Do prekida može doći tokom različitih faza izvršenja tekućeg programa. Instrukcija tokom čijeg izvršenja je nastao zahtev za prekidom naziva se **tačka prekida**.




- Potrebno je obezbediti mehanizam koji omogućava procesoru povratak u tačku prekida nakon izvršenja **prekidne rutine**. Ovo se postiže tako što se neposredno pre skoka na početak prekidne rutine trenutna vrednost **programskog brojača (PC)** koja predstavlja povratnu adresu, postavlja na **stek**.
- Na kraju prekidne rutine, povratna adresa se automatski očitava sa steka i smešta u programski brojač.

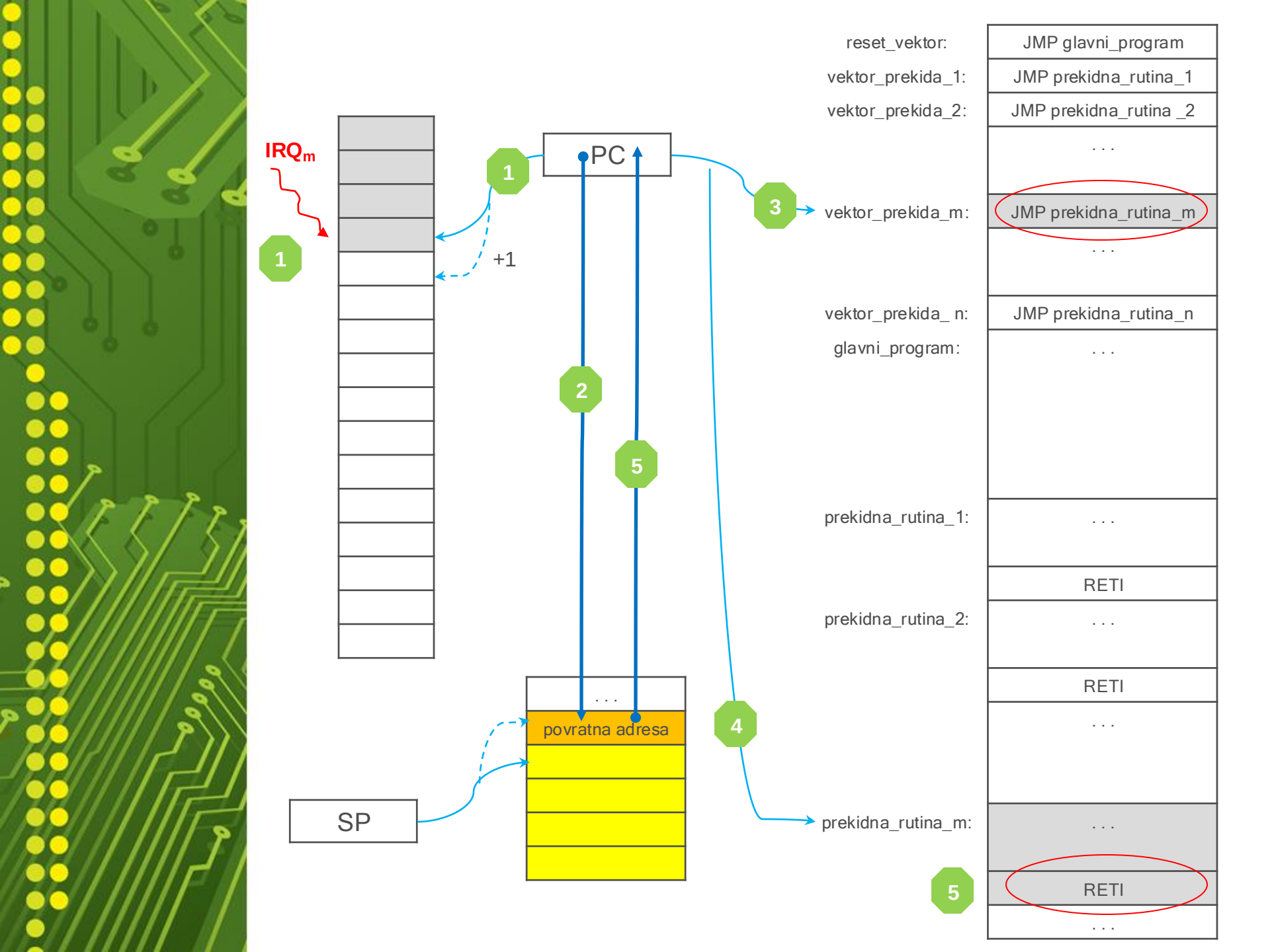
# Višestruki izvori prekida

- U praksi je čest slučaj da postoje višestruki izvori prekida kao što su eksterni signali, serijski port, tajmeri i sl.
- U slučaju višestrukih izvora prekida, podsistem za prekide mora da obezbedi:
  - Potprograme (rutine) za obradu različitih prekida
  - Prepoznavanje izvora prekida
  - Rešavanje prioriteta prekida u slučaju istovremenog aktiviranja više prekidnih signala
  - Prelazak na rutinu koja odgovara izvoru prekida
  - Postupak u slučaju da se tokom obrade jednog prekida aktivira neki drugi signal prekida
- Svakom izvoru prekida pridružena su 2 bita posebne namene, koji pripadaju specijalizovanim kontrolnim registrima:
  - **Bit za maskiranje (dozvolu) prekida**, (engl. *Interrupt Enable* ) - da bi prekid bio dozvoljen, ovaj bit mora biti setovan. Pored toga, uobičajeno je da postoji bit za globalnu zabranu prekida, čijim aktiviranjem se svi prekidi istovremeno zabranjuju.
  - **Indikator zahteva za prekidom** (engl. *Interrupt Flag* ) - Ovaj bit se automatski setuje kada se desi događaj koji zahteva prekid.

# Pozivanje potprograma za obradu prekida

- Na najnižim adresama u programskoj memoriji nalaze se tzv. vektori prekida.
- Po pojavi zahteva za prekidom **m** (u pitanju je jedan od **n** mogućih izvora prekida za dati procesor), automatski se obavljaju sledeće akcije:

1. U trenutku pojave zahteva za prekidom ($IRQ_m$), procesor izvršava instrukciju koja se nalazi na adresi na koju pokazuje programski brojač (**PC**). Procesor automatski inkrementira PC i proverava da li je prekid m dozvoljen. Ukoliko nije, izvršenje se nastavlja kao da se ništa nije desilo, a ukoliko jeste, prelazi na korak br. 2.
2. **Indikator zahteva za prekidom se automatski resetuje**, a istovremeno se setuje bit za **globalnu zabranu prekida**, čime je onemogućena obrada novog prekida sve do povratka iz prekidne rutine. Trenutna vrednost PC predstavlja **povratnu adresu**, koja se postavlja na stek.
3. U PC se upisuje **adresa vektora prekida** m.
4. Vrši se **bezuslovan skok** na početak prekidne rutine.
5. Izvršava se prekidna rutina, koja se završava instrukcijom **RETI**. Po nailasku na ovu instrukciju, povratna adresa se skida sa steka i upisuje u PC, čime se vrši povratak u glavni program. Pored toga, RETI instrukcija resetuje bit za **globalnu zabranu prekida**, čime je omogućena obrada novog prekida.


IRQm
1
1
+1
PC
2
3
4
5
5
. . .
povratna adresa
SP
reset_vektor:
JMP glavni_program
vektor_prekida_1:
JMP prekidna_rutina_1
vektor_prekida_2:
JMP prekidna rutina _2
. . .
vektor_prekida_m:
JMP prekidna_rutina_m
. . .
vektor_prekida_ n:
JMP prekidna_rutina_n
glavni_program:
. . .
prekidna_rutina_1:
. . .
RETI
prekidna_rutina_2:
. . .
RETI
. . .
prekidna_rutina_m:
. . .
RETI
. . .