Upravljačka jedinica

Mikrooperacije

- Aktivnosti koje se obavljaju u koracima svake od faza instrukcionog ciklusa se nazivaju *mikrooperacije*. Primer takvih aktivnosti je npr. manipulacija sa registrima procesora.
- Prefiks *mikro* označava da su te operacije atomske operacije u procesoru, kao i da su na vrlo niskom nivou i vrlo ograničenog efekta.
- Mikrooperacije se po funkciji koju obavljaju mogu podeliti u četiri kategorije:
    1. Prenos podataka izmedju registara.
    2. Prenos podataka iz registara ka spoljašnjem interfejsu odnosno sistemskim magistralama.
    3. Prenos podataka od spoljašnjeg interfejsa ka registrima.
    4. Izvršavanje aritmetičko/logičkih operacija pri čemu se ulaz i izlaz nalaze u registrima procesora.
- Ilustracija osnovnih funkcija mikrooperacija je prikazana na primeru jednoadresnog računara koji podržava i direktno i indirektno adresiranje.

Faza dohvatanja

U najjednostavijoj varijanti faza dohvatanja se sastoji od tri koraka i uključuje četiri mikrooperacije:

- U ovoj fazi se vrši prenošenje sledeće instrukcije koja treba da se izvrši u PRM procesora.
- Na početku ove faze adresa naredne instrukcije je u brojaču instrukcija (PC).
- U prvom koraku se adresa prenosi u RMA
- U sledećem koraku se tražena instrukcija prenosi u PRM a zatim ažurira sadržaj brojača instrukcija tako da pokazuje na narednu instrukciju koja bi trebalo da se izvrši.
- Poslednji korak u ovoj fazi je prenošenje sadržaja iz PRM u instrukcioni registar (IR).

Efekat izvršavanja svake od mikrooperacija je promena sadržaja nekog od registara. Izvršavanje svake od mikrooperacija traje najviše jednu vremensku jedinicu. Ako sa t_i označimo i-tu vremensku jedinicu, tada se niz mikrooperacija koje realizuju fazu dohvatanja može predstaviti na sledeći način:

$$
\begin{array}{lll}
t_1: & (PC) & \longrightarrow \quad RMA \\
t_2: & Memorija_{RMA} & \longrightarrow \quad PRM \\
 & (PC) + 1 & \longrightarrow \quad PC \\
t_3: & (PRM) & \longrightarrow \quad IR
\end{array}
$$

Faza indirekcije

Ciklus indirekcije (ukoliko postoji) se izvršava na sledeći način:

1. Adresni deo instrukcije se prenosi u RMA.
2. Na osnovu sadržaja RMA odredjuje se instrukcija koja se iz memorije prenosi u PRM.
3. Adresni deo IR se ažurira adresnim delom sadržaja PRM, tako da IR sada sadrži instrukciju koja koristi direktno adresiranje.

Ova tri koraka se mogu predstaviti na sledeći način:

t_1: $(IR_{adresa}) \longrightarrow RMA$
t_2: $Memorija_{RMA} \longrightarrow PRM$
t_3: $(PRM_{adresa}) \longrightarrow IR_{adresa}$

Faza izvršenja

Ako u računaru postoji n mašinskih instrukcija sa različitim kodovima operacije, tada se u fazi izvršenja mogu pojaviti n različitih nizova mikroinstrukcija.

Za ilustraciju je prikazana faza izvršavanja instrukcije sabiranja `ADD R1,L` u kojoj se sadržaj memorijske lokacije `L` dodaje na prethodni sadržaj opšteg registra `R1`. Jedan od mogućih nizova mikroinstrukcija je:

$$
\begin{array}{lll}
t_1: & (IR_{adresa}) & \longrightarrow \quad RMA \\
t_2: & Memorija_{RMA} & \longrightarrow \quad PRM \\
t_3: & (R1) + (PRM) & \longrightarrow \quad R1
\end{array}
$$

U realnim izračunavanjima biće uključene i mikrooperacije koje izdvajaju oznaku registra iz IR.

Faza obrade prekida

Osnovna akcija koju treba preduzeti je čuvanje trenutnog stanja programa koji se izvršava, i upisivanje adrese rutine koja vrši obradu prekida u brojač instrukcija. Niz mikroinstrukcija koji realizuje ove korake se može predstaviti na sledeći način:

t_1:	(PC)	$\longrightarrow$	PRM
t_2:	AdresaČuvanja	$\longrightarrow$	RMA
	AdresaRutinePrekida	$\longrightarrow$	PC
t_3:	(PRM)	$\longrightarrow$	Memorija$_{RMA}$

Registar faze

Fazu instrukcije koja se trenutno izvršava, odnosno niz mikrooperacija koji se primenjuje, odredjuje sadržaj posebnog registra upravljačke jedinice koji se naziva **registar faze**. Veličinu registra faze odredjuje broj faza u mašinskoj instrukciji.

Osnovne funkcije upravljačke jedinice

Neke od osnovnih funkcija upravljačke jedinice su:

1. Obezbedjenje korektnog redosleda izvršavanja.
2. Obezbedjenje izvršavanja mikrooperacija.
3. Obrada zahteva za podacima i instrukcijama iz keš memorije.
4. Prevodjenje realnih u virtualne adrese i obratno.
5. Implementacija sistema preklapanja i obavljanje odgovarajuće optimizacije.
6. Izbor naredne instrukcije za izvršavanje u slučaju obrade instrukcije uslovnog grananja, itd.

Kontrolni signali

Kontrolni signali predstavljaju binarne vrednosti koje se dovode direktno na logička kola upravljačke jedinice ili odvode sa logičkih kola ka ostatku procesora ili sistemskim magistralama.

Prema tipu, kontrolni signali se mogu podeliti u sledeće grupe:

- Signali koji aktiviraju neku od funkcija ALU.
- Signali koji zahtevaju prenošenje podataka izmedju registara.
- Signali koji aktiviraju odredjeni put za podatke.
- Signali koji dolaze/odlaze sa nekog od spoljašnjih interfejsa ili neke od sistemskih magistrala.

Kao ilustraciju obrade kontrolnog signala prikazaćemo kontrolne signale koji se generišu pri izvršavanju faze dohvatanja mašinske instrukcije. I u ostalim fazama se kontrolni signali generišu na sličan način. Faza dohvatanja se realizuje preko četiri mikrooperacije u tri vremenska intervala. Izvršavanje mokrooperacija je omogućeno generisanjem sledećih kontrolnih signala:

1. Kontrolni signal koji otvara odredjeno elektronsko kolo i omogućuje prenošenje bitova iz PC u RMA.
2. Upravljačka jedinica generiše sledeće kontrolne signale koji se istovremeno izvršavaju:
    - Signal koji omogućuje prenošenje sadržaja RMA na magistralu adresa.
    - Signal za magistralu kontrole koji omogućuje čitanje podataka iz memorije.
    - Signal koji otvara vrata koja omogućuju prenošenje sadržaja magistrale podataka u PRM.
    - Signal koji dodaje 1 na sadržaj PC i rezultat ponovo upisuje u PC.
3. Kontrolni signal koji omogućuje prenos iz PRM u IR. Po prenošenju podataka u IR upravljačka jedinica ispituje sadržaj IR i, u zavisnosti od postojanja zahteva za indirektnim adresiranjem, generiše signal kojim prelazi u fazu indirekcije ili u fazu izvršenja.

Implementacija upravljačke jedinice

Tehnike korišćene za implementaciju upravljačke jedinice su:

1. hardversko–ožičena implementacija (eng. *hardwired*), i
2. mikroprogramska implementacija (eng. *microprogrammed*).

Skoro svi savremeni procesori koriste mikroprogramsku implementaciju upravljačke jedinice.

Hardversko–ožičena implementacija

Iako obezbedjuje brže izvršavanje operacija u upravljačkoj jedinici, danas je praktično napušten ovakav način implementacije **kompletne** upravljačke jedinice. Karakteristike ovog načina implementacije su:

- Sve funkcije upravljačke jedinice se implementiraju pomoću logičkih elemenata.
- Logički elementi su spojeni medjusobnim vezama (“žicama”), pri čemu su svi segmenti implementacije uradjeni isključivo hardverski.
- Skup ulaznih signala u upravljačku jedinicu predstavlja ulazne signale kombinatorne ili sekvencijalne mreže koja ih transformiše u skup izlaznih kontrolnih signala. item Za svaki izlazni signal odredjuje odgovarajuća logička funkcija koja ga izvodi u zavisnosti od ulaznih signala.

Razlozi napuštanja ovakvog načina implementacije su:

1. Kompleksnost mreže koja realizuje funkcije upravljačke jedinice.
2. Nemogućnost proširenja postojeće upravljačke jedinice uvodjenjem novih funkcija.
3. Teže testiranje ispravnosti rada upravljačke jedinice.

Mikroprogramska implementacija

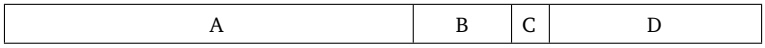
- Termin *mikroprogram* je prvi predložio Moris Vilkis 1951. godine u radu u kome je izložio novi pristup konstrukciji upravljačke jedinice.
- Ideja je prvi put implementirana kod upravljačke jedinice računara serije IBM S/360.

Mikroinstrukcije

- Svaka od mikrooperacija upravljačke jedinice se kodira simboličkom oznakom.
- Nizovi mikrooperacija se nazivaju *mikroprogrami*, a pisanje mikroprograma mikroprogramiranje.
- Mikroprograme sačinjavaju *mikroinstrukcije* koje sadrže simboličke oznake mikrooperacija i njihovih argumenata.
- Mikroprogrami se nazivaju i *firmware* ili *mikrokod*

Implementacija upravljačke jedinice preko mikroprogramiranja se zasniva na činjenici da se kao rezultat izvršavanja bilo koje od mikrooperacija generišu kontrolni signali čije vrednosti uključeno/isključeno mogu da budu predstavljene jednim bitom. Bitove koji odgovaraju svim kontrolnim linijama možemo da grupišemo u *kontrolnu reč*. Svaka od mikrooperacija može da se predstavi nekom od kombinacija bitova u kontrolnoj reči. U opštem slučaju mikroinstrukcije se prema dužini formata mogu podeliti na:

1. Horizontalne, koje sadrže više od 40 bitova.
2. Vertikalne, koje sadrže izmedju 16 i 40 bitova.



A– Interni kontrolni signali procesora
B– Kontrolni signali za sistemske magistrale
C– Grananje (bezuslovno, nula, prekoračenje, indirekcija)
D– Adresa mikroinstrukcije

a) Horizontalna mikroinstrukcija

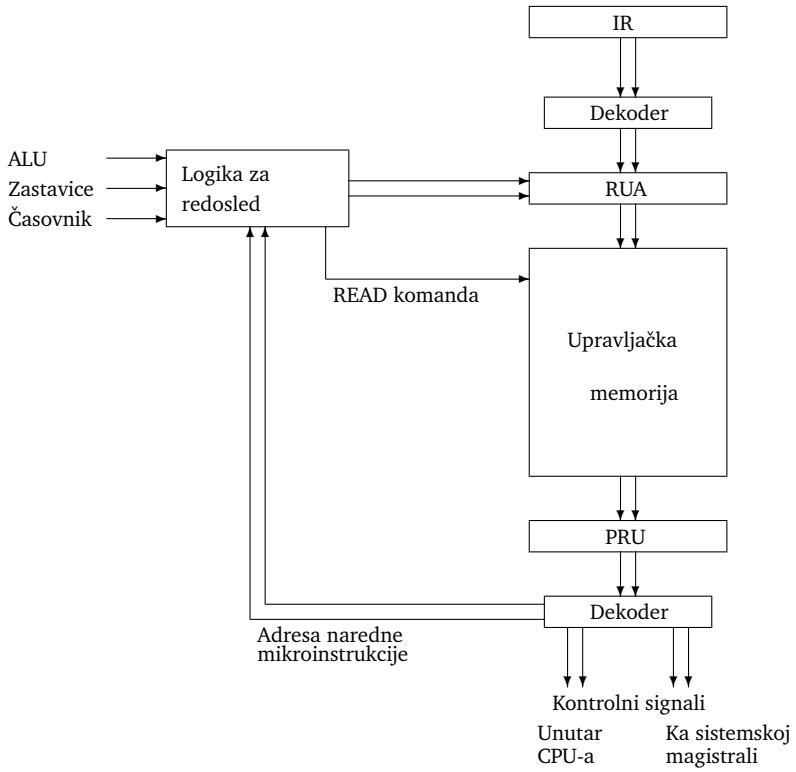


E– Kodovi mikrooperacije
F– Grananje (bezuslovno, nula, prekoračenje, indirekcija)
G– Adresa mikroinstrukcije

b) Vertikalna mikroinstrukcija

Slika 1: Tipovi formata mikroinstrukcija

Mikroprogramska upravljačka jedinica



Slika 2: Izvršavanje mikroinstrukcija u mikroprogramskoj upravljačkoj jedinici

Izvršavanje mikroinstrukcija u mikroprogramskoj upravljačkoj jedinici se odvija na sledeći način:

1. Logika zadužena za odredjivanje redosleda mikroinstrukcija izdaje READ komandu upravljačkoj memoriji.
2. Reč iz upravljačke memorije čija adresa se nalazi u RUA se prenosi u PRU.
3. Sadržaj PRU generiše kontrolne signale i informacije o narednoj instrukciji u nizu.
4. Jedinica zadužena za redosled puni adresu naredne instrukcije u RUA. Adresa se odredjuje na osnovu informacija dobijenih iz PRU o adresi naredne instrukcije i vrednosti zastavica koje su postavljene u ALU, na sledeći način:
   (a) ukoliko se ide na narednu instrukciju u nizu tada se sadržaj RUA poveća za 1;
   (b) ukoliko se vrši naredba grananja tada se u RUA puni sadržaj adresnog polja iz PRU;
   (c) ukoliko se grananje vrši na adresu rutine koja obradjuje narednu mašinsku instrukciju tada se adresa RUA odredjuje na osnovu operacionog koda sadržanog u IR.

Primene mikroprogramiranja

- *Implementacija upravljačke jedinice*
- *Realizacija računara emulacijom*
- *Podrška operativnom sistemu*
- *Podrška čvrsto vezanim multiprocesorskim sistemima*
- *Realizacija uredjaja specijalne namene*
- *Podrška višim programskim jezicima*
- *Mikrodijagnostika*
- *Pisanje korisničkih mikroprograma*