**EMULATOR RAČUNARSKOG SISTEMA ZA IZVRSAVANJE PREDMETNOG KODA IZLAZA DVO PROLAZNOG ASEMBLERA**

Autor: Strahinja Janjic

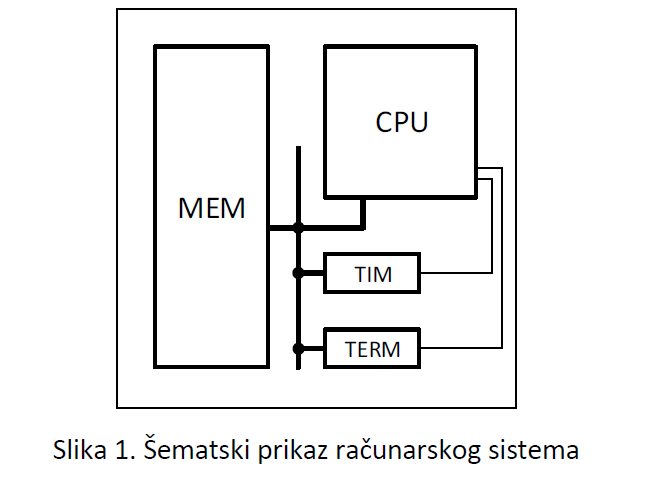
Index: 0411/2015

**UVOD**

U prilogu je dat opis jednog jednostavnog računarskog sistema koji sluzi za učitavanje linkovanje i izvrsavanje predmetnog programa koji je izlaz dvoprolaznog asemblera. U daljem tekstu bice dat opis samog računarskog sistema i njegovih periferija kao i prikaz sekcija koji su učitane i proces izvršavanja.

**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.

****

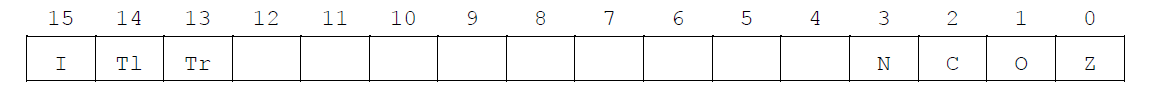
**OPIS PROCESORA**

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 216B. 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.

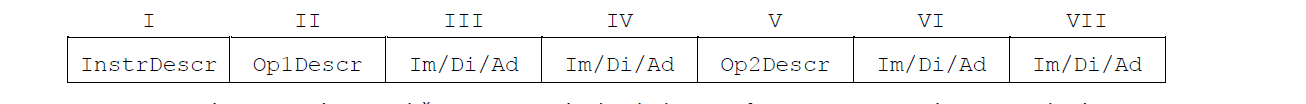
Procesor poseduje osam opštenamenskih 16-bitnih registara označenih sa *r<num>* gde *<num>* može imati vrednosti od nula do sedam. Moguće je zasebno koristiti viših ili nižih osam bita svakog od opštenamenskih registara kao 8-bitni registar označen sa *r<num>h* ili *r<num>l*, respektivno. Registar r7

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).

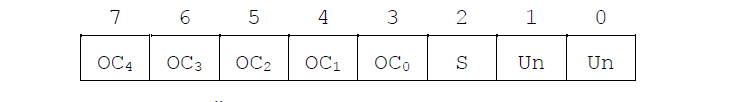


Značenje flegova u psw registru:

* 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).

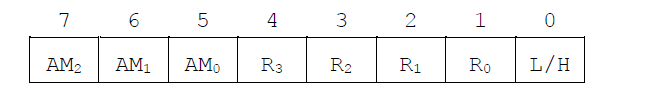
Instrukcije mogu biti veličine od jedan do sedam bajtova. Instrukcija u najopštijem slučaju ima sledeći format:

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.



Značenje bitova InstrDescr bajta instrukcije:

* OC4OC3OC2OC1OC0 - operacioni kod instrukcije,
* S (Size) - veličina operanada instrukcije (0 - jedan bajt; 1 - dva bajta) i
* Un (Unused) - neiskorišćeni bitovi koji imaju fiksnu vrednost nula.



Značenje bitova Op<num>Descr bajta instrukcije:

* AM2AM1AM0 - 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 8-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 jednom Im/Di/Ad bajtu,
* 0x4 - 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
* 0x5 - memorijsko; vrednost operanda nalazi se u memoriji na adresi ukazanoj vrednošću koja se nalazi u dva Im/Di/Ad bajta,
* R2R1R0 - 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.

**OPIS TERMINALA**

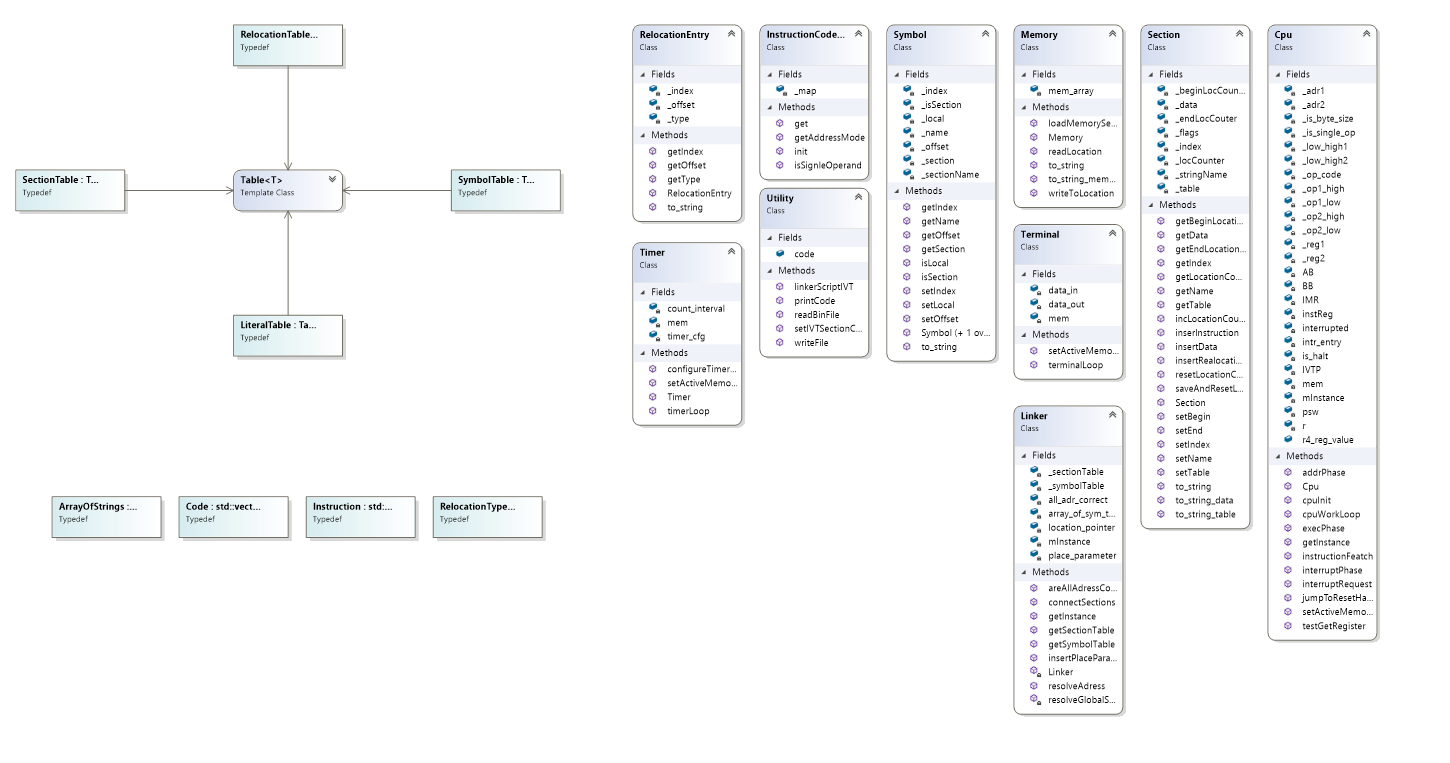
Terminal predstavlja ulazno/izlaznu periferiju koja se sastoji od displeja i tastature. Terminal poseduje dva memorijski mapirana registra. Na adresi 0xFF00 memorijskog adresnog prostora nalazi se data\_out registar izlaznih podataka. Upisom vrednosti u data\_out registar na tekućoj poziciji displeja 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 TIMERA**

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.

Perioda generisanja zahteva za prekid u zavisnosti od T2T1T0 vrednosti je sledeća: 0x0 -> 500ms, 0x1 -> 1000ms, 0x2 -> 1500ms, 0x3 -> 2000ms, 0x4 -> 5000ms, 0x5 -> 10s, 0x6 -> 30s i 0x7 -> 60s.

**DIZAJN EMULATORA**

****

Sam emulator sadrži dosta sličnih struktura podataka kao i dvorolazni asembler što je i logično jer rade nad istim skupom podataka.

Pored već opisanih klasa za Tabele, Sekcije, Simbole I Utility klasa postoje jos dosta klasa specifičnih za samu realizaciju emulatora.

Klasa Utility služi za čitanje ulaznih binarnih fajlova kao, ali I služi I kao loader u kome su definisane specijalne “linkerske skripte” za ucitavanje init objektnog fajla u kojem je u asebleru definisana tabela prekida kao i prekidne rutine.

Klasa Linker sluzi za povezivanje sekcija iz više fajlova kao I razrešavanje realokacija svih simbola nakon učitavanja sekcije na datu lokaciju.

Klasa Memory sadžri jednostavnu implementaciju memorije sa operacijama ucitačitavanje segmenta, ucitavanje reci i čitanje reči. Takodje ovde su implementirane metode za čitljiv ispis memorije

Klasa Cpu ima definisan definisan workLoop koji se sastoji iz 4 faze, faza čitanja instrukcije, faza adresiranja, faza izršavanja, faza obrade prekida. Procesor se vrti u ovom loop-u sve dok ne naiđe na instrukciju halt.

Klasa Timer emulira timer koji na zadato vreme javlja prekid procesoru da je prošao dati interval vremena.

Klasa Terminal emulira jednostavan terminal koji upisuje karaktere u memoriju i javlja prekid procesoru da je došlo do upisa karaktera.

Klase Memory, Timer, Cpu i Terminal imaju dodeljene posebne niti za njihove workLoop metode.

**INSTRUKCIJE**

halt

xchg dst, src

int dst

mov dst, src

add dst, src

sub dst, src

mul dst, src

div dst, src

cmp dst, src

not dst

and dst, src

or dst, src

xor dst, src

test dst, src

shl dst, src

shr dst, src

push src

pop dst

jmp dst

jeq dst

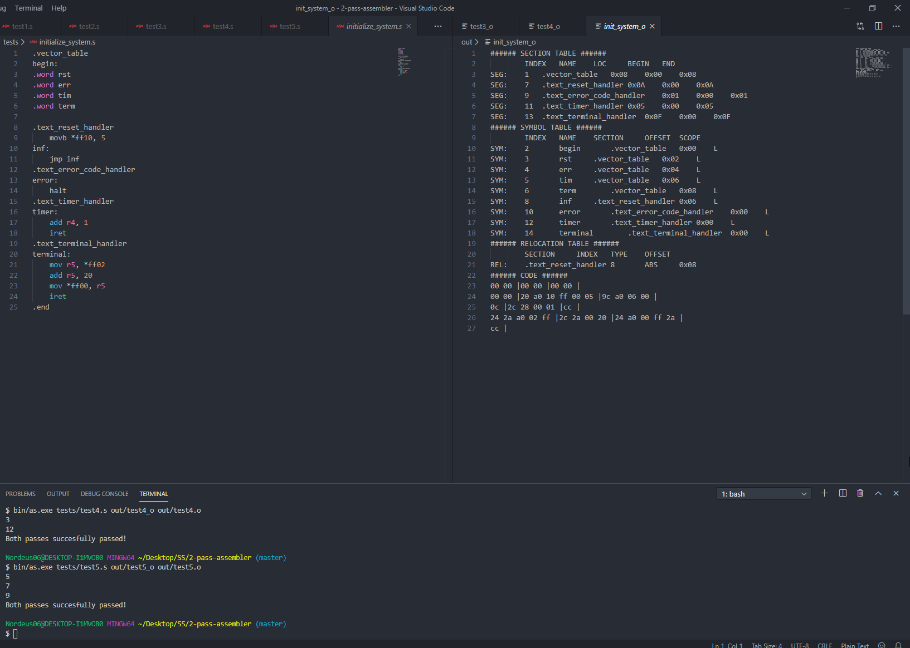
jne dst

jgt dst

call dst

ret

iret

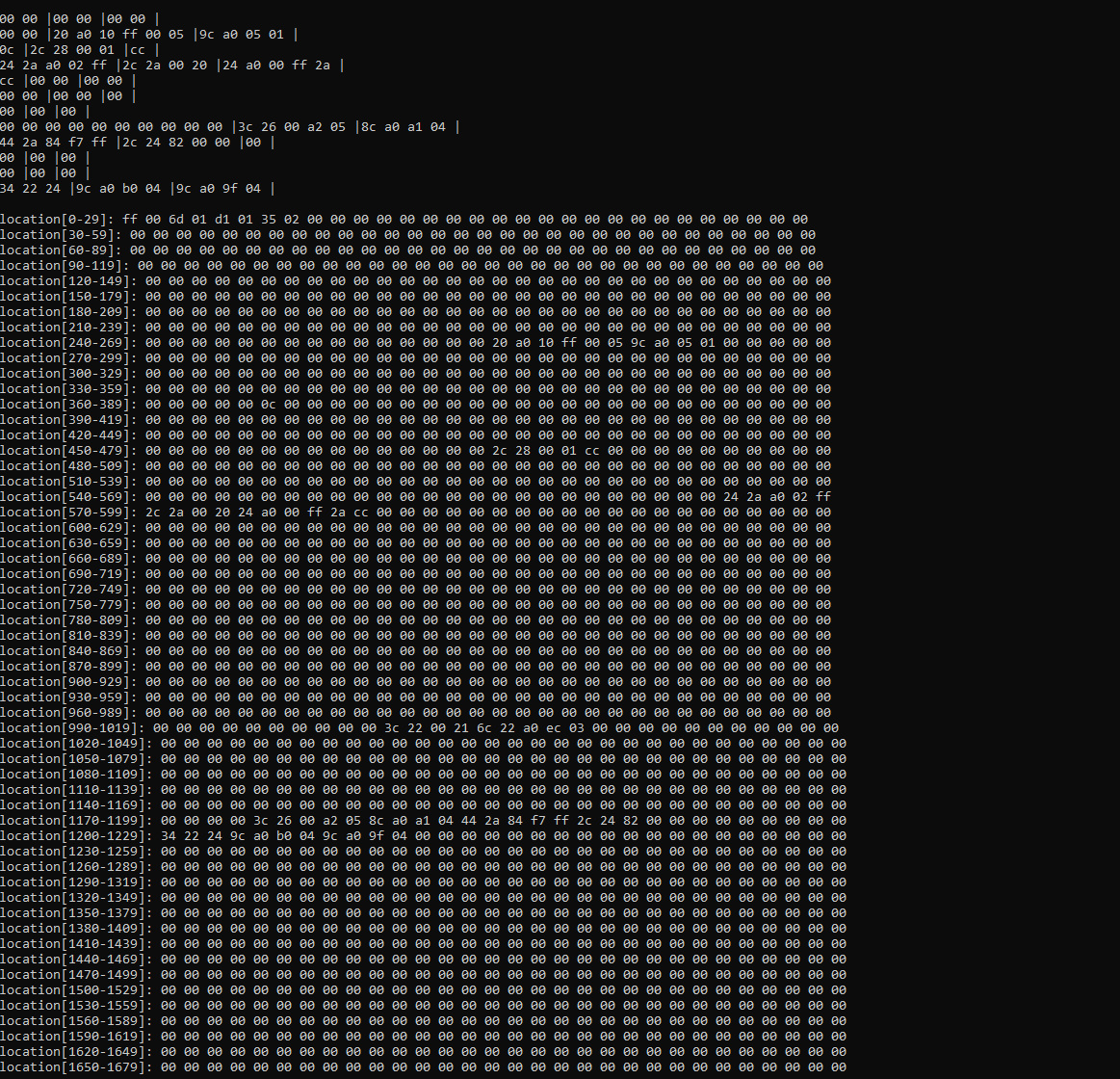
**SYSTEM INITIALIZATION CODE**

U ovom kodu su definisane sekcije vektor tabele i sekcije prekidnih rutina. Uradjeno je tako da reset handler inicijalizuje memorijsku lokaciju koju tajmer čita da bi saznao period brojanja, i nakon toga ulazi u beskonačnu petlju.

Error code handler daje halt instrukciju procesoru, on će se javiti svaki put kada dodje do greske u dekodovanju instrukcije ili u adresiranju.

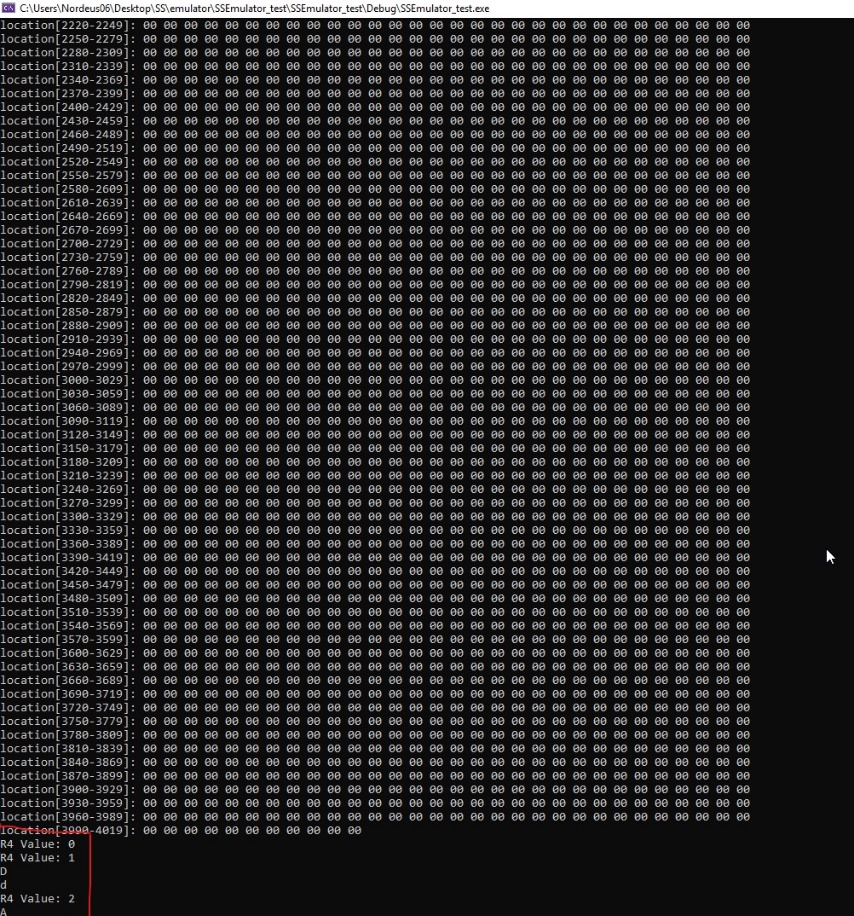
Timer reset handler uvecava vrednost registra r4 i vraca kontrolu rada glavnom procesu.

Terminal handler cita iz memorijske lokacije upisan karakter uvecava ga za 32(20h) što će u ASCII smislu veliko slovo pretvoriti u malo, i ispisuje to u memorijsku lokaciju iz koje cita terminal. Nakod tog upisa na terminalu ce se pojaviti character.

**IZVRŠAVANJE**

Program se izvršava sa ucitavanjem test4.o sa argumentima [-place=.text\_main@0x480](mailto:-place=.text_main@0x480) gde se automatski učitava init skripta u odvojenom procesu loadovanja od ostalih fajlova i sekcije se smeštaju na predefinisane lokacije. Za navedene sekcije može se videti da su smeštene na korektne adrese. Sekcija .text\_main je smestena na lokaciju 0x480(1152) dok je .text\_zma smestena počev od prodrazumevane adrese 1000.

Program se izvršava baš onako kako je zamišljen, naime procesor se vrti u beskonačnoj petlji i čeka na prekid a timer i terminal asinhrono prekidaju procesor i bacaju ga u stanje izvrsavanje prekidnih rutina što se vidi i na narednoj slici izvršavanja.

****