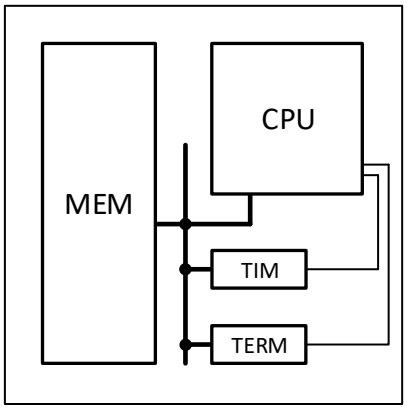
OPIS RAČUNARSKOG SISTEMA

Računarski sistem se sastoji od procesora, operativne memorije, tajmera i terminala. Sve komponente računarskog sistema su međusobno povezane preko sistemske magistrale. Tajmer i terminal, kao periferije, su povezani sa procesorom i preko linija za slanje zahteva za prekid. Slika 1. predstavlja uprošćen šematski prikaz posmatranog računarskog sistema.



Slika 1. Šematski prikaz računarskog sistema

Opis procesora

U nastavku je opisan deo 16-bitnog dvoadresnog procesora sa Von-Neuman arhitekturom. Adresibilna jedinica je jedan bajt, a raspored bajtova u reči je little-endian. Veličina memorijskog adresnog prostora je 2^{16}B. Počev od adrese `0xFF00` memorijskog adresnog prostora nalazi se prostor veličine 256 bajtova rezervisan za memorijski mapirane registre (registri kojima se pristupa instrukcijama za pristup memorijskom adresnom prostoru). Počev od adrese `0x0000` memorijskog adresnog prostora nalazi se IVT (interrupt vector table) sa osam ulaza. Svaki ulaz zauzima dva bajta i sadrži adresu odgovarajuće prekidne rutine. Ulazi u IVT odgovaraju sledećim prekidnim rutinama:

- ulaz 0 sadrži adresu prekidne rutine koja se izvršava prilikom pokretanja odnosno resetovanja čitavog procesora (ne izvodi se kompletna sekvenca obrade prekida već se samo vrši skok na adresu koja se nalazi u okviru datog ulaza),
- ulaz 1 sadrži adresu prekidne rutine koja se izvršava ukoliko se pokuša izvršavanje nekorektne instrukcije (nepostojeći operacioni kod, neispravan način adresiranja itd.),
- ulaz 2 sadrži adresu prekidne rutine koja se izvršava kada stigne zahtev za prekid od tajmera (opis principa rada tajmera i način njegove konfiguracije dat je u zasebnom poglavlju),
- ulaz 3 sadrži adresu prekidne rutine koja se izvršava kada stigne zahtev za prekid od terminala (opis principa rada terminala dat je u zasebnom poglavlju) i
- ostali ulazi su slobodni za korišćenje od strane programera.

Procesor poseduje osam opštenamenskih 16-bitnih registara označenih sa `r<num>` gde `<num>` može imati vrednosti od nula do sedam. Moguće je zasebno koristiti viših ili nižih osam bita svakog od opštenamenskih registara kao 8-bitni registar označen sa `r<num>h` ili `r<num>l`, respektivno. Registar `r7`

se koristi kao `pc` registar (pokazuje na instrukciju koja se u memoriji nalazi neposredno iza trenutno izvršavane instrukcije). Registar `r6` se koristi kao `sp` registar (pokazuje na zauzetu lokaciju na vrhu steka, a stek raste ka nižim adresama). Pored opštenamenskih registara postoji `psw` registar (statusna reč procesora).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I	Tl	Tr										N	C	O	Z

Značenje flegova u `psw` registru:

- `Z` (Zero) - rezultat prethodne operacije je nula,
- `O` (Overflow) – prekoračenje,
- `C` (Carry) - prenos,
- `N` (Negative) - rezultat je negativan,
- `Tr` (Timer) - maskiranje prekida od tajmera (0 - omogućen, 1 - maskiran),
- `Tl` (Terminal) - maskiranje prekida od terminala (0 - omogućen, 1 - maskiran) i
- `I` (Interrupt) - globalno maskiranje spoljašnjih prekida (0 - omogućeni, 1 - maskirani).

Instrukcije mogu biti veličine od jedan do sedam bajtova. Instrukcija u najopštijem slučaju ima sledeći format:

I	II	III	IV	V	VI	VII
InstrDescr	Op1Descr	Im/Di/Ad	Im/Di/Ad	Op2Descr	Im/Di/Ad	Im/Di/Ad

Prvi bajt instrukcije sadrži operacioni kod i dodatne informacije o instrukciji. Naredni bajtovi instrukcije koriste se za kodiranje operanada. Pojedinačni operand može zahtevati jedan, dva ili tri bajta za kodiranje u zavisnosti od načina adresiranja. Detaljan opis `InstrDescr` i `Op<num>Descr` bajtova instrukcije dat je u nastavku.

7	6	5	4	3	2	1	0
OC_4	OC_3	OC_2	OC_1	OC_0	S	Un	Un

Značenje bitova `InstrDescr` bajta instrukcije:

- $OC_4OC_3OC_2OC_1OC_0$ - operacioni kod instrukcije,
- `S` (Size) - veličina operanada instrukcije (0 - jedan bajt; 1 - dva bajta) i
- `Un` (Unused) - neiskorišćeni bitovi koji imaju fiksnu vrednost nula.

7	6	5	4	3	2	1	0
AM_2	AM_1	AM_0	R_3	R_2	R_1	R_0	L/H

Značenje bitova `Op<num>Descr` bajta instrukcije:

- $AM_2AM_1AM_0$ - kodiran način adresiranja pri čemu adresiranje može biti:
  - `0x0` - neposredno; vrednost operanda je kodirana u okviru instrukcije pomoću jednog ili dva `Im/Di/Ad` bajta u zavisnosti od veličine operanda; neposredno adresiranje nije validan način adresiranja za destinacioni operand,

- `0x1` - registarsko direktno; vrednost operanda nalazi se u registru čiji je broj kodiran u okviru instrukcije (nema `Im/Di/Ad` bajtova),
- `0x2` - registarsko indirektno bez pomeraja; vrednost operanda nalazi se u memoriji na adresi ukazanoj vrednošću registra čiji je broj kodiran u okviru instrukcije (nema `Im/Di/Ad` bajtova),
- `0x3` - registarsko indirektno sa 16-bitnim označenim pomerajem; vrednost operanda nalazi se u memoriji na adresi ukazanoj zbirom vrednosti registra, čiji je broj kodiran u okviru instrukcije, i vrednosti koja se nalazi u dva `Im/Di/Ad` bajta i
- `0x4` - memorijsko; vrednost operanda nalazi se u memoriji na adresi ukazanoj vrednošću koja se nalazi u dva `Im/Di/Ad` bajta,

- $R_3R_2R_1R_0$ - kodiran broj korišćenog registra (`psw` registar se kodira vrednošću `0xF`) i
- `L/H` (Low/High) - naznaka da li se koristi nižih ili viših osam bita registra (0 - nižih; 1 - viših) u slučaju registarskog direktnog adresiranja za operand veličine jednog bajta.

Mnemonik	OC	Efekat	Flegovi koji se menjaju
`halt`	0	Zaustavlja izvršavanje instrukcija	-
`iret`	1	`pop psw; pop pc;`	`psw`
`ret`	2	`pop pc;`	-
`int dst`	3	`push psw; pc<=mem16[(dst mod 8)*2];`	-
`call dst`	4	`push pc; pc<=dst;`	-
`jmp dst`	5	`pc<=dst;`	-
`jeq dst`	6	`if (equal) pc<=dst;`	-
`jne dst`	7	`if (not_equal) pc<=dst;`	-
`jgt dst`	8	`if (signed_greater) pc<=dst;`	-
`push src`	9	`sp<=sp-2; mem16[sp]<=src;`	-
`pop dst`	10	`dst<=mem16[sp]; sp<=sp+2;`	-
`xchg src, dst`	11	`temp<=dst; dst<=src; src<=temp;`	-
`mov src, dst`	12	`dst<=src;`	`Z N`
`add src, dst`	13	`dst<=dst+src;`	`Z O C N`
`sub src, dst`	14	`dst<=dst-src;`	`Z O C N`
`mul src, dst`	15	`dst<=dst*src;`	`Z N`
`div src, dst`	16	`dst<=dst/src;`	`Z N`
`cmp src, dst`	17	`temp<=dst-src;`	`Z O C N`
`not src, dst`	18	`dst<=~src;`	`Z N`
`and src, dst`	19	`dst<=dst&src;`	`Z N`
`or src, dst`	20	`dst<=dst|src;`	`Z N`

xor src, dst	21	dst<=dst^src;	Z N
test src, dst	22	temp<=dst&src;	Z N
shl src, dst	23	dst<=dst<<src;	Z C N
shr dst, src	24	dst<=dst>>src;	Z C N

Sintaksa operanada u okviru asemblerskih naredbi koje pristupaju podacima:

- `$<literal>` - neposredna vrednost `<literal>`
- `$<simbol>` - neposredna vrednost `<simbol>`
- `%r<num>` - vrednost iz registra `r<num>`
- `(%r<num>)` - vrednost iz memorije na adresi iz registra `%r<num>`
- `<literal>(%r<num>)` - vrednost iz memorije na adresi `<literal> + r<num>`
- `<simbol>(%r<num>)` - vrednost iz memorije na adresi `<simbol> + r<num>`
- `<simbol>(%pc/%r7)` - vrednost iz memorije na adresi `<simbol>` (PC relativno)
- `<literal>` - vrednost iz memorije na adresi `<literal>` (apsolutno)
- `<simbol>` - vrednost iz memorije na adresi `<simbol>` (apsolutno)

Sintaksa operanada u okviru asemblerskih naredbi predstavljaju skok:

- `<literal>` - skok na adresu `<literal>`
- `<simbol>` - skok na adresu `<simbol>`
- `*%r<num>` - skok na adresu iz registra `r<num>`
- `*(%r<num>)` - skok na adresu iz memorije na adresi iz registra `%r<num>`
- `*<literal>(%r<num>)` - skok na adresu iz memorije na adresi `<literal> + r<num>`
- `*<simbol>(%r<num>)` - skok na adresu iz memorije na adresi `<simbol> + r<num>`
- `*<simbol>(%pc/%r7)` - skok na adresu iz memorije na adresi `<simbol>` (PC relativno)
- `*<literal>` - skok na adresu iz memorije na adresi `<literal>` (apsolutno)
- `*<simbol>` - skok na adresu iz memorije na adresi `<simbol>` (apsolutno)

Dodatne napomene:

- sve aritmetičke operacije se izvode tako da odgovaraju označenim celim brojevima,
- iza mnemonika asemblerske naredbe, bez belih znakova, može se navesti sufiks `b` ili `w` kako bi se eksplicitno naznačila veličina operanada date instrukcije,
- instrukcije `cmp` i `test` nigde ne čuvaju direktni rezultat odgovarajuće operacije, već samo u skladu sa rezultatom postavljaju nove vrednosti flegova u `psw` registru i
- kombinacije instrukcija i operanada, za koje ne postoji razumno tumačenje, proglasiti greškom.

Opis terminala

Terminal predstavlja ulazno/izlaznu periferiju koja se sastoji od displeja i tastature. Terminal poseduje dva memorijski mapirana registra. Na adresi `0xFF00` memorijskog adresnog prostora nalazi se `data_out` registar izlaznih podataka. Upisom vrednosti u `data_out` registar na tekućoj poziciji displeja ispisuje se znak koji prema ASCII tabeli odgovara upisanoj vrednosti. Na adresi `0xFF02` memorijskog adresnog prostora nalazi se `data_in` registar ulaznih podataka. Kada se pritisne neki taster (1) upisuje se ASCII kod pritisnutog tastera u `data_in` registar i (2) terminal, kao periferija posmatranog procesora, generiše zahtev za prekid (u okviru prekidne rutine, koja obrađuje ovaj zahtev za prekid, čitanjem vrednosti `data_in` registra može se saznati koji taster je pritisnut).

Opis tajmera

Tajmer kao periferija periodično generiše zahtev za prekid. Perioda generisanja zahteva za prekid definisana je sadržajem `timer_cfg` konfiguracionog registra tajmera. Registar `timer_cfg` je memorijski mapiran registar i nalazi se na adresi `0xFF10` memorijskog adresnog prostora. Njegova inicijalna vrednost nakon pokretanja odnosno resetovanja računarskog sistema jeste `0x0000`.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													T_2	T_1	T_0

Perioda generisanja zahteva za prekid u zavisnosti od $T_2T_1T_0$ vrednosti je sledeća: `0x0` -> 500ms, `0x1` -> 1000ms, `0x2` -> 1500ms, `0x3` -> 2000ms, `0x4` -> 5000ms, `0x5` -> 10s, `0x6` -> 30s i `0x7` -> 60s.