Miroslav POPOVIĆ
Sonja ZOVKO
Fakultet tehničkih nauka
Institut za računarstvo,
automatiku i merenje
21000 Novi Sad, V.Vlahovića 3.

JEDNO REŠENJE SIMULATORA RAČUNARA EiH6/DPS6/DPS6000

ONE SOLUTION OF THE EiH6/DPS6/DPS6000 COMPUTER SIMULATOR

SADRŽAJ - U radu je opisano jedno rešenje simulatora računara EiH6/DPS6/DPS6000 (obuhvata centralni procesor (CPU), jedinicu za rukovanje memorijom (MMU), memoriju, panel i kontroler diska), koji je realizovan na računaru IBM PC i napisan u programskom jeziku C.

ABSTRACT - The paper describes one solution of the EiH6/DPS6/DPS6000 computer simulator (involves central processor unit (CPU), memory management unit (MMU), main memory, panel, and disk controller), implemented on the IBM PC computer using the C programing language.

1. UVOD

Emulator nekog računarskog sistema se sastoji od odgovarajućih dodatnih fizičkih komponenti, mikroprograma i programa na objektnom računarskom sistemu, a omogućava izvršenje programa sa emuliranog sistema [1]. Izostavljanjem dodatnih fizičkih komponenti i odgovarajućih mikroprograma emulator degradira u simulator [5], [6].

Problem testiranja je jedna od glavnih prepreka u razvoju velikih programskih paketa, uključujući i sisteme sa podelom vremena (time-sharing). Testiranje ovih sistema zahteva dosta vremena i napora za rešavanje sledećih problema:

a) vraćanje ili postavljanje sistema u početno ili zadato stanje,

b) asinhrono generisanje zahteva od strane perifernih uredjaja u realnom vremenu,

c) očuvanje stanja sistema prilikom prekida i slično.

U-I uredjaji i CPU računarskog sistema rade istovremeno. Moguća je njihova simulacija korišćenjem tehnike podele vremena sa odgovarajućim korutinama [3],[4]. U-I uredjaji se dele u dve klase. U prvu spadaju uredjaji kao što su diskovi, magnetne trake i štampači i za ovu klasu može se izračunati vreme završetka U-I radnje. Drugu klasu uredjaja čine terminali, kod kojih je nemoguće izračunati vreme završetka U-I radnje [2].

Simulacija uredjaja iz prve klase je jednostavna. Simulaciona rutina za početak U-I radnje poziva korutinu koja simulira uredjaj. Ova korutina bezuslovno obavlja zahtevanu U-I operaciju i završava se. Za drugu klasu uredjaja ne može se uvek unapred predvideti vreme završetka U-I radnje, zbog mogućeg prozivanja više uredjaja.

Cilj ovog rada je da se obezbedi potpuni radni ambijent (prisustvo operativnog sistema, komandi i aplikacija), jednog miniračunara kao što je EiH6 na mikroračunaru IBM PC. Neki od motiva za realizaciju ovakvog simulatora su:

- razvoj programa na IBM PC računaru radi rasterećenja osnovnog računarskog sistema;
- mogućnost korišćenja programske podrške osnovnog sistema na IBM PC mikroračunaru;
- mogućnost generalizacije rešenja za širu klasu mini i srednjih računarskih sistema (VAX, IBM OS/370 i sl.);
- preuzimanje obrade u slučaju kvara osnovnog sistema.

Simulator se sastoji iz četiri dela:

a) *Simulator panela* - omogućava kontrolu toka izvršenja programa uz mogućnost prikaza sadržaja registara i željene memorijske lokacije kao i njihovu promenu. Panel vrši inicijalizaciju simulatora i njegovo pokretanje.

b) *Simulator centralnog procesora* - koji u sebi sadrži tri faze:

- fazu zahvatanja koda instrukcije iz operativne memorije i odredjivanje efektivne adrese operanda;

- fazu pripreme za izvršenje u kojoj se na osnovu koda instrukcije odredjuje da li data instrukcija zahteva pristup memoriji (očitavanje i/ili upis) i
- fazu izvršenja instrukcije.

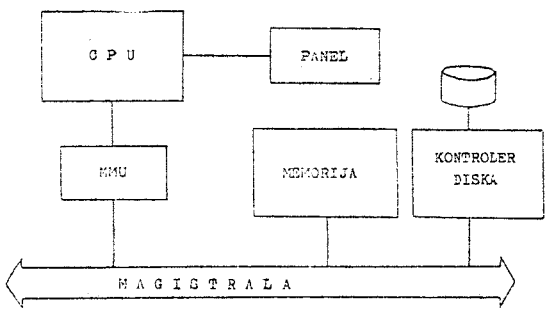
U okviru ovog dela realizovana je i simulacija sistema prekida i sistema zamki.

c) *Simulator jedinice za rukovanje memorijom* - u sebi sadrži:

- deo za ispitivanje prava izvršenja instrukcije, i
- deo za ispitivanje prava pristupa memoriji (očitavanje i/ili upis).

d) *Simulator kontrolera diska* - koji obavlja samo neke od njegovih stvarnih operacija (očitavanje sa diska, upis na disk).

Fizička arhitektura simuliranog sistema je prikazana na slici 1.



Slika 1: Fizička arhitektura simuliranog sistema

2. SIMULATOR CENTRALNOG PROCESORA

Za svaku instrukciju nekog programa prolazi se kroz najviše četiri stanja: stanje zahvatanja instrukcije, stanje dekodiranja, stanje zahvatanja operanda i stanje izvršenja instrukcije.

Nakon inicijalizacije sistema u programski brojač smešta se adresa prve instrukcije koja treba da se izvrši. U fazi zahvatanja instrukcije, očitava se sadržaj memorijske lokacije na koju ukazuje programski brojač i smešta u registar instrukcije (rin). U fazi dekodiranja se, na osnovu zadnjih sedam bita registra instrukcije, odredjuje kojom tehnikom adresiranja je adresiran operand i sračunava se efektivna adresa operanda (ukoliko postoji). Preostali biti rin koriste se za odredjivanje same instrukcije.

Nakon odredjivanja efektivne adrese (ako postoji) preuzima se operand i prelazi u fazu izvršenja. U fazi izvršenja se na osnovu koda operacije utvrdjuje o kojoj instrukciji je reč i ona se izvršava.

3. SISTEM PREKIDA

Simulator podržava programski izazvane prekide (LEV instrukcija) i spoljne prekide (signal prekida generiše neki spoljni uredjaj - npr. kontroler diska).

Simulator podržava 64 nivoa prioriteta prekida (od Ø do 63). Najviši prioritet ima nivo Ø, a najniži prioritet ima nivo 63. Za svaki nivo prioriteta u nultoj memorijskoj stanici (hardverska baza) se rezerviše po jedan bit aktivnosti (IAF), koji označava da li je proces na tom nivou aktivan ili ne.

Osim toga, za svaki nivo prioriteta u Hardverskoj bazi, rezervisane su memorijske lokacije u koje se smešta adresa kontrolnog bloka zadatka (TCB). U okviru TCB smeštaju se razne informacije o zadatku, a postoji i prostor za čuvanje stanja programa (ISA) u slučaju da bude prekinut. Svaki zadatak u sistemu ima svoj TCB i pridružuje mu se jedan od nivoa prioriteta. Nivo prioriteta tekućeg izvršivog programa smešten je u registru stanja (S).

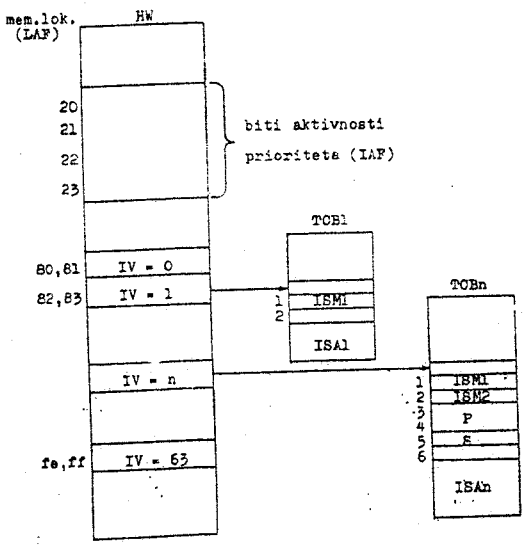
Tekući obradjivani proces može biti prekinut nailaskom na odredjenu LEV instrukciju, pri čemu se vrši očuvanje stanja programa koji se prekida u njemu odgovarajuću ISA i vraćanje stanja programa koji

će početi da se izvršava iz njegove odgovarajuće ISA. Ukoliko prekid izaziva neki uredjaj, bez obzira u kom trenutku se pojavio signal prekida, tekući obradjivani proces može da se prekine tek nakon faze izvršenja tekuće instrukcije. Zbog toga, pre zahvatanja koda instrukcije vrši se ispitivanje da li se u medjuvremenu javio signal prekida ili ne. Ako nije nastavlja se izvršenje tekućeg obradjivanog procesa.

U slučaju pojave prekida, postavlja se njemu odgovarajući bit aktivnosti (IAF) i nakon toga, uporedjuju se nivoi prioriteta tekućeg obradjivanog procesa (TOP) i uredjaja koji je poslao signal prekida.

Ako je nivo prioriteta TOP manji (brojčano) od prioriteta uredjaja, sledi da je TOP većeg prioriteta i on nastavlja da se izvršava. U protivnom sledi, da je uredjaj prioritetniji, te se nakon očuvanja stanja TOP i vraćanja stanja rukovaoca uredjaja, CPU prepušta rukovaocu uredjaja na korišćenje.

Izgled strukture podataka korišćene u sistemu prekida, dat je na slici 2.



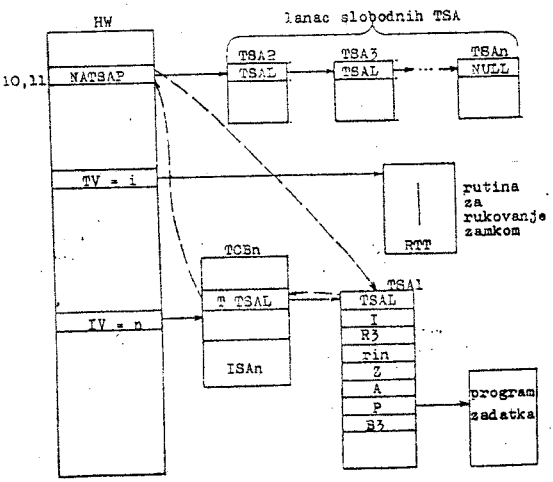
Slika 2. Struktura podataka sistema prekida

4. SISTEM ZAMKI

Simulator podržava zamke, tj. u slučajevima koji dovode do prekida TOP (zbog neispunjenih unapred zadatih uslova ili nailaska na instrukciju koja izaziva ulazak u zamku) obezbedjuje prelazak na rutinu za rukovanje zamkom.

Povratak iz rutine za rukovanje zamkom vrši se instrukcijom RTT (Return From Trap). U simulatoru, postoji 46 mogućih zamki za koje su vezani odredjeni problemi. Za svaku zamku, u Hardverskoj bazi se rezervišu memorijske reči koje sadrže adresu rutine za rukovanje tom zamkom. Prilikom pojave zamke izdvaja se blok TSA za čuvanje stanja prekinutog programa iz lanca slobodnih TSA i adresa tog bloka se smešta u odredjene lokacije odgovarajućeg TCB. Nakon toga, prelazi se na rutinu koja rukuje zamkom.

Vraćanje u prekinuti program se vrši instrukcijom RTT. Ova instrukcija izaziva vraćanje stanja prekinutog programa iz izdvojenog TSA, a nakon toga vraćanje izdvojenog TSA u listu slobodnih TSA, što je prikazano slikom 3.



Slika 3: Struktura podataka sistema zamki

5. SIMULATOR KONTROLERA DISKA

Omogućava izvršenje U-I instrukcija. Kontroler diska u sebi sadrži malu memoriju - SPM (Scratch Pad Memory) u kojoj se čuvaju informacije potrebne za rad sa diskom. Kao disk koristi se relativna datoteka DISK.

U-I instrukcije kodiraju se sa dva ili tri izraza sa adresama (tri - u slučaju IOLD instrukcije). Reč adresirana drugim izrazom sadrži broj kanala adresiranog kontrolera i funkcijski kod zahtevane radnje (FC - nižih 6 bita).

Izlazne komande sadržaj koji odredjuje prvi izraz sa adresom U-I instrukcije smeštaju u odgovarajuću lokaciju SPM. Ulazne komande odgovarajući sadržaj SPM prenose u odgovarajuću lokaciju memorije (adresa je odredjena prvim izrazom sa adresom U-I instrukcije).

Izlazna komanda zadavanja zadatka (FC=Ø7) upisuje odgovarajući sadržaj u SPM. Na osnovu tog sadržaja definisane su sledeće radnje:

a) Ø - RECALIBRATE - postavljanje početnog stanja
b) 1 - seek - pozicioniranje na cilindar zadat u konfiguracionoj reči A
c) 1ØAAAØØ1 - READ/WRITE - čitanje/upis sa/na disk.

Da li je reč o čitanju ili pisanju, zavisi od bita najmanje težine CH (=1 čitanje sa diska; =Ø - upis na disk).

Na osnovu vrednosti konfiguracionih reči A i B vrši se pozicioniranje na željeni sektor. Vrednost odstojanja predstavlja relativno odstojanje željenog bajta od početka željenog sektora. Opseg predstavlja broj znakova koji se prenosi. Memorijska adresa sa koje se očitava sadržaj ili na koju se upisuje sadržaj, data je memorijskom adresom znaka (nižih 16 bita) i memorijskom adresom modula (viša 4 bita).

6. SIMULATOR JEDINICE ZA RUKOVANJE MEMORIJOM

Osnovni zadatak mu je da virtualne adrese programa pretvara u stvarne adrese, koje ukazuje na željene lokacije u memoriji. MMU deli ukupn emoriju sistema na 31 segment. Od toga ima 16 malih segmenata i 15 velikih segmenata. Svakom segmentu dodeljuje se segment deskriptor.

Segment deskriptor sadrži sledeće informacije o segmentu: da li segment može da se koristi ili ne, početnu adresu segmenta, da li je dozvoljeno čitanje iz segmenta, upis u segment i/ili izvršenje in-

strukcije u tom segmentu i veličinu segmenta.

Simulator MMU obavlja dve funkcije:

a) proveru prava izvršenja instrukcije i

b) proveru prava pristupa memoriji (čitanje i pisanje).

7. ZAKLJUČAK

Rad predstavlja jedno rešenje simulatora računara EiH6/DPS6/ /DPS6000. Kontrola izvršenja zadatka vrši se korišćenjem mogućnosti panela CPU. Punjenje objektnog programa u korisnički deo memorije obavlja funkcija LOAD, realizovana u panelu.

Konačni cilj ovog rada je formiranje potpunog radnog ambijenta koji pruža računar EiH6/DPS6/DPS6000, a što znači da potpun simulator mora da prihvati OS GCOS6 MOD 400, sve sistemske komande (SYSLIB2 i SYSLIB1) kao i korisničke aplikacije računara EiH6/DPS6/DPS6000.

Na simulatoru je oživljen i napunjen OS GCOS MOD 400 rel 3.1.

Simulator je pisan u TURBO C (BORLAND) i imajući u vidu resurse mikroračunara IBM PC može se zaključiti da će performansa simulatora zadovoljiti postavljene ciljeve u razvoju.

Imajući u vidu svojstva C-jezika, put ka generalizaciji simulatora na druge računarske sisteme je očigledan i opravdan.

Iz dosadašnjeg rada uočene su prednosti u pogledu mogućnosti:

- razvoja programa na IBM PC računaru radi rasterećenja osnovnog računarskog sistema,
- mogućnost korišćenja programske podrške osnovnog sistema na IBM PC mikroračunaru,
- mogućnost generalizacije rešenja za širu klasu mini i srednjih računarskoh sistema (VAX, IBM 370 i sl.) i
- mogućnost preuzimanja obrade u slučaju kvara osnovnog sistema.

8. LITERATURA

[1] E.G.Mallach, EMULATION A SURVEY, Honeywell Computer Journal, vol. 6, No.4, (1972).

[2] K.Fuchi, H.Tanaka, Y.Namago and T.Yuba, A PROGRAM SIMULATION BY PARTIAL INTERPRETATION, 2nd Symposium on Operating Systems Principles, Princenton (1969).

[3] H.Markowitz, B.Hausner and H.Karr, SIMSCRIPT: A SIMULATION PROGRAMMING LANGUAGE, Prentice-Hall, Inc. (1963).

[4] D.E.Knuth and J.L.Mcneley, SOL-A SIMBOLIC LANGUAGE FOR GENERAL-PURPOSE SYSTEMS SIMULATION, IEEE Transactions on Electric Computers, Avg.(1964).

[5] E.G.Mallach, SIMULATING THE VIATRON 2140/2150 ON THE IBM 360, Simulation 16, No. 3, (1971).

[6] G.R.Trimble, USING A COMPUTER TO SIMULATE A COMPUTER, Data Processing 7, No. 10, (1965).