Elektrotehnički fakultet u Beogradu 06. maj 2008.
Katedra za računarsku tehniku i informatiku

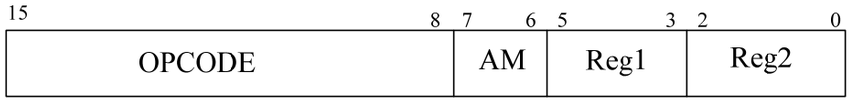
Kolokvijum iz Arhitekture i organizacije računara 2

Kolokvijum traje 120 minuta

Posmatra se dvoadresni processor koji ima 8 registra opšte namene, R0 do R7 i svi su 16-bitni. Postoji registar PSW sa uobičajenim značenjem. Memorijske adrese su širine 20 bita, širina magistrale podataka je 16 bita, a adresiranje je na nivou 16-bitnih reči. Procesor operiše samo sa 16-bitnim celobrojnim veličinama (u daljem tekstu *reč* označava 16-bitnu veličinu). Vreme odziva memorije je neodređeno, magistrala je asinhrona.

Pristup memorijskim lokacijama se vrši posredstvom specijalizovanih registara, tzv. s*egmentnih registara*. Postoje tri segmentna registra, CS (*Code Segment*), DS (*Data Segment*) i SS (*Stack Segment*). Adresa memorijske lokacije dobija se tako što se sadržaj odgovarajućeg segmentnog registra pomnoži sa 16 i na dobijenu vrednost doda pomeraj definisan načinom adresiranja. Prilikom pristupanja instrukcijama koristi se CS, prilikom pristupanja podacima DS, a prilikom pristupanja steku SS. Kao posledica ovakve organizacije, registar PC se sastoji od dva 16-bitna registra. Prvi registar je CS (*Code Segment*), a drugi IP (*Instruction Pointer*). Takođe, pokazivač na stek se sastoji iz registara SS (*Stack Segment*) i SP (*Stack Pointer*). Stek raste prema višim adresama, a par SS:SP pokazuje na prvu slobodnu lokaciju.

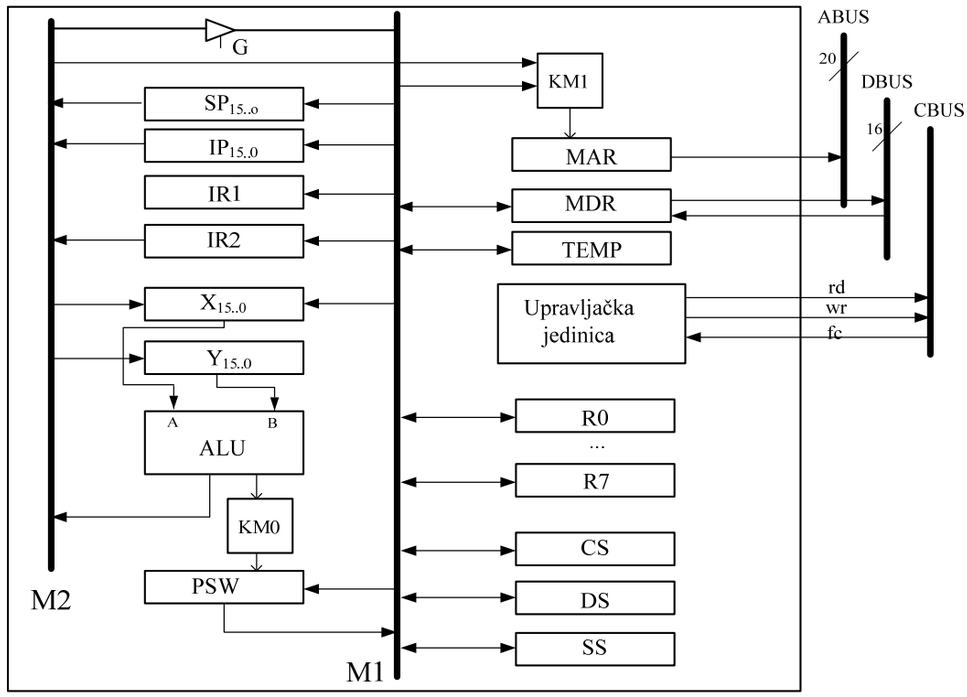
Instrukcije su dužine jedne ili dve memorijske reči. Format prve instrukcijske reči prikazan je na slici. Polje *Reg1* određuje registar koji se koristi za adresiranje prvog operanda u nekim od načina adresiranja (jedini operand kod jednoadresnih instrukcija, izuzev kod instrukcija skoka). Polje *Reg2* određuje drugi operand koji je uvek u jednom od registara opšte namene. Način adresiranja za prvi operand kodiran je u polju *AM* (00 – neposredno, 01 – registarsko direktno, 10 – memorijsko direktno, 11- registarsko indirektno sa pomerajem). Sve dvoadresne instrukcije kao odredište koriste registar naveden kao drugi operand. Instrukcija STORE je jedina koja vrši upis u memoriju, a njen efekat je da drugi operand (registar) stavi na adresu određenu prvim operandom.



Za instrukcije kontrole toka (skokovi, poziv i povratak iz potprograma) postoje dva različita tipa. Prva mogućnost je da je ciljna adresa unutar istog segmenta (*Code Segment* se ne menja), a druga mogućnost je inter-segmentni skok (ciljna adresa je određena novim segmentom i novim *Instruction Pointer*-om). Tako npr, postoje sledeće grupe instrukcija kontrole toka:

CALL	Poziv potrograma	Adresa skoka je u istom segmentu, IP je određen prvim operandom
FCALL	Intersegmentni poziv Potprograma (*Far Call*)	CS je određen registrom, koji je kodiran poljem *Reg2*, a IP je određen prvim operandom
RET	Povratak iz potrograma	Bezadresna instrukcija
FRET	Povratak iz udaljenog potrograma	Bezadresna instrukcija
FJMP	Bezuslovni intersegmentni skok	CS je određen poljem *Reg2*, a IP je određen prvim operandom
JMP	Bezuslovni unutarsegmentni skok	Adresa skoka je u istom segmentu, IP je određen prvim operandom
JN, JZ, JV, JC	Uslovni skokovi	Uvek unutarsegmentni skok, IP je određen prvim operandom

Principijelna šema organizacije procesora data je na sledećoj slici:



a) (4p) Nacrtati kompletnu strukturnu šemu mreže KM1 koja pravilno povezuje registar MAR u sistem.

b) (20p) Napisati mikroprogram za ovaj procesor sa fazom izvršavanja samo za sledeće instrukcije: CALL, RET, FCALL, FRET, JMP, FJMP i sve naredbe uslovnog skoka. Predvideti postojanje ostalih naredbi. **Čitanje svih instrukcijskih reči se izvršava pre faze izvršavanja bilo koje instrukcije**. Pretpostaviti postojanje kombinacione mreže koja će generisati signal *I* ukoliko postoji više od jedne instrukcijske reči. **Nije potrebno** pisati mikroprogram za obradu prekida, ali predvideti njegovo postojanje. Zanemariti postojanje internih procesorskih prekida. Ukoliko se koriste kombinacione mreže za generisanje signala uslova skoka, potrebno ih je nacrtati. Mikroprogram treba da bude prilagođen mikroprogramskoj upravljačkoj jedinici sa vertikalnim mikroprogramiranjem i spajanjem operacionih i upravljačkih koraka. Potrebno je da traženi deo mikroprograma bude što kraći.

c) (6p) Procesor ne poseduje mogućnost **uslovnog** skoka na lokacije van tekućeg segmenta. Kako se razrešava dati problem? Ilustrovati na sledećem primeru:

```
CMP  R0, #0
JZ   LabA
ADD  R1, R2
…
```

`LabA` se nalazi u drugom segmentu. Potrebno je napisati ispravni kod ekvivalentan datom, koji će omogućiti uslovni skok na udaljenu labelu `LabA`.