

Korisnička dokumentacija SSPARCSS simulatora

v.1.0

Danko Basch

Sadržaj

1	Uvod.....	1
2	SSPARCSS simulator	2
2.1	Osnovno korištenje simulatora	2
2.2	Pokretanje simulatora	2
2.2.1	Pokretanje simulatora i obnavljanje postavki	2
3	Pregled sučelja SSPARCSS simulatora	3
4	Osnovni tijek rada u simulatoru	6
5	Dijelovi glavnog prozora	7
5.1	Izbornici	7
5.1.1	Izbornik File	7
5.1.2	Izbornik View	7
5.1.3	Izbornik Options	8
5.2	Statusna traka	8
6	Najvažnija polja u sučelju	9
6.1	Polje Preview	9
6.1.1	Polje Preview prilikom prikaza modela	10
6.1.2	Kontekstni izbornici polja Preview	10
6.1.3	Gumbi u polju Preview	12
6.1.4	Polje Preview prilikom prikaza komponente	13
6.2	Polje Debug	14
6.2.1	Stupac s prekidnim točkama	14
6.2.2	Adresni stupac	16
6.2.3	Stupac s programom	17
6.2.4	Alatna traka	18
6.3	Polje MemView	20
6.3.1	Izgled polja MemView	21
6.3.2	Akcije u polju MemView	22
6.4	Polje StackView	23
7	Ostala polja u sučelju	25
7.1	Polje Console	25
7.1.1	Poruke u polju Console	25
7.1.2	Praćenje stanja simulacije u polju Console	25
7.1.3	Akcije u polju Console	26
7.2	Polje Navigator	26
7.2.1	Komponente u polju Navigator	27
7.2.2	Lokacije u polju Navigator	27
7.3	Polje Display	29
7.4	Polje Dump	30
8	Opcije	31
8.1	Raspoložive opcije	31
8.2	Spremanje i obnavljanje opcija i postavki	32

Predgovor

Ovaj dokument opisuje simulatorsku aplikaciju programskog sustava SSPARCSS (engl. *Software Suite for Processor ARChitectue Simulation and Study*). SSPARCSS assembler zajedno sa simulatorom čini programski paket SSPARCSS čija glavna namjena je simuliranje rada procesora i jednostavnih računalnih sustava u edukacijske svrhe. SSPARCSS je nasljednik programskog paketa ATLAS koji se u istu svrhu na FER-u koristio od 1991. godine.

Dokument je namijenjen krajnjim korisnicima u slučaju da im sučelje nije dovoljno intuitivno i da im je potreban detaljniji opis svih funkcionalnosti simulatora.

U dokumentu se koristi svega nekoliko konvencija:

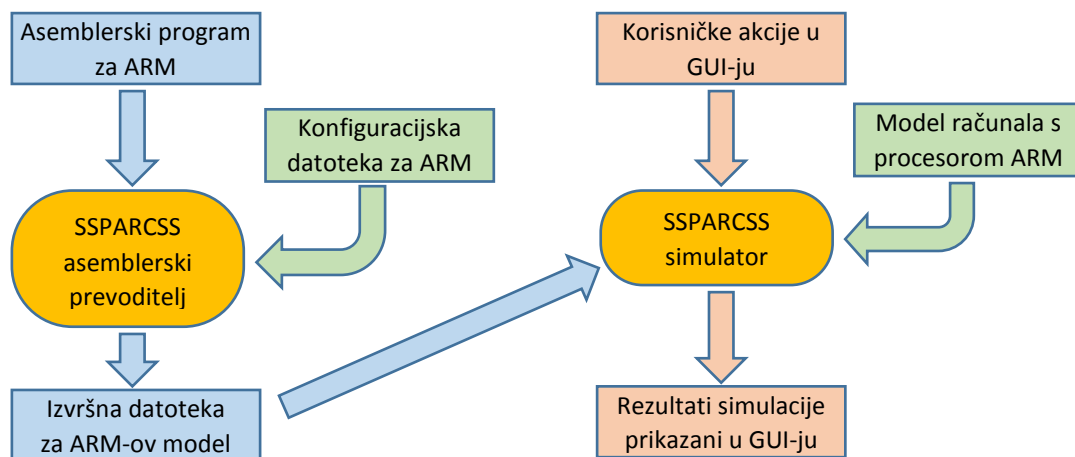
- **podebljana slova** služe za isticanje važnijih dijelova teksta
- *kosa slova* označavaju engleske riječi za koje nismo pronašli prikladan hrvatski prijevod (*debugger*, *default*, *radio-button*, *drag-and-drop*, *breadcrumbs*, *zoom* itd.)
- *plava kosa slova* označavaju engleske riječi koje nema smisla prevoditi jer su to originalna imena dijelova simulatora ili doslovni natpisi vidljivi u samom simulatoru pa tako i na slikama u tekstu

1 Uvod

Općenito, **simulatori** su programi kojima možemo oponašati rad nekog **sustava** bez potrebe da zaista imamo na raspolaganju taj sustav. To se postiže tako da se prvo **sustav opiše modelom**, odnosno da se **sustav modelira**. Zatim simulator na temelju **modela** oponaša rad sustava.

Postoje najrazličitije vrste modela i simulatora s najrazličitijim namjenama - od bioloških, medicinskih, kemijskih, meteoroloških, tehničkih, mehaničkih, prometnih, električkih, pa do računalnih. Poznati primjer jezika za modeliranje digitalnih sustava je VHDL. Ovaj dokument opisuje **simulator programskog paketa SSPARCSS** koji omogućuje simulaciju rada **procesora**. Ovaj dokument ne opisuje jezik za modeliranje (što možemo smatrati naprednim korištenjem), već samo uporabu simulatora sa stajališta krajnjeg korisnika koji ima na raspolaganju gotove modele. Kao što prikazuje slika, korisnik će:

1. pisati svoje programe u asemblerskom jeziku
2. asemblirati ih SSPARCSS-ovim asemblerom
3. učitavati izvršne datoteke u memoriju modela
4. simulirati njihovo izvođenje na modelu procesora.



Na slici su zelenom bojom prikazane različite konfiguracijske datoteke koji već postoje i za korisnika su uglavnom nevidljive. Plavom bojom prikazane su datoteke s kojima barata korisnik. Crvenkastom bojom prikazana je interakcija korisnika sa simulatorom.

Svaki program napisan u asemblerskom jeziku treba asemblirati kako bi se dobila izvršna datoteka. Ovaj postupak opisan je u drugom dokumentu koji opisuje aplikaciju SSPARCSS asemblerskog prevoditelja. U simulator treba učitati model računala s kojim želimo raditi (npr. Intel ili ARM). U modelu računala se nalaze pod-modeli svih dijelova računala: procesora, memorije, sabirnica, vanjskih jedinica itd. Izvršnu datoteku treba učitati u model memorije. Nakon toga, korisnik pomoću GUI-ja upravlja radom simulatora, pokreće razne akcije i promatra dobivene rezultate te tako dobiva uvid u funkcioniranje procesora i ostalih dijelova računala. Svi primjeri u ovom dokumentu koriste asembler i model za ARM koji se obrađuju na predmetu Arhitektura računala 1E.

2 SSPARCSS simulator

2.1 Osnovno korištenje simulatora

Asemblerski programi su programi pisani u asemblerskom jeziku, naravno za određeni procesor. Njihovim asembliranjem dobivamo izvršnu datoteku s ekstenzijom **.e* ("e" je kratica od engl. *executable file*). Ova datoteka je najvažniji ulazni podatak koji korisnik predaje simulatoru, jer će simulator izvoditi ovu izvršnu datoteku, a to predstavlja temelj rada svakog računala: izvođenje programa na procesoru.

Osim izvršne datoteke, korisnik upravlja radom simulatora slično kao što se upravlja radom bilo kojeg ispravljača grešaka (engl. *debugger*) u bilo kojem integriranom radnom okruženju za razvoj programa. Akcije koje stoje na raspolaganju su pokretanje i zaustavljanje programa, izvođenje korak-po-korak, postavljanje prekidnih točaka, pregledavanje stanja modela što uključuje registre u procesoru, lokacije u memoriji, stanja priključka i sabirnica, stanja raznih vanjskih jedinica itd.

Stanje modela korisniku se prikazuje u grafičkom obliku, pomoću različitih animacija i naravno numerički, jer računala u svojoj osnovi rade s binarnim podacima (koje najčešće promatramo kao brojeve).

2.2 Pokretanje simulatora

Simulatorska se aplikacija može pokretati na sve uobičajene načine: klikom na ikonu simulatorske aplikacije, izravnim pozivom iz komandne linije, ili klikom na ikonu datoteke s ekstenzijom **.system*.

U praksi je najpogodnije i preporučeno pokretanje **klikom na ikonu **.system***. Ova datoteka je zapravo model sustava pisan u posebnom modelacijskom jeziku COMDEL v.2. Slično kao kod asemblerske aplikacije, koja ima konfiguracijsku datoteku **.assembler*, tako i datoteku **.system* možemo promatrati kao svojevrsnu konfiguracijsku datoteku za simulator. U ovom dokumentu ćemo često koristiti pojam sistem misleći zapravo na model, jer model je reprezentacija nekog stvarnog računalnog sistema s kojim radimo. Također ćemo često koristiti pojam konfiguracija ili konfiguracijska datoteka, misleći pritom na model i datoteku s opisom modela, jer model na neki način konfigurira simulatorsku aplikaciju.

Preporuča se da korisnik obje konfiguracijske datoteke skupa sa svim svojim asemblerskim programima i izvršnim datotekama drži u **jednom direktoriju**, koji nazivamo **radnim direktorijem**. Tada će se prilikom rada s assemblerom i simulatorom automatski učitavati potrebne konfiguracijske datoteke bez potreba za dodatnim intervencijama korisnika. Ako korisnik koristi neki drugi raspored datoteka, onda konfiguracijske datoteke mora ručno učitati sam. Također, ako se aplikacija pokreće ikonom same aplikacije, onda aplikacija ne može znati koji radni direktorij se želi koristiti pa opet treba ručno učitavati konfiguracijske datoteke.

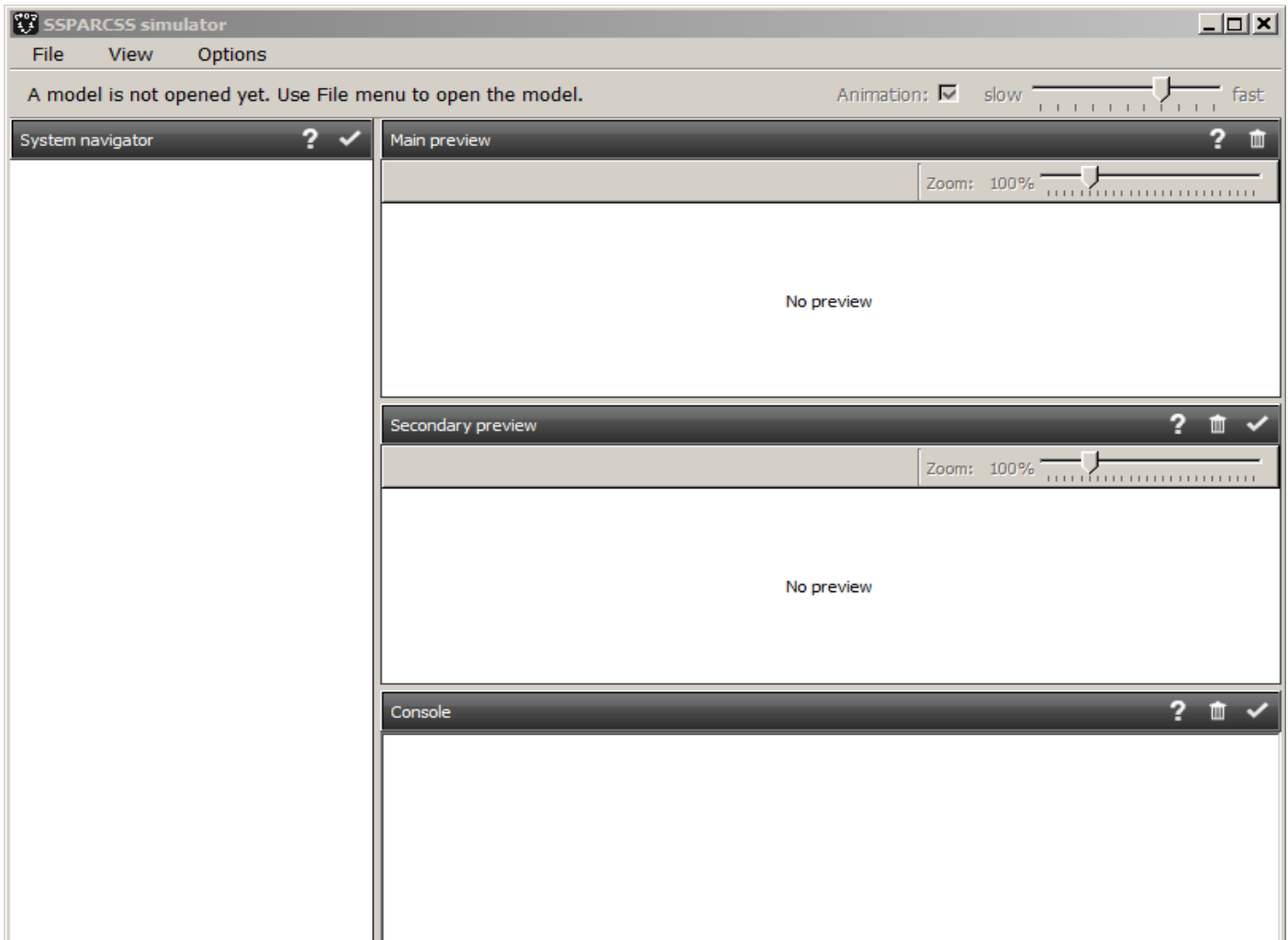
Dakle, najpogodnije je pokretanje simulatora klikom na **.system* datoteku pri čemu koristimo preporučeni raspored datoteka u radnom direktoriju.

2.2.1 Pokretanje simulatora i obnavljanje postavki

Simulator **prilikom pokretanja** pokušava u **trenutačnom radnom direktoriju** pronaći datoteku **simulator.ini** u kojoj se pamte razne postavke simulatora (položaji prozora, postavke zadane sa izbornikom *Options*, različite postavke vezane za sučelje koje je korisnik podesio tijekom rada). Ako je datoteka **pronađena**, postavke se **automatsku obnavljaju**. Ako se simulator pokreće **ikonom aplikacije**, onda radni direktorij ne postoji i **postavke se ne obnavljaju**, nego se postavljaju na *default* vrijednosti. Kako sučelje simulatora ima mnogo elemenata (pamti se oko 50 postavki) koje će korisnik namjestiti prema svojem ukusu, preporučeni način pokretanja svakako bitno olakšava korištenje simulatora. Pri **izlasku** iz simulatora, u istu datoteku se **spremaju trenutačne postavke**.

3 Pregled sučelja SSPARCSS simulatora

Sučelje simulatora ima raspored glavnih dijelova vrlo sličan uobičajenim aplikacijama s nekim malim razlikama. Na vrhu se nalazi traka s izbornicima (engl. *menubar*), kao što je uobičajeno, ali statusna traka se nalazi odmah ispod, jer su statusne poruke u simulatoru veće važnosti nego u većini drugih aplikacija. Traka s alatima (engl. *toolbar*) se *defaultno* ne nalazi ispod izbornika (iako je korisnik može tu postaviti), nego se nalazi u polju *Debug* (koje se otvara naknadno). Glavni prozor sastoji se od mnogobrojnih "polja", koja će biti postupno opisana kroz ovaj dokument.



Slika prikazuje mogući izgled simulatora koji je pokrenut aplikacijskom ikonom, tako da model nije automatski učitani pa su zbog toga sva polja u prozoru prazna. U ovakvom slučaju korisnik mora sam pomoću izbornika *File* učitati željeni model (ovo je opisano u poglavlju 5.1.1). Međutim, trenutno su od interesa osnovni dijelovi prozora. Na vrhu se nalazi već spomenuta **traka s izbornicima** koji su kasnije detaljnije opisani, a ovdje je samo kratak opis:

- *File* - izbornik koji služi za učitavanje modela i ima uobičajenu stavku za izlazak iz aplikacije
- *View* - izbornik koji služi za prikazivanje i sakrivanje pojedinih polja u glavnom prozoru
- *Options* - izbornik koji služi za otvaranje prozora za namještanje opcija

Ispod izbornika je **statusna traka** koja opisuje trenutno stanje simulatora i logičnu akciju koja se očekuje od korisnika. Na desnoj strani statusne trake je klizač za namještanje brzine simulacije.

Ispod statusne trake su **četiri glavna polja** koja uvijek postoje u simulatoru, bez obzira na učitani model i na korisnikove akcije. Raspored glavnih polja je fiksni, a korisnik može jedino odabrati hoće li glavna polja biti prikazana ili sakrivena.

Dodatna polja se otvaraju ovisno o modelu i korisnikovim akcijama. Ona imaju jednak izgled kao i glavna, ali ih korisnik može slobodno **razmještati** s lijeve, desne i donje strane od glavnih polja. Korisnik također može "**odvojiti**" dodatna polja od glavnog prozora i tada to više nisu polja nego nezavisni prozori. Razmještanje i odvajanje korisnik postiže tako

da mišem uhvati gornju crnu naslovnu traku i odvuče polje na željeno mjesto (uobičajeni *drag-and-drop*). Za razliku od toga, glavna polja imaju fiksne položaje i ne mogu se pomicati - ona čine središte glavnog prozora oko kojega se razmještaju dodatna polja.

Svim poljima korisnik može namještati veličinu na uobičajen način: mišem se uhvati rub polja i pomakne se u željenom smjeru. Od svih navedenih polja, sa stajališta krajnjeg korisnika neka će se upotrebljavati uvijek, neka samo povremeno, a neka će biti potrebna vrlo rijetko ili nikad.

Svako polje ima traku s naslovom u kojoj je nekoliko ikona (crna traka na gornjem rubu svakog polja). Naslovna traka opisuje namjenu svakog polja, a ikone na njoj desnoj strani omogućuju četiri akcije:

- **Ikona upitnika:** prikaz kratkih uputa vezanih za pripadno polje (akcije u polju, značenja prikaza u polju, itd.).
- **Ikona kante za smeće:** brisanje prikaza u polju, polje ostaje na svojem mjestu.
- **Ikona kvačice:** sakrivanje polja, polje i dalje postoji ali se ne prikazuje (može se ponovno prikazati korištenjem izbornika [View](#) u kojemu je popis svih polja koja se mogu sakriti).
- **Ikona križića:** brisanje polja iz simulatora, polje se potpuno uklanja i ne može se ponovno prikazati. Ova ikona nije prikazana na gornjoj slici jer glavna polja ne mogu biti obrisana.

Glavna polja detaljnije su opisana naknadno, a ovdje je kratak opis njihove namjene:

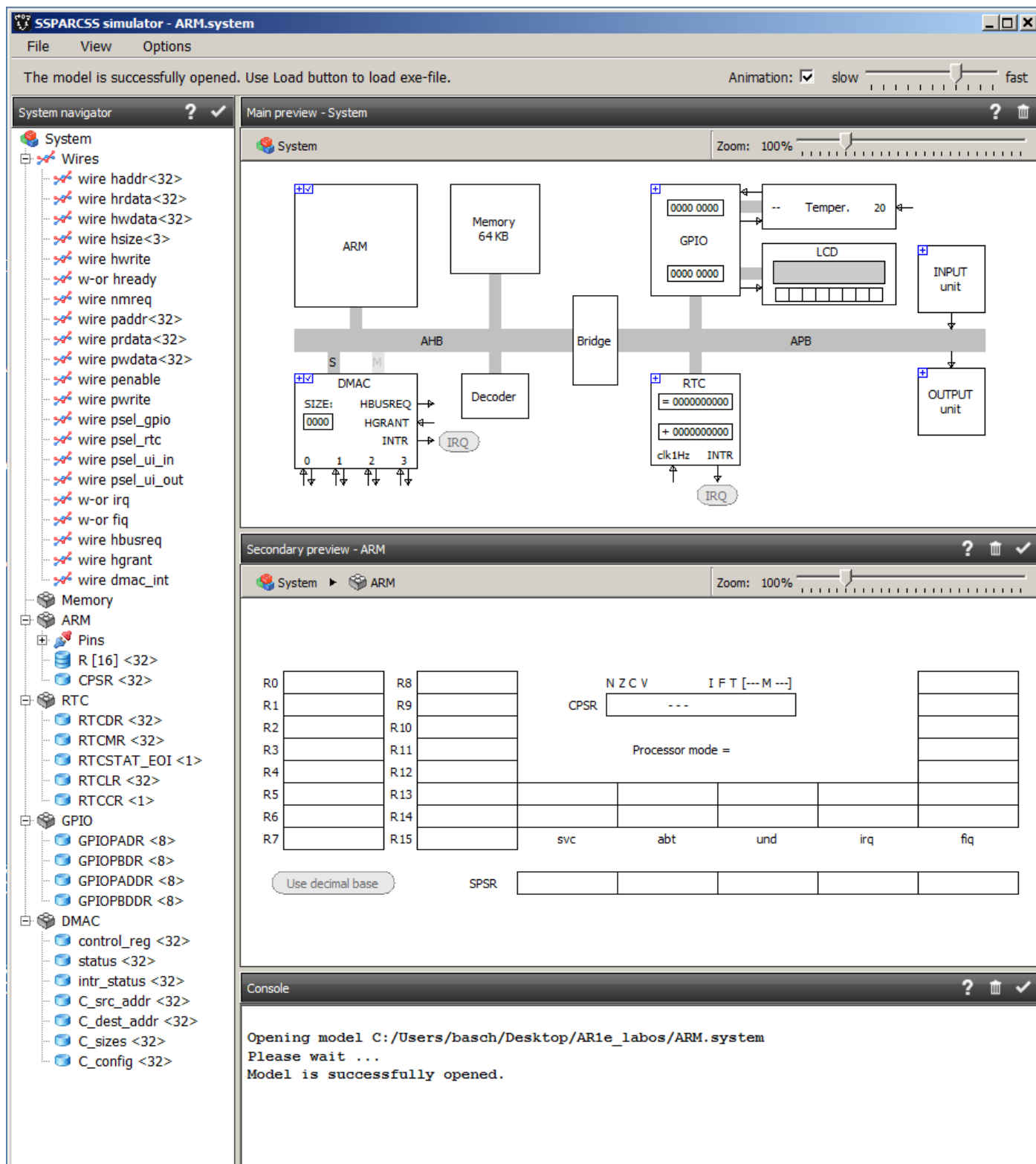
- **[System navigator](#)** ili kraće **[Navigator](#)**: hijerarhijski/stablasti **prikaz cijelog modela i svih njegovih dijelova** dostupnih korisniku. Ovisno o modelu, upotrebljivost ovog prozora može biti veća ili manja, a za model procesora ARM će se rijetko koristiti pa se može sakriti da ne zauzima mjesto na ekranu.
- **[Preview](#)**: jedno od dva najvažnija polja za prikaz stanja modela. U poljima **[Preview](#)** se prikazuje **animacija** rada pojedinih dijelova računala. Osim glavnog i pomoćnog polja **[Preview](#)**, korisnik po želji može otvoriti i dodatna polja **[Preview](#)**.
 - **[Main preview](#)**: glavno animacijsko polje koje je uvijek prikazano. Jedino polje koje korisnik ne može čak niti sakriti (uočite da nema ikonu kvačice). U njemu se automatski prikazuje animacija modela nakon učitavanja modela.
 - **[Secondary preview](#)**: pomoćno animacijsko polje, koje ima istu namjenu kao i glavno, a koristi se kada uz animaciju u glavnom polju želimo paralelno pratiti animaciju još nekog dijela sustava.
- **[Console](#)**: **tekstualni ispis poruka i stanja simulacije**. Ovisno o modelu i postavkama koje zada korisnik, dio ispisa stanja može se odvijati u polju **[Console](#)**. U tom slučaju, polje **[Console](#)** treba držati prikazanim. Međutim, modeli su obično koncipirani tako da što više koriste animacije i da minimiziraju tekstualni prikaz stanja modela. U tom slučaju treba sakriti polje **[Console](#)** da ne zauzima mjesto na ekranu. Ako dođe do greške u simulaciji, u polju **[Console](#)** je koji puta detaljniji opis greške nego što se prikazuje u standardnom *popup*-prozoru za grešku.

Općenito, sučelje je koncipirano tako da bude **minimalističko**, sa što manjim brojem izbornika, alatnih traka i različitih gumbi. Osim želje za što većom jednostavnošću, drugi bitan razlog za ovakvo sučelje je taj da sučelje jednim dijelom ovisi o modelu, koji može biti bilo kakav i ograničen je samo maštom osobe koja izrađuje model. Također, model računala je relativno složen jer su i sama računala složena. Zato se korisniku ostavlja mogućnost da detaljnije prikazuje određene dijelove modela, a druge ne mora uopće prikazivati. Sučelje je **prilagodljivo** s različitim stajalištima. Korisnik može utjecati primjerice na:

- broj i vrstu polja u kojima se prikazuje stanje modela
- raspored i veličinu polja u kojima se prikazuje stanje modela
- brojeve baze u kojima se prikazuju neki od rezultata (ne svi)
- odabir dijelova modela koji se prikazuju
- veličinu prikaza i fontove koji se koriste za prikaz
- položaj trake s alatima i veličinu ikona u traci
- ponašanje simulatora u određenim okolnostima

Većinu akcija korisnik zato zadaje preko različitih **kontekstnih izbornika** koji su prilagođeni pojedinim poljima. Dio općenitih postavki korisnik zadaje pomoću opcija dostupnih u izborniku **[Options](#)** (ovo je opisano u poglavlju 8).

Sljedeća slika ilustrira kako može izgledati glavni prozor nakon preporučenog pokretanja - klikom na konfiguracijsku datoteku. Isti izgled može se dobiti ako se datoteka učitava ručno pomoću izbornika [File](#). Na ovoj slici je u pomoćnom polju [Preview](#) ([Secondary preview](#)) dodatno otvoren prikaz unutrašnjosti procesora ARM - vide se pojedini registri.



Vidljiva su pojedina glavna polja tijekom korištenja simulatora: stablasti prikaz dijelova modela u polju [Navigator](#), poruka u polju [Console](#), animacijski prikaz cijelog modela ([System](#)) i procesora ([ARM](#)) u poljima [Preview](#). Sadržaj glavnih polja ovisi: o modelu (za neki drugi model izgledat će drugačije), o tome što je korisnik odabrao da se prikazuje u poljima, te o stanju modela (prije simulacije, u tijeku simulacije, poslije simulacije, nakon greške u simulaciji itd.).

Nakon ovog preglednog prikaza glavnog prozora, u daljnjim poglavljima slijede detaljniji opisi pojedinih polja i dijelova sučelja simulatora te korištenja simulatora.

4 Osnovni tijek rada u simulatoru

Ovo poglavlje pokušava ukratko objasniti osnovnu koncepciju simulatora i logični slijed akcija koje korisnik treba poduzimati da bi na efikasan i smislen način upravljao simulacijom. Nažalost, dosta je teško objasniti neke osnovne pojmove prije nego se detaljnije objasne pojedine mogućnosti simulatora, a također je teško objasniti pojedine mogućnosti ako prije toga nisu objašnjeni osnovni pojmovi. Zbog ovih kontradiktornih zahtjeva, objašnjenje nije idealno. Zato se ovdje pokušava dati "šira slika" u obliku niza pravila/recepata, ali će neke stvari možda postati jasnije tek naknadno kad se prouče detaljniji opisi pojedinih mogućnosti simulatora.

Kao prvo, pretpostavimo da krajnji korisnik neće sam pisati svoje modele, nego će koristiti već gotove, jer rad u simulatoru je bitno složeniji prilikom modeliranja.

- Da bismo bilo što mogli raditi sa simulatorom, **prvo treba učitati neki model** - bilo ručno, bilo tako da zadamo model već prilikom pokretanja (što je praktičnije).
- Uz pretpostavku da ne želimo mijenjati model s kojim radimo (a u praksi ćemo na predmetu AR1e koristiti samo jedan model), tada više **ne moramo ponavljati učitavanje modela**. Ponovno učitavanje modela mogli bi usporediti kao kad bismo prije svake nove rečenice koju želimo utipkati u *Wordu*, ponovno učitali cijeli dokument - to bi bilo nepotrebno ponavljanje.
- Simuliranje će uvijek biti **izvođenje nekog asemblerskog programa** kojega smo sami napisali i asemblirali.
 - To može biti **jednostavni program** koji izvodi nekoliko operacija nad podacima u memoriji. Takav program će se izvesti praktički **trenutačno** i nakon toga će biti **zaustavljen naredbom SWI**.
 - To također može biti neki **dugotrajniji program** koji upravlja vanjskim jedinicama, prenosi podatke između njih i memorije i obavlja to **kontinuirano** sve dok ga sami ne **zaustavimo gumbom Pause**.
 - To također može biti program koji tek razvijamo i koji još uvijek **ne radi dobro** pa izaziva razne vrste grešaka ili se jednostavno ne ponaša ispravno. Tada možemo koristiti **izvođenje korak-po-korak** da bismo detaljno pratili izvođenje programa u cilju otkrivanja greške.
- Bez obzira o kojem od tri gornja slučaja se radi, simulacija se uvijek odvija u tzv. **rundama**. To znači da ćemo **izvođenje istog/izmijenjenog programa ponavljati više puta**. Često ćemo pritom popravljati uočene greške i ponavljati izvođenje.
- **Upravljanje simulacijom** ostvaruje se **alatnom trakom** polja **Debug**.
- Važno je shvatiti da se **svaka runda simuliranja uvijek pokreće tako da se sustav prvo inicijalizira, a zatim se pokrene simulacija**.
 - **Između dviju rundi** često želimo napraviti **preinake ili popravke na asemblerskom programu**. U tu svrhu koristimo gumb **Edit** koji **poziva asemblerski prevoditelj** i nakon asembliranja nas **automatski vraća u simulator**.
- **Inicijalizacija** se odvija uglavnom nevidljivo i automatski, osim jedne stvari koju **moramo napraviti eksplicitno**. To je **učitavanje izvršne datoteke** u memoriju modela. Učitavanje se zadaje gumbima **Load** ili **Reload**.
 - Ako smo koristili gumb **Edit**, onda će **automatski doći do inicijalizacije i ponovnog punjenja izvršne datoteke** čim završimo s radom u assembleru.
- **Nakon inicijalizacije se može pokretati simulacija u normalnom izvođenju (Run) ili korak-po-korak (Step) ili naizmjenično**. Dok je simulacija u normalnom izvođenju, možemo je **pauzirati gumbom Pause**.
- Također možemo postavljati **prekidne točke** na pojedine naredbe u asemblerskom programu i nailaskom na njih izvođenje će se automatski **pauzirati**.
- Kad je simulacija **pauzirana**, uvijek je možemo **nastaviti gumbima Run ili Step**.
- Ako želimo započeti **novu rundu** simuliranja, moramo obavezno **ponovno učitati izvršnu datoteku** pomoću **Load** ili **Reload**.
- Ako je simulacija **dovršena naredbom SWI ili prekinuta uslijed greške**, onda je **ne možemo nastaviti sa Run ili Step**. Tada je moguće samo započeti **novu rundu** simuliranja.
- Dok je simulacija pauzirana ili dok se izvodi, možemo **uključivati i isključivati pojedine prikaze i animacije** i tako pregledavati stanje i pratiti tijek simulacije.

5 Dijelovi glavnog prozora

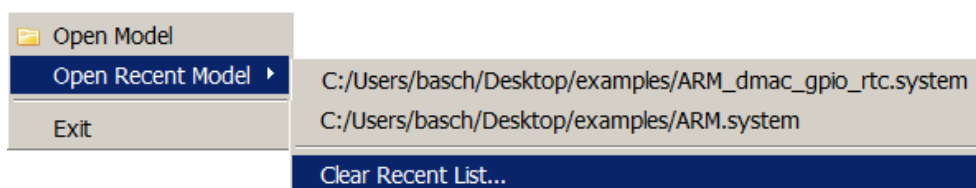
Ne računajući razna polja, glavni prozor sadrži samo traku s izbornicima, alatnu traku koja je opcionalno prikazana, i statusnu traku. Alatna traka za modele s jednom procesorom sadrži iste stavke kao i alatna traka polja [Debug](#), pa za detaljniji opis treba pogledati potpoglavlje 6.2.4. Polja glavnog prozora opisana su u sljedećim poglavljima 6 i 7.

5.1 Izbornici

Izbornici simulatora nude minimalan broj mogućnosti. Postoje samo tri izbornika: [File](#), [View](#) i [Options](#).

5.1.1 Izbornik File

[File](#) je uobičajeni izbornik koji u simulatoru nudi samo **otvaranje datoteke s novim modelom** i izlazak iz aplikacije. Prikazan je slikom:



Stavka [Open](#) model otvara dijaloški prozor za odabir nove datoteke s ekstenzijom [*.system](#). Također je moguće odabrati jedan od deset zadnjih učitanih modela pomoću stavke [Open Recent Model](#). Moguće je obrisati listu zadnjih modela korištenjem stavke [Clear Recent List](#). Na laboratorijskim vježbama koristit će se samo jedan model, tako da sve ove opcije nisu od prevelike koristi. Također je bolje koristiti preporučeni način pokretanja klikom na ikonu datoteke s ekstenzijom [*.system](#) i tada izbornik [File](#) skoro da nije ni potreban.

Aplikacija se može zatvoriti na standardan način: odabirom stavke [Exit](#) u izborniku [File](#) ili klikom na "X-ikonu" u gornjem desnom kutu glavnog prozora. Ako je u tijeku simulacija, korisnik se o tome obavještava i mora potvrditi da želi prekinuti simulaciju.

5.1.2 Izbornik View

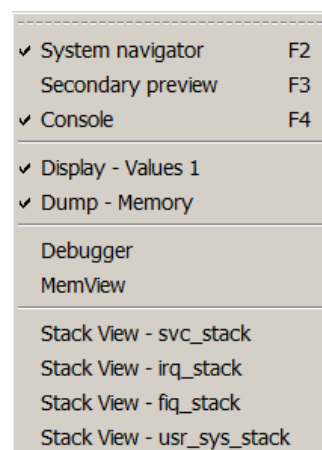
Sljedeći izbornik je [View](#). Njegova namjena je isključivo **sakrivanje i prikazivanje svih polja** koje trenutno postoje, to uključuje glavna polja i sva dodatna polja koja je stvorio korisnik. Dakle, sadržaj izbornika [View](#) mijenja se ovisno o postojećim poljima. Izbornik je prikazan slikom.

Na početku su prikazana stavke za glavna polja: [Navigator](#), pomoćni [Preview](#) i [Console](#) s pripadnim kraticama F2, F3 i F4 koje se mogu koristiti za njihovo brzo sakrivanje i prikazivanje.

Ispod toga se nalaze stavke za dodatna polja koja je otvorio korisnik (na slici su to jedno polje [Display](#) i jedan [Dump](#)).

Na dnu su stavke za polja koja ovise o otvorenom modelu i uvijek su u redoslijedu: [Debugger-MemView-StackView](#).

Svako polje ima jednu stavku u izborniku. Na lijevoj strani stavke je **kvačica** koja označava da je polje **prikazano**. Klikom na stavku se mijenja prikazanost/sakrivenost polja i pripadna kvačica.



Sva polja se mogu sakrivati pomoću ikone/gumba "kvačice" u svojoj naslovnoj traci, no kad su jednom sakrivena moguće ih je **ponovno prikazati jedino korištenjem izbornika [View](#)**. Uklanjanje polja iz simulatora (za polja za koja je to moguće) ostvaruje se isključivo ikonom/gumbom "križić" u naslovnoj traci. Izbornik [View](#) ne nudi mogućnost uklanjanja polja iz simulatora.

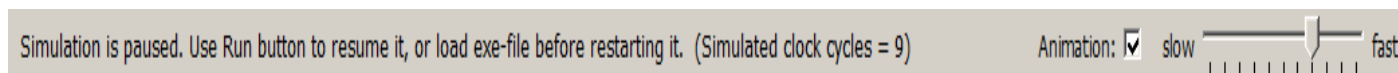
Ako se mišem klikne na vrh izbornika [View](#), na isprekidanu horizontalnu crt, onda se izbornik odvaja (*tear-off*) i postaje mali privremeni nezavisni prozor, što je zgodno ako želimo prikazivati i/ili sakrivati više polja odjednom.

5.1.3 Izbornik Options

Zadnji izbornik *Options* je trivijalan jer ima samo jednu stavku *Options*. Ona otvara zasebni prozor za namještanje raznih opcija što je opisano u poglavlju 8.

5.2 Statusna traka

Statusna traka nalazi se odmah ispod trake s izbornicima. U njoj se ispisuje uobičajeni status aplikacije i očekivana akcija od strane korisnika. U slučaju grešaka i upozorenja statusna traka zatreperi, a može davati i zvučni signal (ovisno o namještenim opcijama). Kad je simulacija pauzirana ili je završila, ispisuje se i broj simuliranih ciklusa.



Na desnoj strani statusne trake nalazi se tzv. *checkbox Animation* kojim se mogu posve **isključiti animacije** u poljima *Preview*. Ovo će rijetko biti potrebno, ali može donekle ubrzati simulaciju, tj. izvođenje asemblerskog programa.

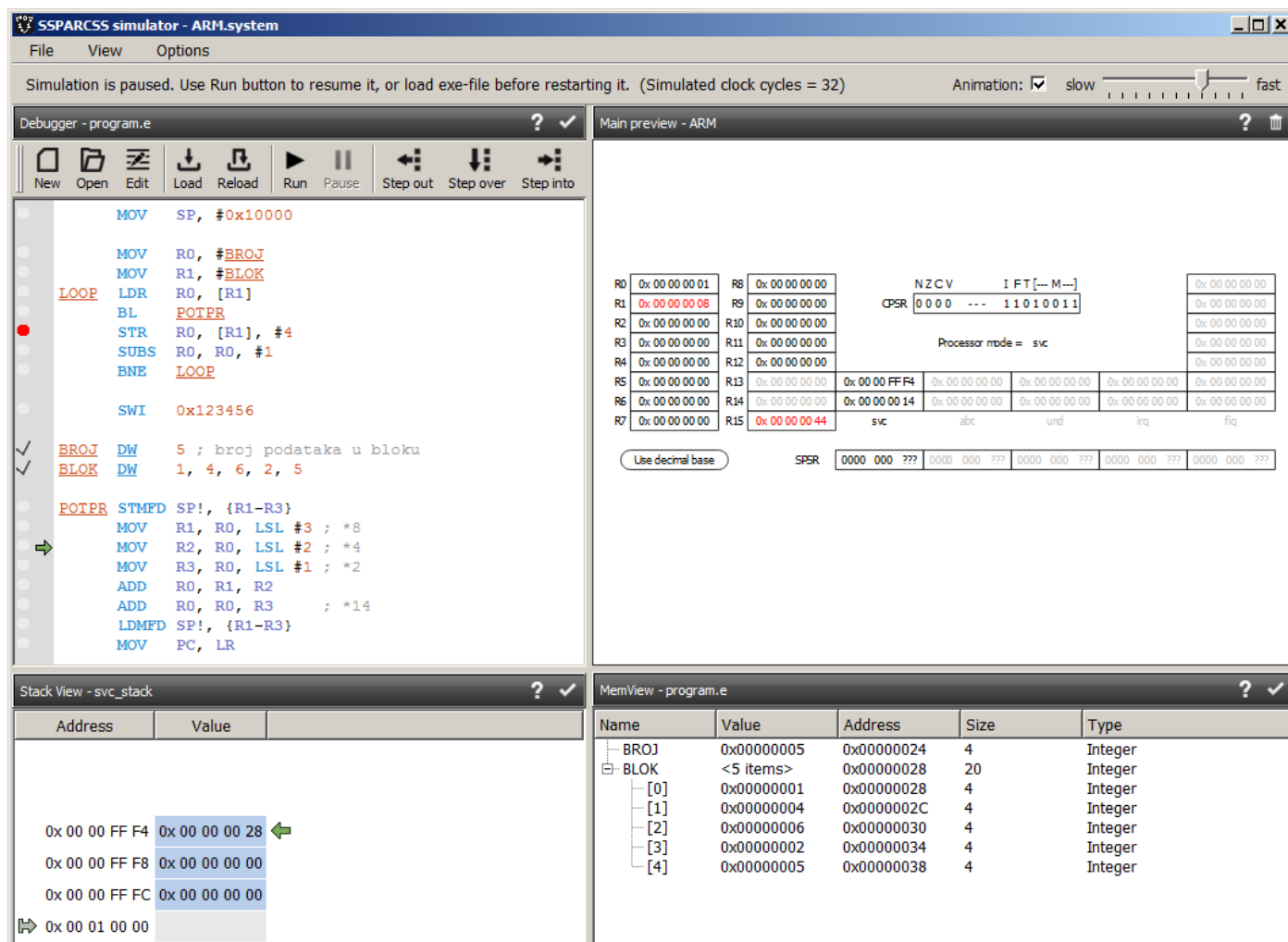
Desno od *checkboxa* je **klizač** (engl. *slider*) za **zadavanje brzine animacije/simulacije** s opisnim naslovima *slow* i *fast*. Iako konkretne brojke nisu bitne, pozicioniranjem miša iznad klizača dobiva se točnije trajanje pauze koja se umeće u simulaciju na svaku četvrtinu signala *clock*. Najveći iznos je 1 sekunda, najmanji 0 sekundi, ali tri najbrže postavke zapravo preskaču određeni broj animacijskih koraka kako bi se dobila što veća brzina simulacije (nauštrb preciznosti animacije, ali u toj brzini rada to ionako nema efekta jer su promjene prebrze za ljudsko oko).

Brzina simulacije će u praksi obično biti postavljena na najveću moguću brzinu, a tek kad se pokušava pratiti kontinuirani rad simulacije prateći animaciju u polju *Preview*, i kada je animacija prebrza, tek onda ima smisla postupno smanjivati brzinu dok je ne namjestimo dovoljno sporo da je možemo pratiti.

U vezi s brzinom simulacije treba reći da, nasuprot očekivanju mnogih korisnika, ona nema nikakve veze s brzinom modela, tj. procesora u modelu. Brzina simulacije ovisi prije svega o složenosti modela i snazi vašeg računala. Simulacija je **znatno sporija** od stvarnog sustava kojega modeliramo. Procesori, čiji rad simuliramo, rade na brzinama od cca 100-injak megahertza do par gigahertza. Simulacija može simulirati svega 10,000-50,000 ciklusa *clocka* u sekundi (ovisno o složenosti modela i o snazi vašeg računala, kako je već rečeno). Ona može, dakle, biti i milijun puta sporija od stvarnog sustava.

6 Najvažnija polja u sučelju

Ovo poglavlje objašnjava četiri najvažnije vrste polja u sučelju simulatora. Ona se koriste uvijek (ili skoro uvijek), pri svakoj simulaciji rada procesorskog sustava. Glavni prozor s ova četiri polja prikazan je slikom. Kao što je već rečeno, razmještaj dodatnih polja je proizvoljan, kao i njihova veličina. Jedino se [Main preview](#) uvijek nalazi u sredini glavnog prozora (ovdje s desne strane od [Main preview](#)-a korisnik nije razmjestio niti jedno od dodatnih polja).



Glavna namjena polja je upravljanje tijekom simulacije i/ili prikaz stanja modela. Na slici se vidi:

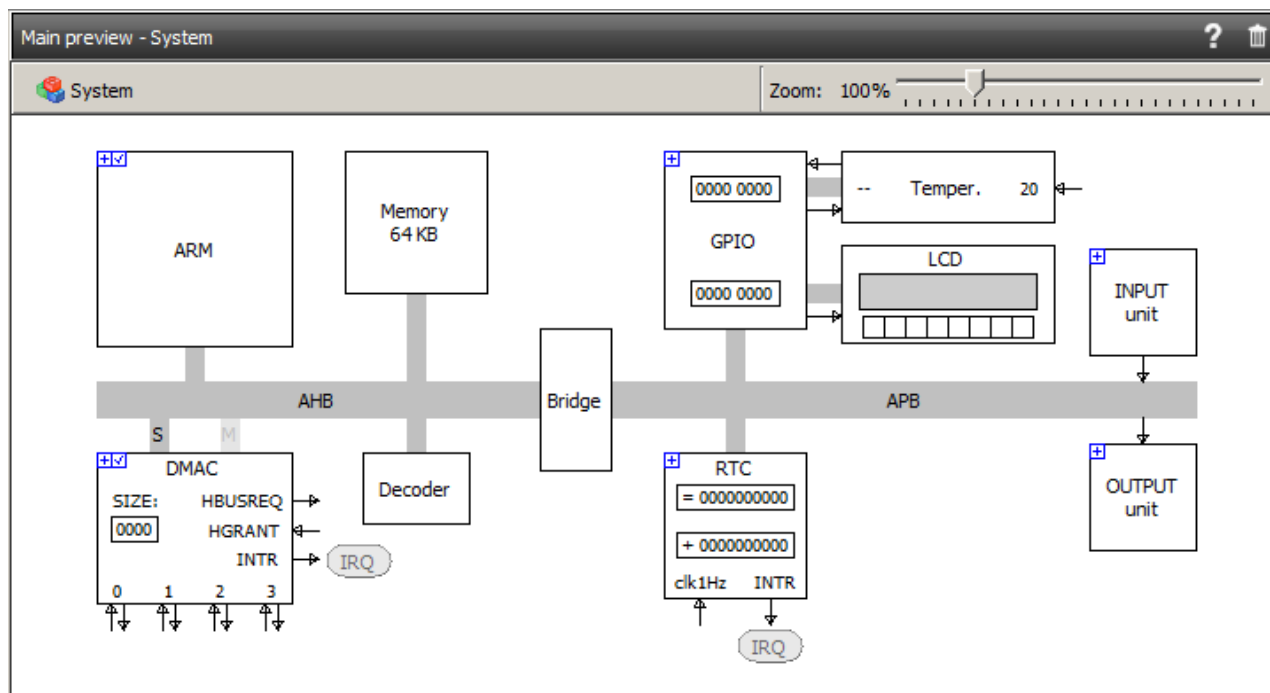
- Polje **Preview**: **animacija** odabranog dijela modela (kao što je već spomenuto ranije).
- Polje **Debugger** ili kraće polje **Debug**: **prikaz asemblerskog programa** koji se trenutno izvodi i **upravljanje** njegovim izvođenjem. Uz **Preview**, ovo je najvažnije polje simulatora.
- Polje **MemView**: **prikaz podataka** na odabranim **memorijskim lokacijama**.
- Polje **StackView**: **prikaz stanja** na **stogu** u memoriji.

6.1 Polje Preview

Polje [Preview](#) prvenstveno služi za **prikaz i animaciju** rada **pojedine komponente** u modelu. Također ima i neke jednostavnije akcije vezane za dijelove modela, a većina akcija služi prilagodbi prikaza. Izgled polja [MemView](#) posve je različit u ovisnosti o komponenti koja se trenutačno prikazuje. Polje može biti i prazno, ako ne postoje komponente ili ako smo obrisali polje pomoću ikone ("kanta za smeće").

6.1.1 Polje Preview prilikom prikaza modela

Već smo pokazali i ukratko objasnili glavno i pomoćno polje (*Main* i *Secondary Preview*) i spomenuli da se može otvoriti proizvoljan broj dodatnih polja *Preview*. Sva ova polja imaju istu funkcionalnost i akcije. Pokažimo sliku polja *Preview*



koje prikazuje **glavnu ili korijensku komponentu** sustava koja sadrži sve ostale komponente - zapravo se prikazuje **cijeli model**, koji se u simulatoru vidi pod imenom *System*.

Na vrhu vidimo već poznatu naslovnu traku koja ispisuje ime trenutno prikazane komponente. Budući da je ovo glavno polje, na raspolaganju su samo ikone "upitnika" i "kante za smeće". Pomoćno polje ima dodatno i ikonu "kvačice" jer se ono može sakriti. Dodatna polja imaju još i ikonu "križića" kojom se polje može ukloniti iz simulatora.

Ispod naslovne trake se nalazi alatna traka koja nema gumbe kao što je uobičajeno, nego je prilagođena polju *Preview*. S lijeve strane nalazi se tzv. *breadcrumbs*, tj. niz komponenata u kojima se nalazi trenutno prikazana komponenta - nešto poput puta u kojem se nalazi neka datoteka. Za slučaj korijenske komponente na gornjoj slici, prikaz je trivijalan i sastoji se samo od te komponente. Na desnoj strani se nalazi klizač kojim je moguće namještati veličinu prikaza u polju. Alatna traka može se sakriti pomoću kontekstnog izbornika polja *Preview*, što je preporučljivo (jer modeli u simulatoru su najčešće "plitki" i nemaju puno razina ugniježđenih komponenata, a veličina se može namještati i na druge načine).

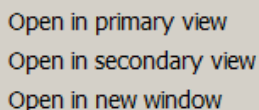
Glavni dio polja *Preview* je površina za prikaz modela. Sam prikaz ovisi o modelu i trenutnoj komponenti koja se prikazuje, ali postoje neke konvencije koje su vidljive na gornjoj slici. Komponenta kao što je korijenska sadrži u sebi **druge komponente** koje prikazuje kao obične pravokutnike, s eventualnim "dekoracijama" u vidu natpisa, strelica koje predstavljaju priključke, jednostavnijih prikaza nekih unutrašnjih dijelova komponente itd. Na slici vidimo komponente: *ARM*, *Memory*, *GPIO*, *Temper.*, *LCD*, *INPUT unit*, *OUTPUT unit*, *DMAC*, *Decoder*, *Bridge* i *RTC*. **Sive veze** između komponenata su **sabirnice AHB** i **APB**.

6.1.2 Kontekstni izbornici polja Preview

Sve akcije dostupne su iz **kontekstnih izbornika** koji ovise o tome na koji dio prikaza je kliknuto. Dio akcija dostupan je i putem kratice ili izravnim korištenjem miša.

6.1.2.1 Kontekstni izbornici komponentata u polju Preview

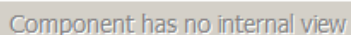
Ako želimo u polju [Preview](#) početi prikazivati jednu od navedenih komponentata, jednostavno na njoj napravimo dvostruki klik mišem. Hoće li se nešto dogoditi - to ovisi o tome da li kliknuta komponenta ima definiran interni prikaz. Zbog toga sve komponente koje imaju interni prikaz imaju u svojem gornjem lijevom uglu malu plavu ikonu - "pravokutnik sa znakom plus". Ako se otvori kontekstni izbornik na takvoj komponenti onda on izgleda otprilike ovako:



Open in primary view
Open in secondary view
Open in new window

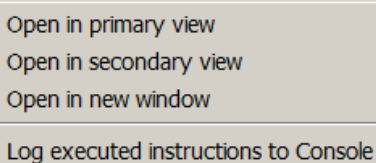
Akcije koje se nude su prikazivanje kliknute komponente u glavnom ili pomoćnom polju ili stvaranje novog dodatnog polja. Ako glavno ili pomoćno polje već sadrže neki prikaz, odabir nove komponente jednostavno će zamijeniti postojeći prikaz.

Ako se napravi dvostruki klik na komponenti koja nema interni prikaz, neće se dogoditi ništa (osim zvučnog signala upozorenja). Ako se otvori kontekstni izbornik na takvoj komponenti onda on izgleda otprilike ovako:



Component has no internal view

Neke komponente uz ikonu sa "plusom" imaju još i plavu ikonu "pravokutnik s kvačicom". Ova oznaka znači da komponenta nudi određene postavke koje se mogu uključivati/isključivati pomoću njenog kontekstnog izbornika. Primjerice, ako se otvori kontekstni izbornik od komponente [ARM](#) (procesor), onda on izgleda ovako:



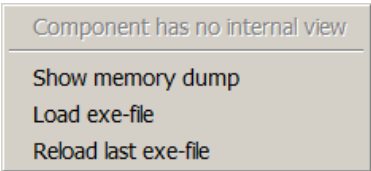
Open in primary view
Open in secondary view
Open in new window
Log executed instructions to Console

Osim prve tri stavke koje su posljedica postojanja ikone s "plusom", ispod njih nalaze se i postavke. Njihov broj i značenje ovise isključivo o tome kako je napravljen model dotične komponente. Primjerice, za procesor je moguće uključiti da se sve izvedene naredbe "logiraju", tj. zapisuju u polje [Console](#). Ovo može biti pogodno prije svega kod nekih ispitivanja rada procesora ili kod

traženja grešaka. Moguća je i kombinacija da komponenta ima samo ikonu s "kvačicom" pa će onda u kontekstnom izborniku imati neosjetljivu prvu stavku, a ispod nje će biti postavke.

6.1.2.2 Kontekstni izbornik memorijske komponente

Na kraju, pokažimo kontekstni izbornik za jednu specifičnu komponentu, a to je memorijska komponenta. Ona ima neke dodatne akcije svojstvene samo memoriji:



Component has no internal view

Show memory dump
Load exe-file
Reload last exe-file

Kao prvo vidi se da memorija nema interni prikaz. Međutim, zbog velikog broja memorijskih lokacija, on ne bi ni bio previše praktičan. Osim toga, u memoriji se nalazi program, podatci te područje stoga, a za njihov prikaz se brinu redom specijalizirana polja [Debug](#), [MemView](#) i [StackView](#) koja su objašnjena u ostatku ovog poglavlja.

Tri akcije specifične za memoriju su:

- [Show memory dump](#): tzv. *dump* memorije, ili prikaz memorije u "sirovom" obliku. Otvara se posebno polje *Dump* (opisano u 7.4) u kojoj se prikazuju sve lokacije u memoriji, ali isključivo u numeričkom obliku, bez naznake radi li se o programu, podacima, stogu, ili pak o neiskorištenim dijelovima memorije. Primjenjivost ovog prikaza je ograničena.
- [Load exe-file](#): učitavanje izvršnog programa, tj. izvršne datoteke u memoriju.
- [Reload exe-file](#): ponovno učitavanje trenutno učitanih izvršnog programa u memoriju.

Akcije istovjetne stavkama [Load](#) i [Reload](#) mogu se zadati i na druge načine i to prvenstveno u alatnoj traci [Debug](#) prozora (vidi potpoglavlje 6.2.4) što je možda praktičniji način za učitavanje izvršnog programa zbog veće vidljivosti gumbi u alatnoj traci.

6.1.2.3 Kontekstni izbornik pozadine polja Preview

Tzv. **pozadinski kontekstni izbornik otvara** se klikom na pozadini, tj. na područjima koje ne zauzimaju komponente i gumbi. Treba napomenuti da su granice komponenata pomalo neprecizne, jer npr. komponenta **DMAC** ima strelice koje "strše" iz tijela komponente pa se područja oko i između strelica također smatraju komponentom. Pozadinski izbornik omogućava neke općenite akcije vezane za namještanje prikaza i "navigaciju" i izgleda ovako:

✓ Preview Toolbar	Ctrl+T
Go one component up	Backspace
Go to the System component	Home
Center the scene	
Zoom in	Ctrl++
Zoom out	Ctrl+-
Reset zoom level	Ctrl+0

Raspoložive akcije u izborniku su:

- **Preview toolbar**: sakrivanje i prikazivanje alatne trake u ovom polju **Preview** (kratica: Ctrl+T)
- **Go one component up**: prelazak u komponentu "iznad", tj. onu koja u sebi sadrži trenutnu komponentu (kratica: Backspace)
- **Go to the System component**: prelazak u najvišu komponentu, **System** ili korijensku komponentu (kratica: Home)
- **Center the scene**: postavlja prikaz komponente u centar polja **Preview** (kratica: Ctrl-double-click, nije prikazana u izborniku)
- **Zoom in**: povećanje prikaza (kratica Ctrl+'plus')
- **Zoom out**: smanjivanje prikaza (kratica Ctrl+'minus')
- **Reset zoom level**: postavljanje veličine prikaza na *defaultnu* vrijednost (kratica Ctrl+0)

Da bi se pojednostavnili kontekstni izbornici, zadnje tri stavke vezane za *zoom* mogu se sakriti. Istovjetne stavke postoje i u **Debug** poljima te u polju **Console** pa sakrivanje vrijedi za sve tri vrste polja odjednom. Sakrivanje se ostvaruje pomoću opcija u izborniku **Options**.

6.1.2.4 Izravne akcije u polju Preview

U polju za prikaz moguće su i tri izravne akcije:

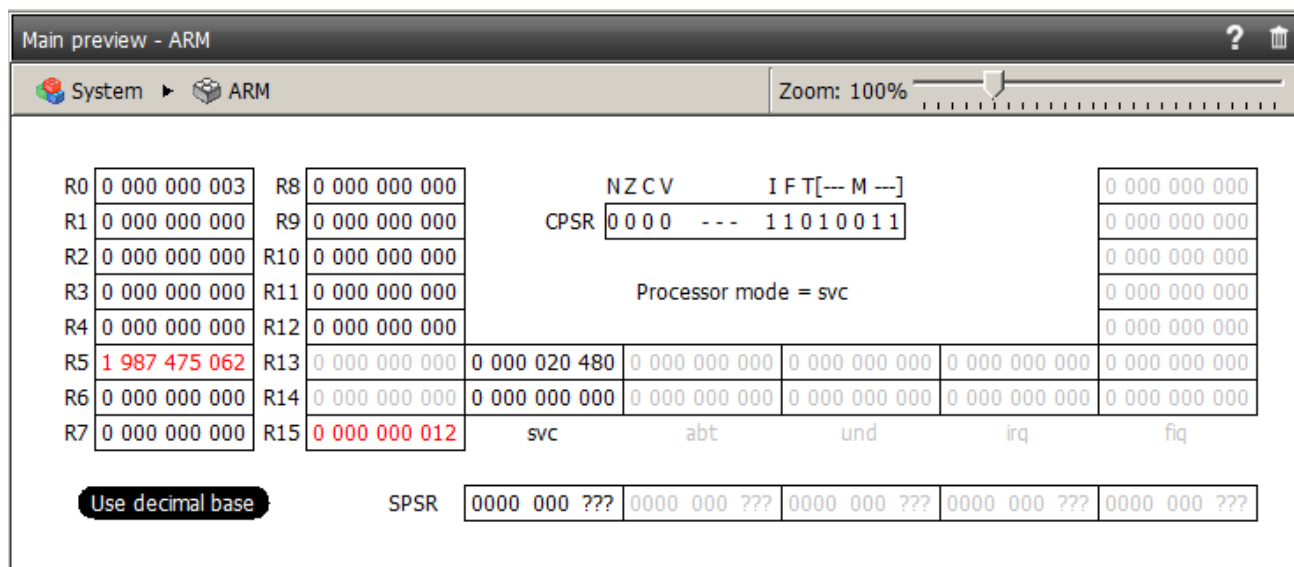
- Već spomenuti **dvostruki klik** na komponentu **otvara prikaz te komponente** (ako njen interni prikaz postoji)
- **Pomicanje cijelog prikaza** lijevim gumbom miša, tako da se **prikaz uhvati mišem** na bilo kojem mjestu, **pomakne** na željeni položaj i ispusti (*drag-and-drop*).
- **Promjena veličine prikaza** pomoću kombinacije **Ctrl-kotačić miša**.

6.1.3 Gumbi u polju Preview

Još jedan dio prikaza su **ovali s natpisom**, kao što su na slici dva siva ovala s natpisom **IRQ** (oni zapravo mogu izgledati bilo kako, ali ovalni oblik je konvencija). Ovo su zapravo **gumbi**, koji su trenutno deaktivirani (sive su boje), i kojima se može prije ili za vrijeme simulacije upravljati određenim zbivanjima u modelu. Konkretno, ova dva gumba omogućuju odabir vrste prekida koji će komponente **DMAC** i **RTC** postavljati **ARM**-u (prekidi mogu biti **IRQ** ili **FIQ**). Odabir vrste prekida u ovom modelu moguć je tek nakon što se model inicijalizira, tj. nakon što se u model napuni izvršni program, a kad jednom započne simulacija, onda ovi gumbi opet postaju neosjetljivi. Neki drugi gumbi u nekom drugom modelu mogu imati drugačije ponašanje, što ovisi o samom modelu - primjerice mogu biti aktivni tijekom cijele simulacije.

6.1.4 Polje Preview prilikom prikaza komponente

Sljedeća slika pokazuje polje *Preview* kad je u njemu odabrana procesorska komponenta (*ARM*):



U prikazu sada **nema drugih komponenata**, jer ih procesor ne sadrži. Umjesto toga prikazuju se neki od **dijelova procesora** - konkretno registri. Ako se desnim mišem klikne na područje koji zauzimaju dijelovi procesora, dobiva se sljedeći kontekstni izbornik:

Log executed instructions to Console

Zapravo se radi o dijelu kontekstnog izbornika procesora kojega smo već vidjeli - prikazane su samo **postavke** koje se mogu zadavati za procesor. Stavki vezanih za prikaz poljima *Preview* nema jer ne bi imale smisla (jer komponenta već jeste prikazana).

I ovdje je područje procesora prilično neprecizno - čak i svi dijelovi između registara promatraju se kao dio procesora koji je omeđen jednim zamišljenim pravokutnikom koji obuhvaća sve dijelove crteža. I ovdje se može otvoriti pozadinski kontekstni izbornik kada se klikne negdje izvan tog zamišljenog pravokutnika. Izbornik izgleda potpuno isto onome koji je gore već objašnjen za pozadinu prikaza komponente *System*.

Na gornjoj slici polja *Preview* vidimo još jednu konvenciju - **crvena boja** se koristi da bi se naznačili dijelovi koji su se nedavno **promijenili**. Ovdje su to sadržaji registara. Ova konvencija koristi se na mjestima gdje su promjene relativno rijetke, a prikaz je "kompliciran" pa bi promjene mogle proći nezapaženo. Tamo gdje su promjene vidljivije, crvena boja se ne koristi kako se korisniku ne bi nepotrebno odvlačila pozornost.

Spomenimo još i alatnu traku u kojoj se sada na lijevoj strani bolje vidi značenje tzv. *breadcrumbsa*. Sada se bolje vidi da je trenutni "položaj prikaza" u komponenti *ARM* koja se nalazi unutar komponente *System*. Može se kliknuti na ikonu u *breadcrumbsima* kako bi se izravno prešlo na prikaz kliknute komponente.

Konačno, na slici se vidi i ovalni gumb s natpisom "*Use decimal base*" koji je "uključen" što se vidi po tamnoj pozadini. Ovim gumbom se može odabrati prikaz stanja registara u dekadskoj bazi (*defaultno* je heksadekadska baza koja je obično praktičnija). Ako radimo neke numeričke izračune, onda je možda pogodno povremeno uključiti dekadsku bazu kako bismo lakše provjerili rezultate u registrima. Baza se mijenja tek kod sljedećeg osvježavanja stanja u registrima, tj. kad se izvede prva sljedeća naredba. Registri stanja *CPSR* i *SPSR* uvijek prikazuju svoje stanje binarno - na njih se odabir baze ne odnosi. Inače se ispred heksadekadskog broja ispisuje sufiks *0x* kako bi jasno bilo naznačeno s kojom bazom se radi, ali to nije do simulatora nego do načina na koji je modeliran procesor *ARM*.

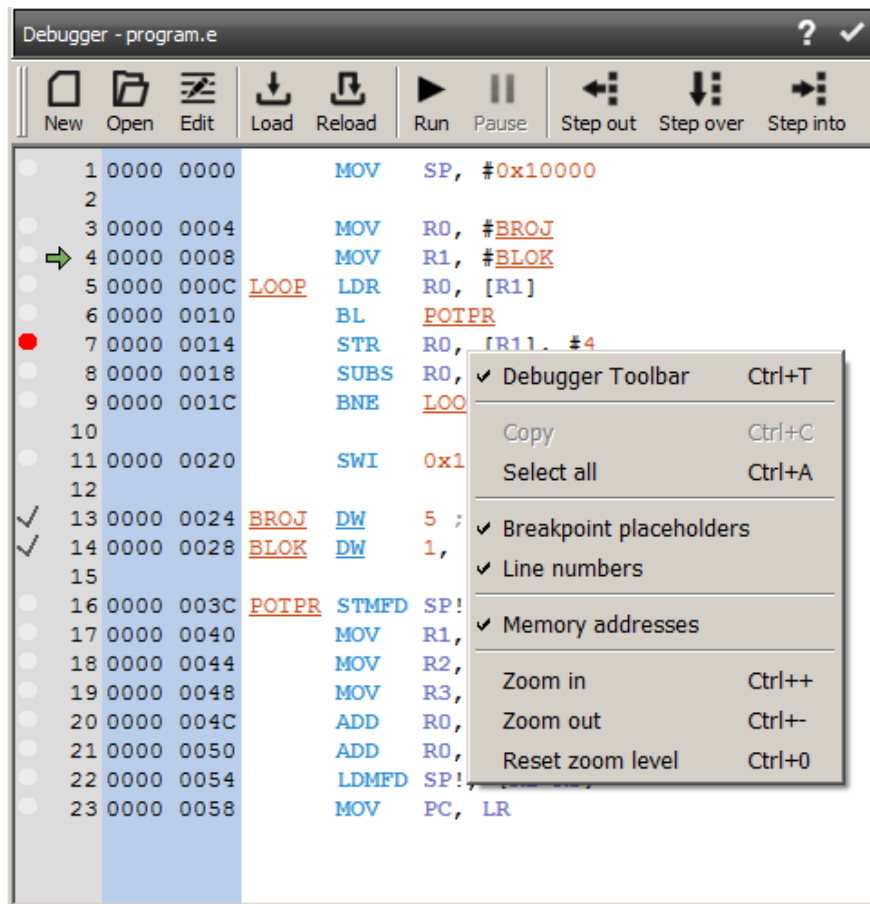
6.2 Polje Debug

Uz upravo opisano polje *Preview*, polje *Debug* najvažnije je polje simulatora. Već samo s ova dva polja dobiva se većina funkcionalnosti simulatora.

Polje *Debug* ima namjenu jednaku *debuggerima* u programskim razvojnim okolinama (engl. *IDE, Integrated Development Environment*). Polje *Debug* prikazuje **asemblerški program** koji je korisnik napunio u memoriju (točnije: korisnik je u memoriju napunio njegov prijevod u izvršni program). Polje *Debug* također omogućuje **upravljanje**

tijekom izvođenja asemblerskog programa. Izvođenjem se upravlja na uobičajeni način - pomoću **alatne trake**. Dok se program izvodi **korak-po-korak**, **strelica** stalno pokazuje na **trenutačnu naredbu** koja će upravo biti izvedena. Na pojedine asemblerske naredbe mogu se postavljati **prekidne točke**, a **podatci** unutar programa mogu se **odabrati za prikaz** u polju *MemView*. Brojne opcije čine ovo polje jednim od složenijih u simulatoru. Slijedi detaljan opis.

Polje *Debug* stvara se **automatski** prilikom učitavanja modela u simulator. Pri tome ono može biti sakriveno ili prikazano, ovisno o zapamćenim postavkama zadnjeg korištenja. Ovo polje ne može se obrisati iz simulatora, već se automatski briše tek ako se otvori novi model. Ako model u sebi ima više procesora, za svaki procesor se stvara njegovo polje *Debug*, ali to spada u naprednije korištenje i neće se dalje objašnjavati.



Na gornjoj slici se vide dijelovi polja *Debug* i kontekstni izbornik u stupcu s programom:

- Na vrhu je **naslovna traka** koja je vrlo jednostavna. U naslovu se nalazi ime izvršne datoteke (*program.e*) koja je napunjena u memoriju, a ikone (upitnik i kvačica) na desnoj strani omogućuju prikaz kratkog opisa polja *Debug* i sakrivanje polja *Debug*.
- Ispod naslovne trake je **alatna traka**, no njen položaj je promjenjiv i može biti na bilo kojem rubu polja. U njoj se nalazi niz gumbi/ikona (*New, Open, Edit, Load*, itd.)
- Ispod alatne trake nalazi se glavni dio polja. U njemu se prikazuje asemblerski program tijekom izvođenja i moguće je postavljanje prekidnih točaka i zadavanje praćenja memorijskih lokacija. Glavni dio polja je podijeljen u tri stupca, redom s lijeva na desno:
 - sivi **stupac s prekidnim točkama**
 - plavi **adresni stupac**
 - bijeli **stupac s programom**.

6.2.1 Stupac s prekidnim točkama

Stupac s prekidnim točkama (engl. *breakpoint*) zapravo ima i niz drugih funkcionalnosti osim što, naravno, radi s prekidnim točkama. Opis svih funkcionalnosti slijedi ispod.

- **Prikazuje sve prekidne točke** i njihova stanja. Prikaz izgleda kao kružić na lijevom rubu:
 - Svijetlo sivi kružić (tzv. *breakpoint placeholder*) označava retke na kojima se može postaviti prekidna točka, jer se prekidna točka ne može postaviti na prazne redove, na redove s asemblerskim direktivama, kao ni na redove s podatcima - na tim mjestima nema oznake za prekidne točke. *Breakpoint-placeholder* se može isključiti u kontekstnom izborniku Stupca s programom (stavka nije u kontekstnom izborniku stupca s prekidnim točkama kako ga se ne bi nepotrebno opterećivalo, jer on ima već dosta stavki i često se koristi)
 - Crveni kružić označava aktivnu ili omogućenu prekidnu točku (*enabled*)
 - Svjetlo crveni kružić označava deaktiviranu ili onemogućenu prekidnu točku (*disabled*)
 - **Uključivanje i isključivanje prekidnih točaka.** Može se zadati na dva načina:
 - Klikom miša na položaj kružića prekidna točka se naizmjenice uključuje i isključuje. Ovo je najpraktičniji način upravljanja.
 - Otvaranjem kontekstnog izbornika na položaju kružića može se odabrati stavka za uključivanje (*Set breakpoint*) ili isključivanje prekidne točke (*Remove breakpoint*).
 - **Isključivanje svih prekidnih točaka odjednom.** Može se zadati samo na jedan način:
 - Nakon otvaranja kontekstnog izbornika bilo gdje u stupcu, može se odabrati stavka za isključivanje svih prekidnih točaka (*Remove all breakpoints*).
 - **Aktiviranje i deaktiviranje prekidnih točaka.** Svaka prekidna točka može se privremeno deaktivirati, ako se želi da ona određeno vrijeme u budućnosti ne prekida izvođenje, a da ne "izgubimo" njen položaj. Može se zadati na dva načina:
 - Prekidna točka se naizmjenice aktivira i deaktivira kad se uz držanje tipke Ctrl klikne mišem na položaj kružića. Ovo je najpraktičniji način upravljanja.
 - Otvaranjem kontekstnog izbornika na položaju kružića može se odabrati stavka za aktiviranje (*Enable breakpoint*) ili deaktiviranje prekidne točke (*Disable breakpoint*).
- Ne postoji mogućnost deaktiviranja svih prekidnih točaka odjednom.

6.2.1.2 Prikaz lokacija u memView polju

- **Prikazuje uključenost ispisa stanja lokacija** u polju *MemView*. Prikaz izgleda kao kvačica na lijevom rubu:
 - Svijetlo siva kvačica (tzv. *placeholder*) označava retke na kojima se može uključiti ispis stanja lokacija u polju *MemView*. Stanje lokacija se ne može uključiti na praznim redovima, na redovima s naredbom, kao ni na nekim asemblerskim direktivama - na tim mjestima nema nikakve oznake. Uočite da se na redovima gdje se može postaviti prekidna točka sigurno ne može biti ispis lokacije i obratno, jer na jednim redovima su naredbe, a na drugima su podatci. Treća vrsta redova su oni na kojima se ne može uključiti ni jedno ni drugo - to su prazni redovi ili redovi s nekim specifičnim direktivama (npr. ORG ili EQU). *Placeholder* se može isključiti u kontekstnom izborniku Stupca s programom s istom stavkom koja se koristi za *Breakpoint-placeholder*.
 - Crna kvačica označava da se pripadna lokacija prikazuje u polju *MemView*.
 - **Uključivanje i isključivanje ispisa lokacija** u polju *MemView* izvodi se slično kao i uključivanje i isključivanje prekidnih točaka. Moguća su dva načina uključivanja i isključivanja:
 - Klikom miša na položaj kvačice se naizmjenice uključuje i isključuje prikaz lokacije. Ovo je najpraktičniji način upravljanja.
 - Otvaranjem kontekstnog izbornika na položaju kvačice može se odabrati stavka za uključivanje (*Watch location*) ili isključivanje prikaza lokacije (*Unwatch location*).
- Za lokacije ne postoji mogućnost aktiviranja i deaktiviranja prikaza.
- **Uključivanje i isključivanje prikaza svih lokacija odjednom.** Može se zadati pomoću kontekstnog izbornika (a može se zadati i u polju *MemView* u kojima se odabrane lokacije prikazuju):
 - Nakon otvaranja kontekstnog izbornika bilo gdje u stupcu, može se odabrati stavka za uključivanje (*Watch all*) ili stavka za isključivanje prikaza svih lokacija (*Unwatch all*).

6.2.1.3 Strelica za označavanje trenutne naredbe i redni broj retka

- U sredini stupca prikazuje se **strelica koja pokazuje na naredbu koja se upravo treba izvesti**, ali još nije izvedena. Ovo omogućuje praćenje izvođenja programa **korak-po-korak**. Strelica se pomiče kako se program izvodi i kako korisnik **pritiska gumb** *Step*. Strelica je **zelene** boje, ali mijenja boju u **crvenu** kada ARM izvodi naredbu s **lažnim uvjetom**, dakle naredbu koja efektivno neće biti izvedena.
- Prikaz **rednog broja retka** u asemblerskoj datoteci nalazi se na desnoj strani stupca. Može poslužiti kao orijentacija položaja nekog retka u datoteci, ali u praksi se ne koristi prečesto. Može se isključiti pomoću kontekstnog izbornika stupca s programom (stavka nije u kontekstnom izborniku stupca s prekidnim točkama kako ga se ne bi nepotrebno opterećivalo, jer on ima već dosta stavki i često se koristi).

6.2.1.4 Kontekstni izbornik stupca s prekidnim točkama

Popis stavaka u kontekstnom izborniku (**stavke ovise o retku** na kojemu je izbornik otvoren):

- Redci bez naredaba i podataka imaju samo **općenite stavke**:
 - *Remove all breakpoints*: isključi sve prekidne točke
 - *Watch all*: uključi sve lokacije za prikaz u polju *MemView*
 - *Unwatch all*: isključi sve lokacije iz prikaza u polju *MemView*
- Redak s **naredbom** ima **dodatne stavke vezane za prekidne točke**:
 - *Set breakpoint*: uključi prekidnu točku (kratica: Left click)
 - *Remove breakpoint*: isključi prekidnu točku (kratica: Left click)
 - *Disable breakpoint*: deaktiviraj prekidnu točku (kratica: Ctrl+Left click)
 - *Enable breakpoint*: aktiviraj prekidnu točku (kratica: Ctrl+Left click)
 - **Općenite** stavke za prazne retke - kao što je gore opisano
- Redak s **podatkom** ima **dodatne stavke za uključivanje i isključivanje prikaza lokacija**:
 - *Watch location*: uključi lokaciju za prikaz u polju *MemView* (kratica: Left click)
 - *Unwatch location*: isključi lokaciju iz prikaza u polju *MemView* (kratica: Left click)
 - **Općenite** stavke za prazne retke - kao što je gore opisano

Stavke u kontekstnim izbornicima postaju neosjetljive ili nevidljive ako nisu moguće ili nemaju smisla. Primjerice, ako u retku već postoji prekidna točka, onda stavka *Set breakpoint* nije vidljiva, nego je vidljiva samo stavka *Remove Breakpoint*.

6.2.2 Adresni stupac

Adresni stupac ima plavu pozadinu zbog lakšeg razlikovanja od susjednih stupaca. Vrlo je jednostavan i **prikazuje početnu adresu** svakog retka. Svaki **redak zauzima određeni broj adresa, tj. lokacija** - ovisno što se u retku nalazi. Ako se u retku nalazi jedna naredba onda ona zauzima 4 adrese, a ako se nalazi niz podataka (npr. definiranih direktivom DW ili DB), onda može zauzimati i manji ili veći broj adresa.

Adresni stupac najčešće je koristan samo kad iz nekog razloga želimo vidjeti točnu adresu pojedine naredbe ili podatka, a to u praksi nije prečesto. Zato je najbolje adresni stupac sakriti. Za sakrivanje i prikazivanje adresnog stupca koristi se stavka u kontekstnom izborniku stupca s programom. Kontekstni izbornik samog adresnog stupca također omogućuje sakrivanje stupca, ali dodatno i odabir brojeve baze kojima se prikazuju adrese u stupcu.

6.2.2.1 Kontekstni izbornik adresnog stupca

Kontekstni izbornik je vrlo jednostavan i uvijek izgleda jednako:

- **Memory address**: sakrivanje adresnog stupca
- Tri stavke koje čine tzv. *radio-button* grupu, tj. točno jedna stavka može biti odabrana:
 - **Binary**: zadaje binarnu bazu za prikaz adresa
 - **Decimal**: zadaje dekadsku bazu za prikaz adresa
 - **Hexadecimal**: zadaje heksadekadsku bazu za prikaz adresa

6.2.3 Stupac s programom

Stupac s programom zauzima najveću površinu polja **Debug**. U njemu se prikazuje asemblerski program koji se nalazi u memoriji i koji se izvodi na procesoru. Specifičnost prikaza je da se **pozivi makro zamjena prikazuju u proširenom obliku** tako da se točno mogu vidjeti naredbe koje se izvode, za razliku od uobičajenih prikaza gdje se vidi samo poziv i ne može se pratiti izvođenje naredaba u tijelu makronaredbe (npr. u *debuggerima* za C/C++).

Kako bi se što lakše pratilo izvođenje, program se prikazuje s bojenjem sintakse. Ono je radi uniformnosti jednako kao i u editoru od asemblerske aplikacije. Bojenje sintakse se može isključiti ili uključiti pomoću menija **Options**, ali ova promjena nije vidljiva odmah, nego tek nakon sljedećeg punjenja izvršne datoteke u memoriju.

Stupac s programom, osim prikaza programa, nema nekih posebnih funkcionalnosti osim:

- Selektiranje teksta mišem
- Povećavanje i smanjivanje prikaza kombinacijom tipke Ctrl i kotačića na mišu

6.2.3.1 Kontekstni izbornik stupca s programom

Kontekstni izbornik uvijek je jednak i nudi sve ostale akcije (većinom namještanje prikaza), od kojih su neke dostupne i preko kratica:

- **Debug Toolbar**: sakrivanje i prikazivanje alatne trake u polju **Debug** (kratica Ctrl+T)
- **Copy**: kopiranje selektiranog teksta (kratica Ctrl+C)
- **Select All**: selektiranje cijelog teksta (kratica Ctrl+A)
- **Breakpoint placeholders**: sakrivanje ili prikazivanje *placeholdera* za prekidne točke i za oznake ispisa lokacija u stupcu za prekidne točke
- **Line numbers**: sakrivanje ili prikazivanje rednih brojeva redaka u stupcu za prekidne točke
- **Memory Addresses**: sakrivanje ili prikazivanje adresnog stupca (ovo je jedini način za ponovni prikaz sakrivenog adresnog stupca)
- **Zoom in**: povećanje prikaza (kratica Ctrl+'plus')
- **Zoom out**: smanjivanje prikaza (kratica Ctrl+'minus')
- **Reset zoom level**: postavljanje veličine prikaza na *defaultnu* vrijednost (kratica Ctrl+0)

Da bi se pojednostavnili kontekstni izbornici, zadnje tri stavke vezane za *zoom* mogu se sakriti. Istovjetne stavke postoje i u poljima **Preview** te u polju **Console** pa sakrivanje vrijedi za sve tri vrste polja odjednom. Sakrivanje se ostvaruje pomoću opcija u izborniku **Options**.

6.2.4 Alatna traka

Ispod naslovne trake se *defaultno* nalazi **alatna traka sa svim raspoloživim akcijama za upravljanje izvođenjem asemblerskog programa**.

U alatnoj traci su još i **akcije za editiranje asemblerskog programa** te za **učitavanje izvršnog programa**. Ove se akcije mogu ostvariti i izravnim pozivanjem SSPARCSS asemblera, ali ih je lakše po potrebi pozivati iz simulatora. Razlog je u tome što i simulator i asembler mijenjaju izvršnu datoteku. Zato se ona zaključava dok se koristi u jednom programu kako bi se spriječila njena istovremena promjena u drugom programu, jer bi to dovelo do korumpiranog stanja datoteke. Prilikom korištenja alatne trake se sva zaključavanja i otključavanja ostvaruju koordinirano i automatski kako bi se datoteka mogla izmjenično koristiti u oba programa, a rad u simulatoru se blokira dok se ne završi rad u asembleru. Prilikom ručnog pozivanja asemblera, neće biti moguće istovremeno raditi na programu u asembleru ako se taj program drži otvorenim u simulatoru.

Alatna traka može se sakriti pomoću kontekstnog izbornika. Umjesto nje se pomoću izbornika *Options* može uključiti istovjetna alatna traka u glavnom prozoru. Alatna traka se može premještati po rubovima polja *Debug* tako da se mišem uhvati oznaka na njenom lijevom rubu (dvije paralelne crte) i da se traka odvuče na željeni položaj.

Dio akcija iz alatne trake može se pokrenuti pomoću kratica (engl. *shortcuts*). Akcije iz alatne trake nisu dostupne iz izbornika u glavnom prozoru jer one nisu općenite i ovise o modelu (o broju procesora u modelu, o mogućnostima koračnog izvođenja svakog pojedinog procesora itd.). To znači da u modelu nekog drugog procesora možda neće biti vidljive sve ikone koje su prikazane na gornjoj slici. Raspoložive akcije opisane su redom s lijeva na desno.

6.2.4.1 Gumbi za pozivanje asemblerskog prevoditelja

- **New: Stvaranje novog asemblerskog programa.** Akcija se izvodi na sljedeći način:
 - Prvo se automatski otvara prozor za zadavanje nove asemblerske datoteke (sa ekstenzijom **.a*)
 - Zatim se automatski pokreće asemblerski prevoditelj za zadanu datoteku (za to vrijeme u simulatoru ništa nije moguće raditi, sve dok se ne završi rad u asembleru)
 - Nakon uspješnog asembliranja, model u simulatoru se automatski inicijalizira, a dobiveni izvršni program se automatski puni u memoriju
- **Open: Asembliranje postojećeg asemblerskog programa.** Akcija se izvodi na sljedeći način:
 - Prvo se automatski otvara prozor za zadavanje postojeće asemblerske datoteke (sa ekstenzijom **.a*)
 - Zatim se automatski pokreće asemblerski prevoditelj za zadanu datoteku (za to vrijeme u simulatoru ništa nije moguće raditi, sve dok se ne završi rad u asembleru)
 - Nakon uspješnog asembliranja, model u simulatoru se automatski inicijalizira, a dobiveni izvršni program se automatski puni u memoriju
- **Edit: Asembliranje trenutnog asemblerskog programa.** Akcija je dostupna samo ako je u modelu već napunjen izvršni program. Akcija se može izvesti i pomoću *Open*, ali onda treba zadavati datoteku, a *Edit* je samo praktičniji način da se postigne isti učinak. Akcija se izvodi na sljedeći način:
 - Automatski se pokreće asemblerski prevoditelj za trenutnu asemblersku datoteku (za to vrijeme u simulatoru ništa nije moguće raditi, sve dok se ne završi rad u asembleru)
 - Nakon uspješnog asembliranja, model u simulatoru se automatski inicijalizira, a dobiveni izvršni program se automatski puni u memoriju

6.2.4.2 Gumbi za učitavanje izvršnog programa u memoriju

- **Load: Učitavanje nove izvršne datoteke.** Na disku već mora postojati izvršna datoteka dobivena prethodnim asembliranjem. Uobičajeno se izvodi nakon učitavanja modela, kad se prvi puta puni izvršni program u memoriju. Akcija se izvodi na sljedeći način:
 - Prvo se automatski otvara prozor za zadavanje nove izvršne datoteke (sa ekstenzijom **.e*)
 - Model u simulatoru se automatski inicijalizira, a izvršni program se automatski puni u memoriju
- **Reload: Ponovno učitavanje trenutne izvršne datoteke.** Akcija je dostupna samo ako je prethodno već bila učitana neka izvršna datoteka. Akcija se može izvesti i pomoću *Load*, ali onda treba zadavati datoteku, a *Reload*

je samo praktičniji način da se postigne isti učinak. Uobičajeno se izvodi na početku svake runde simuliranja. Akcija se izvodi na sljedeći način:

- Model u simulatoru se automatski inicijalizira, a trenutni izvršni program se automatski ponovno puni u memoriju

6.2.4.3 Gumbi za pokretanje i pauziranje simulacije

- **Run: Pokretanje ili nastavljavanje izvođenja programa** bez označavanja pojedine naredbe strelicom. Izvođenje teče sve do pojave jednog od četiriju događaja:
 - Ako se tijekom izvođenja naiđe na **aktivnu prekidnu točku**, izvođenje će biti **pauzirano**. Nakon pauziranog izvođenja moguće je nastaviti izvođenje gumbom **Run**.
 - Ako se tijekom izvođenja **pritisne ikona Pause**, izvođenje će biti **pauzirano**. Nakon pauziranog izvođenja moguće je nastaviti izvođenje gumbom **Run**.
 - Ako se tijekom izvođenja izvede **naredba za zaustavljanje procesora** (u **ARM**-u je to naredba **SWI**), simulacija se **završava** i ne može se više nastaviti, nego se samo može pokrenuti nova runda simuliranja.
 - Ako tijekom izvođenja dođe do **greške**, otvorit će se **popup**-prozor s porukom i simulacija će se **završiti** bez mogućnosti nastavljanja. Opet se može pokrenuti nova runda simuliranja.
- **Pause: Pauziranje simulacije** tijekom njenog izvođenja, tj. pauziranje izvođenja asemblerskog programa. Akcija je dostupna samo dok se simulacija izvodi, a inače je neosjetljiva. Kao što je iznad opisano, pauziranje omogućava kasnije nastavljavanje simulacije pomoću **Run**, ali nastavljavanje je moguće i pomoću bilo kojeg od tri vrste **Stepa**.

6.2.4.4 Gumbi Step za koračno izvođenje

Gumbi Step omogućavaju izvođenje korak-po-korak ili kraće koračno izvođenje. To je izvođenje uobičajeno u *debuggerima*. U koračnom izvođenju se izvodi samo jedna naredba i uz nju se prikazuje strelica s lijeve strane. Strelica pokazuje na naredbu koja se upravo treba izvesti, ali još nije izvedena.

Ponašanje izvođenja opisano za **Run** vrijedi i za koračno izvođenje i četiri moguća događaja će također zaustaviti ili prekinuti simulaciju. Dodatno, dovršetak jedne naredbe i dohvat druge, će biti peti događaj koji također pauzira izvođenje. Efektivno, jedina vidljiva razlika je u ponašanju gumba **Pauze**, jer kod koračnog izvođenja je izvođenje jedne naredbe toliko brzo da korisnik ne stigne pritisnuti gumb **Pauze**. Svaka pauzirana simulacija se može nastaviti ili pomoću **Run** ili pomoću nekog od **Stepova**. Prekidne točke ponašaju se kao što je uobičajeno, iako njihov efekt nije vidljiv ako koračno izvodimo onu naredbu na kojoj je prekidna točka, jer i prekidna točka i koračno izvođenje imaju isti učinak - pauziraju izvođenje na naredbi.

Podržane su sve tri vrste koračnog izvođenja uobičajenog u *debuggerima*.

- **Step out: ovo je izlaženje iz trenutne procedure** bez koračnog izvođenja i zaustavljanje na prvoj naredbi nakon povratka iz procedure. Ovisno o broju naredbi između trenutne naredbe i naredbe za izlazak iz procedure, ova akcija može imati određeno trajanje. Koristi se kada više ne želimo promatrati ponašanje trenutne procedure, nego želimo prijeći na niz naredaba odakle je procedura pozvana.
- **Step over: ovo je izvođenje trenutne naredbe i zaustavljanje na sljedećoj**. Jedini izuzetak su naredbe za ulazak u proceduru koje se ne izvode koračno, nego normalno, kao i sve naredbe u pozvanoj proceduri, a izvođenje se zaustavlja na naredbi iza naredbe poziva. Efektivno se poziv procedure i cijela procedura tretiraju kao jedna naredba. Koristi se kada želimo promatrati ponašanje samo niza naredaba u kojemu se trenutno nalazimo, ali ne želimo promatrati naredbe u procedurama koje se odatle pozivaju. Ovisno o broju naredbi u proceduri, ova akcija može imati određeno trajanje.
- **Step in: ovo je izvođenje naredbe i zaustavljanje na sljedećoj, ali bez obzira o kojoj naredbi se radi**. Ako je trenutna naredba poziv procedure, **Step in** će ući u proceduru i zaustavit će se na prvoj naredbi procedure. Koristi se kada želimo pratiti izvođenje baš svih naredbi, bez obzira jesu li u procedurama ili nisu.

6.2.4.5 Kratice za upravljanje izvođenjem

Gumbi *Run*, *Pause*, *Step out*, *Step over* i *Step in* imaju kratice koje omogućuju brzo i jednostavno upravljanje radom programa bez korištenja miša. Kratice su:

- *Run* i *Pause*: tipka **Return**
- *Step out*: tipka **Left** (strelica lijevo)
- *Step over*: tipka **Down** (strelica dolje)
- *Step in*: tipka **Right** (strelica desno)

Uočite da se ista kratica **Return** može koristiti i za *Run* i za *Pause* jer ova dva gumba **nikada** nisu aktivni u isto vrijeme. Na ovaj način možemo uzastopnim pritiscima iste tipke može "puštati i zaustavljati" simulaciju.

Također uočite da se raspored *Step* gumbi podudara s rasporedom strelica na tipkovnici i da strelice sugeriraju koji *Step* će se izvesti: strelica prema dolje sugerira da u nizu asemblerskih naredba idemo prema sljedećoj naredbi. Ako smo na naredbi poziva potprograma, to znači da idemo na naredbu ispod nje i da preskačemo promatranje potprograma (*Step over*). Analogno tome, ako smo na naredbi poziva potprograma, i ako pritisnemo tipku desno, to znači da želimo ući jednu razinu dublje u potprogramima, tj. da želimo promatrati izvođenje potprograma (*Step in*). Ako već jesmo u nekom potprogramu, onda lijeva strelica sugerira da se želimo vratiti jednu razinu potprograma iznad, tj. u pozivatelja potprograma bez da dalje pratimo tekući potprogram (*Step out*).

Zbog cjelovitosti opisa, navedimo još da u slučaju modela s više procesora, alatna traka izgleda malo drugačije - među *Step* gumbima postoji dodatni gumb *Normal run* koji zajedno s već opisana tri gumba *Step* tvori tzv. *radio-buttons* grupu, tj. grupu gumbi od kojih točno jedan može biti uključen. Na ovaj način se za svaki procesor zasebno može zadati na koji način se promatra izvođenje njegovih naredaba. Tada gumb *Run* služi samo za pokretanje simulacije, a navedena četiri gumba *Step* odabiru vrstu izvođenja za svaki procesor zasebno. Gumb *Normal run* ima kraticu **Up** (strelica gore).

6.3 Polje MemView

Polje *MemView*, kako mu i ime sugerira, služi za "prikazivanje memorije". Pod time se misli na **prikaz stanja u memorijskim lokacijama** koje su u asemblerskom programu definirane kao **podatci**, dakle definirani su pomoću jedne od asemblerskih direktiva (tj. pseudonaredaba) koje za to služe. Važno je napomenuti da *MemView* ne prikazuje zasebne memorijske lokacije (tj. adresabilne lokacije, a to su bajtovi), nego **prikazuje podatak u nizu lokacija i to u onoj širini kako je definiran direktivom**. Pri tome se uzima u obzir i **redoslijed bajtova** ili engl. *endianness* (little endian ili big endian) koji se koristi u modelu. Budući da su podatci definirani direktivama slični globalnim varijablama i poljima (engl. *array*) u višim programskim jezicima, onda možemo reći da *MemView* prikazuje varijable asemblerskog programa.

Polje *MemView* **stvara se i uništava automatski** - zajedno s poljem *Debug*. Za svaki procesor u modelu postoji jedan par polja *Debug* i *MemView*. Kao i polje *Debug*, korisnik ne može ukloniti polje *MemView*, nego ga može samo sakriti.

Prije objašnjenja načina prikazivanja u polju *MemView* moramo nešto reći o tome kako se definiraju podatci u asemblerskom programu. Za to će nam opet poslužiti primjer modela za ARM. U Asemblerskom jeziku za ARM postoji više direktiva za definiranje podataka, a svaka od njih definira određenu vrstu i širinu podatka:

- **DW** - Define word (4 bajta)
- **DH** - Define halfword (2 bajta)
- **DB** - Define Byte (1 bajt)
- **DD** - Define Doubleword (8 bajtova)
- **DS** - Define Space (niz od N bajtova)
- **DSTR** - Define String (niz od N bajtova)

Direktive DW, DH, DB i DD zadaju cjelobrojne podatke navedenih širina u bitovima, pa se dotični podatci tako i prikazuju u polju [MemView](#). Sve ove direktive imaju mogućnost zadavanja liste podataka i tada se ti **podatci prikazuju kao polja podataka** odgovarajuće širine.

Direktiva DSTR jedina zadaje drugu vrstu podataka - niz ASCII znakova ili *string*. On se prikazuje kao **polje znakova**, ali se uz svaki znak prikazuje i numerička vrijednost ASCII-koda.

Direktiva DS, definira **niz bajtova** određene duljine, pri čemu vrijednosti nisu inicijalizirane. Bajtovi nemaju nikakvo dodijeljeno značenje, pa se prikazuju i kao ASCII znakovi i kao cijeli brojevi.

Ispred direktive obično se nalazi **labela** pomoću koje se adresiraju definirani podatci, ali labela ne mora postojati.

Slijedi primjer odsječka programa u kojemu se definiraju podatci i njihov prikaz u polju [MemView](#). U odsječku se vidi da su uključene kvačice uz sve podatke pa se svi podatci prikazuju u polju [MemView](#), kao što je već objašnjeno u 6.2.1.2. U odsječku programa vidimo redom:

✓	0000	0050	REZULT	DW	0
✓	0000	0054	H1	DH	0x1234, 0x5678, 0x90AB
✓	0000	005A	B1	DB	33, 44, 55, 66
✓	0000	005E		DB	77
✓	0000	005F	PORUKA	DSTR	"OK\n"
✓	0000	0063	BLOK	DS	8

Name	Address	Value	Size	Type
REZULT	0x00000050	0x00000000	4	Integer
[-] H1	0x00000054	<3 items>	6	Short
[0]	0x00000054	0x1234	2	Short
[1]	0x00000056	0x5678	2	Short
[2]	0x00000058	0x90AB	2	Short
[-] B1	0x0000005A	<4 items>	4	Byte
[0]	0x0000005A	33	1	Byte
[1]	0x0000005B	44	1	Byte
[2]	0x0000005C	55	1	Byte
[3]	0x0000005D	66	1	Byte
BYTE@25:0	0x0000005E	0b01001101	1	Byte
[-] PORUKA	0x0000005F	<4 items>	4	String
[0]	0x0000005F	'O' (0x4F)	1	Char
[1]	0x00000060	'K' (0x4B)	1	Char
[2]	0x00000061	. (0x0A)	1	Char
[3]	0x00000062	. (0x00)	1	Char
[-] BLOK	0x00000063	<8 items>	8	String
[0]	0x00000063	. (0x00)	1	Char
[1]	0x00000064	. (0x00)	1	Char
[2]	0x00000065	. (0x00)	1	Char
[3]	0x00000066	. (0x00)	1	Char
[4]	0x00000067	. (0x00)	1	Char
[5]	0x00000068	. (0x00)	1	Char
[6]	0x00000069	. (0x00)	1	Char
[7]	0x0000006A	. (0x00)	1	Char

- labela [REZULT](#) s direktivom [DW](#) i jednim podatkom od 32 bita
- labela [H1](#) s direktivom [DH](#) i tri podatka od 16 bita
- labela [B1](#) s direktivom [DB](#) i četiri podatka od 8 bita
- direktiva [DB](#) bez labela i s jednim podatkom od 8 bita
- labela [PORUKA](#) s direktivom [DSTR](#) koja definira *string* "OK\n" u koji se automatski dodaje i \0 na kraj
- labela [BLOK](#) s direktivom [DS](#) koja definira blok podataka od 8 bajtova (inicijalno su svi u nuli)

6.3.1 Izgled polja MemView

[MemView](#) polje sastoji se od tri dijela. Prvi dio je uobičajeno naslovno polje u kojemu se vidi ime asemblerskog programa, a odmah ispod njega je **zaglavlje** s imenima stupaca koji se prikazuju: [Name](#), [Address](#), [Value](#), [Size](#) i [Type](#). Ispod toga se prikazuju svi podatci odabrani u polju [Debug](#). To su redom:

- Podatak s imenom [REZULT](#) na adresi [0x50](#), veličine 4-bajta i s trenutnom vrijednošću [0x0](#). Uz to se ispisuje veličina u bajtovima i vrsta podatka što oboje ovisi o direktivi. Budući da direktivom [DW](#) definiran jedan podatak od 4 bajta, onda on ukupno zauzima 4 lokacije (bajta), a vrsta svakog podatka je [Integer](#) (nazivi tipova podataka slični su kao u jezicima C/C++).
- Polje podataka s imenom [H1](#) s početnom adresom [0x54](#) i sa tri 2-bajtna podatka u polju. Ukupna veličina sada je 6 bajtova (3 podatka od 2 bajta svaki), a tip podataka u polju je [Short](#). Koristi se stablasti prikaz u kojemu su u granama prikazani pojedini podatci s pripadnim indeksima u polju, adresama, vrijednostima, veličinom i tipom. Naravno stablo se može zatvoriti da zauzima manje mjesta, ako trenutno ne želimo gledati pojedine podatke.
- Polje podataka s imenom [B1](#) s početnom adresom [0x5A](#) i sa četiri 1-bajtna podatka u polju. Ukupna veličina sada je 4 bajta, a tip podataka u polju je [Byte](#). Ovdje vidimo da su vrijednosti podataka prikazane u dekadskoj bazi, što je odabrao korisnik pomoću kontekstnog izbornika (po defaultu se sve prikazuje u heksadekadskoj bazi).

- Slijedi jedan bajt podatka bez zadanog imena u asemblerskoj datoteci. Njegov prikaz imena sastoji se od *imena tipa, znaka @ i broja retka u asemblerskoj datoteci (25)* kako bi ga se moglo identificirati. Ovo je opet pojedinačni podatak kao i *REZULT* pa se ne prikazuje stablasto.
- *String* s imenom *PORUKA* duljine 4 znaka ima opet stablasti prikaz, ali sada se kao vrijednost prikazuju znakovi (znak *točke* se prikazuje za neispisive znakove *\n* i *\0* koji se nalaze na kraju *stringa*), ali i ASCII-kodovi tih znakova s desne strane.
- Zadnji je prikazan blok dobiven direktivom *DS* sa zadanom duljinom od 8 lokacija (bajtova). Budući da za njega nije poznato hoće li sadržati ASCII-znakove ili brojeve, on se interpretira slično *stringu* kako bi pokazivao vrijednosti na oba načina.

Ovime su prikazane sve moguće varijante prikaza koji se može dobiti u polju *MemView*. Kod promjena vrijednosti, **zadnja će promjena** biti označena **crvenom** bojom.

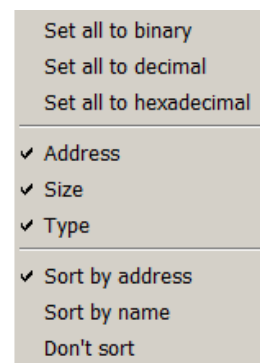
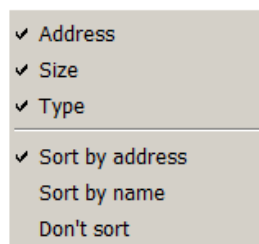
6.3.2 Akcije u polju MemView

MemView je polje namijenjeno prvenstveno prikazu pa su u njemu moguće samo akcije kojima se namješta prikaz prema željama korisnika. Sve akcije dostupne su iz **kontekstnih izbornika i pomoću zaglavlja**.

Redoslijed stupaca može se namjestiti na uobičajen način. Bilo koji naziv stupca u zaglavlju može se uhvatiti mišem i odvući na željenu poziciju (*drag-and-drop*). Izuzetak je jedino prvi stupac *Name* kojemu se položaj ne može mijenjati. Svim stupcima može se **mijenjati širina** tako da se mišem pomakne granica između stupaca.

U zaglavlju je moguće otvoriti dva različita kontekstna izbornika prikazana slikama. Klikom na bilo koji dio zaglavlja, osim stupca *Value*, dobiva se izbornik na lijevoj strani, a u stupcu *Value* dobiva se izbornik na desnoj strani. Zajedničke stavke su:

- *Address*: prikaz ili sakrivanje stupca *Address*
- *Size*: prikaz ili sakrivanje stupca *Size*
- *Type*: prikaz ili sakrivanje stupca *Type*



Ove stavke omogućuju **sakrivanje navedena tri stupca**, što je preporučljivo da bi se uštedio prostor. Stupci *Name* i *Value* ne mogu se sakrivati.

Slijedeće zajedničke stavke služe za zadavanje **redoslijeda redaka**. One tvore tzv. *radio-button* grupu, tj. točno jedna stavka mora biti odabrana (po *defaultu* je to stavka *Sort by address*):

- *Sort by address*: dobiva se jednaki redoslijed podataka kao što je to u asemblerskom programu, jer u njemu su svi redovi zapravo uvijek poredani po rastućim adresama
- *Sort by name*: omogućuje sortiranje po imenima lokacija, ako korisnik smatra da će tako lakše pronaći pojedine lokacije
- *Don't sort*: redovi se ne sortiraju, nego će njihov redoslijed ovisiti o redoslijedu uključivanja njihovog prikaza u polju *Debug*. Pri uključivanju pojedinog retka on se uvijek dodaje na kraj.

Kontekstni izbornik zaglavlja za stupac *Value* ima dodatne tri stavke koje se odnose na taj stupac. Njima se može **odabrati brojeva baza za prikaz svih vrijednosti** u polju *MemView*. Odabirom bilo koje stavke, odmah se mijenja baza za sve vrijednosti. Na raspolaganju su standardne baze sa stavkama:

- *Set all to binary*: prikaži sve brojeve binarno
- *Set all to decimal*: prikaži sve brojeve dekadski
- *Set all to hexadecimal*: prikaži sve brojeve heksadekadski.

Za adresni stupac ova mogućnost ne postoji, a adrese se uvijek prikazuju heksadekadski. Brojeva baza može se odabrati i za svako polje podataka ili za svaki pojedinačni podatak u polju ili izvan polja. Za to se naravno koriste **kontekstni izbornici** koji se otvaraju na **prvom retku polja** ili na **pojedinih podatcima**. Ovi izbornici imaju slične stavke

kao upravo pokazane stavke za **odabir brojevn**e baze, a dodatno imaju i stavke kojima se može **isključiti prikaz dotičnog podatka**. Pri tome se za polja ne mogu isključivati pojedini podatci, nego samo cijelo polje. Zbog sličnosti s prikazanim izbornikom i zbog jednostavnosti ovi izbornici nisu prikazani slikama.

Veličina prikaza u polju *MemView* može se namještati reguliranjem veličine fonta za što se koristi izbornik *Options*. Tako podešena veličina fonta vrijedi i za polja *StackView*, *Navigator*, *Display* i *Dump*.

6.4 Polje StackView

Polje *StackView* prikazuje stog u memoriji računala. Njegov prikaz pomoću *MemView* ne bi bio praktičan jer je stog specifičan: ima promjenjivu veličinu. U slučaju ARM-a postoji više stogova (po jedan za svaki način rada procesora) pa zato simulator pruža mogućnost rada s više polja *StackView*.

Polja *StackView* stvaraju se i uništavaju automatski - zajedno s poljem *Debug* i poljem *MemView*. Za svaki procesor u modelu postoji točno određeni broj polja *StackView* (četiri za ARM). Kao i polja *Debug* i *MemView*, korisnik ne može uklanjati polja *StackView*, nego ih može samo sakriti.

Sljedeća slika prikazuje izgled dvaju polja *StackView* za dva ARM-ova stoga (*svc* i *irq*). Na lijevo je prikazan slučaj kad su oba stoga prazna, a desno kad na oba stoga ima podataka. Kao podatci na stogu prikazuju se **skupine od 4 bajta**, jer se na stogu uvijek nalaze samo podatci takve veličine. Za svaki podatak se prikazuje **adresa** i **vrijednost** u odgovarajućem stupcu. Osim toga je **sivom strelicom** s desne strane označena **početna adresa stoga (dno)**, a **zelenom strelicom** s lijeve strane **trenutačni vrh stoga**. Siva strelica nikada ne mijenja položaj, za razliku od zelene koja se pomiče kod svakog stavljanja ili skidanja podataka sa stoga.

Stack View - svc_stack		Stack View - irq_stack	
Address	Value	Address	Value
0x 00 01 00 00		0x 00 01 00 00	
		0x 00 00 FF F0	0x 00 00 00 D3
		0x 00 00 FF F4	0x 00 00 00 7B
		0x 00 00 FF F8	0x 00 00 00 00
		0x 00 00 FF FC	0x 00 00 01 C8
		0x 00 00 00 00	

Boje u prikazu imaju sljedeće značenje. Na **plavoj** podlozi prikazani su **podatci koji se nalaze na stogu**. **Siva** podloga označava **dno stoga** koje ne sadrži podatak koji bi pripadao stogu, jer se ovdje radi o *Full-Descending* stogu, tj. o stogu kojemu je **vrh pun (Full)**, a **dno je prazno**.

Siva podloga je **prazna** za *svc-stog*, a za *irq-stog* **sadrži podatak**. Razlog leži u tome što je *svc-stog* inicijaliziran na najvišim memorijskim adresama i početna vrijednost SP-a kod njega pokazuje na *0x10000*, što je izvan modela *ARM*-ove memorije. Zato se **ne prikazuje podatak**, jer se radi o **adresi koja ne postoji u fizičkoj memoriji**.

Za *irq-stog*, početna adresa je *0xD000* što je **postojeća adresa u fizičkoj memoriji** *ARM*-ovog modela pa automatski na toj adresi sigurno **postoji i neki podatak**, koji se zato **prikazuje** na sivoj podlozi. Stogovi se uvijek stavljaju na najviše memorijske adrese, ali samo jedan od njih može biti na krajnjoj adresi, kao što je ovdje *svc-stog*.

U ARM-u stogovi nisu sklopovski predefinirani (kao u većini drugih procesora) nego su potpuno pod upravljanjem asemblerskog programa. Zato pokažimo još i kako se prikazuje *Empty-Descending* stog, iako se preporuča upotreba *Full-Descending* stogova.

Stack View - svc_stack			Stack View - irq_stack		
Address	Value		Address	Value	
⇒ 0x 00 00 FF FC 0x 00 00 00 00 ←			⇒ 0x 00 00 D0 00 0x 00 00 00 00 ←		

Stack View - svc_stack			Stack View - irq_stack		
Address	Value		Address	Value	
0x 00 00 FF EC	0x 00 00 00 00	←	0x 00 00 CF E8	0x 00 00 00 00	←
0x 00 00 FF F0	0x 00 00 00 D3		0x 00 00 CF EC	0x 00 00 00 53	
0x 00 00 FF F4	0x 00 00 00 7B		0x 00 00 CF F0	0x FF FF 0E 00	
0x 00 00 FF F8	0x 00 00 00 00		0x 00 00 CF F4	0x 00 00 00 00	
0x 00 00 FF FC	0x 00 00 01 C8		0x 00 00 CF F8	0x 00 00 01 C8	
⇒ 0x 00 00 FF FC	0x 00 00 01 C8		0x 00 00 CF FC	0x 00 00 00 00	
			⇒ 0x 00 00 D0 00	0x 00 00 00 00	

Položaj sivih polja sada je, naravno, obrnut. To je zato jer je kod *Empty-Descending* stogova **vrh prazan (Empty)**, a **dno stoga je puno**. Vidimo da je sada trenutačni vrh stoga (označen zelenom strelicom) uvijek siv, a dno (označeno sivom strelicom) je uvijek plavo. Naravno, na početku dok je stog prazan, dno je na istom položaju kao i vrh pa je tada ta lokacija siva za bilo koju vrstu stoga - jer na stogu još nema podataka. Napomenimo da se sada, jer ovdje se radi o *Empty* stogu, *svc-stog* ne smije inicijalizirati na adresu *0x10000*. Razlog je taj što bi prvo stavljanje podatka na stog prouzročilo grešku upisa na nepostojeću memorijsku lokaciju s adresom *0x10000*. Zato je adresa dna za 4 manja, tj. iznosi *0xFFFFC*, što je adresa zadnje 32-bitne riječi u memoriji.

Osim *Descending* stogova, kod ARM-a su mogući i *Ascending* stogovi, ali oni su vrlo neuobičajeni pa ovdje nema njihovih slika (koje su analogne prikazanima, samo što stog raste u suprotnom smjeru - od nižih prema višim adresama).

Kao što se vidi na gornjim slikama, izgled prozora je vrlo jednostavan. Osim uobičajenog naslovnog polja (koje sadrži ime stoga), postoji još samo zaglavlje s nazivima stupaca (*Address* i *Value*) i prikaz podataka na stogu.

Na samom početku svake runde simuliranja, **prije nego se asemblerski program počne izvoditi**, još uvijek nije poznat budući položaj stoga - ni njegovo dno ni njegov vrh. U tom razdoblju, prikaz u poljima *StackView* je **posve prazan**. Tek nakon prvih upisa u registre SP, počinje se prikazivati prazni stog sa dnom i vrhom kao na gornjim slikama na lijevoj strani. *StackView* ne obilježava crvenom bojom punjenje i pražnjenje stoga jer je to dovoljno uočljivo. Promjene podataka unutar stoga se također ne obilježavaju crvenom bojom, ali to nije velik nedostatak jer će ove promjene biti relativno rijetke.

Još par napomena u vezi namještanja prikaza koje je različito od polja *MemView* iako su vizualno relativno slični. Ukratko, u *StackView* nema skoro nikakvih mogućnosti korisničkom namještanja izgleda polja. I za prikaz adresa i za prikaz podataka uvijek se koristi **heksadekadaska** baza - ona se ne može mijenjati. Također nema mogućnosti sakrivanja ili promjene redoslijeda stupaca ili promjene širine stupaca. Ni inače *StackView* ne pruža nikakve akcije korisniku.

Može se mijenjati jedino veličina prikaza u polju *StackView*, ali ne izravno nego reguliranjem veličine fonta za što se koristi izbornik *Options*. Tako podešena veličina fonta vrijedi i za polja *MemView*, *Navigator*, *Display* i *Dump*.

7 Ostala polja u sučelju

Ostala polja su od manje važnosti i krajnji korisnici će ih vjerojatno rjeđe koristiti tijekom rada sa simulatorom. Korisnicima koji sami izrađuju svoje modele će ova polja biti od veće koristi.

7.1 Polje Console

Polje **Console** je jedno od glavnih polja koje je uvijek prisutno u sučelju i može se samo sakriti. Polje postoji u točno **jednom primjerku** i predstavlja **centralno mjesto za razne ispise** tijekom rada sa simulatorom. **Console** služi za:

1. tekstualni ispis poruka
2. praćenje stanja simulacije.

Console se obično može sakriti da ne zauzima prostor na zaslonu, a preporuča se otvoriti tek ako dođe do greške da se provjeri je li možda greška detaljnije opisana u polju **Console**.

7.1.1 Poruke u polju Console

Poruke koje se ispisuju uglavnom su vezane za različite **greške** ili se radi o **kratkim porukama da je uspješno izvedena neka akcija koju je zadao korisnik**. Primjerice:

- poruke o očitavanju modela
- poruke o punjenju izvršne datoteke
- poruke o greškama tijekom simulacije, itd.

Budući da je pretpostavka da korisnici koriste gotove modele, vjerojatnost da će doći do ovakvih grešaka je minimalna. Jedino može doći do određenih grešaka u simulaciji u sljedećim slučajevima:

- **pristup nepostojećim adresama** pomoću naredaba LDR/STR i sličnih
- **pristup riječima ili poluriječima na neporavnatim adresama**
- **skok na nepostojeću naredbu** pomoću naredbe B/BL ili nailazak na nepostojeću naredbu ako nedostaje naredba skoka/zaustavljanja simulacije
- **čitanje sa sabirnica kojima nitko ne upravlja** prilikom pristupanja vanjskim jedinicama
- **pisanje na zauzetoj sabirnici** prilikom pristupanja vanjskim jedinicama, itd.

Prilikom grešaka otvara se *popup*-prozor s kratkim opisom greške, a detaljniji opis greške se često ispisuje u **Console**. **U 99% slučajeva, do ovih grešaka dolazi zbog neispravno napisanih asemblerskih programa.**

7.1.2 Praćenje stanja simulacije u polju Console

Praćenje stanja simulacije je isključivo **ovisno o modelu** i postavkama koje zada korisnik. Modeli su uglavnom koncipirani tako da minimiziraju ili da uopće ne koriste **Console** za praćenje stanja jer su za tu svrhu pogodnije animacije u poljima **Preview** i prikazi u poljima **MemView** i **StackView**.

U konkretnom slučaju modela računala s ARM-om, postoje svega tri postavke koje korisnik može uključiti u cilju praćenja simulacije u polju **Console**. Jedna je u samom procesoru i već je prikazana u 6.1.2.1 gdje se pomoću kontekstnog izbornika komponente u polju **Preview** mogu uključivati i isključivati postavke. Podsjetimo, to je moguće

za komponente koje u uglu imaju nacrtan malo plavi pravokutnik s kvačicom. Ova postavka zadaje da tijekom izvođenja asemblerskog programa treba ispisivati ili *logirati* izvedene naredbe u polje *Console*. Slične opcije postoje i za komponentu *DMAC* kod kojih se u *Console* ispisuju pojedini podatci koji se prenose pomoću *DMAC*-a. Naravno da je u slučaju uključenja ovih postavki logično da će korisnik prikazati polje *Console*.

7.1.3 Akcije u polju Console

U polju *Console* postoji svega nekoliko akcija i jednostavni kontekstni izbornik. Kontekstni izbornik od *Console* ima stavke koje smo već vidjeli kod kontekstnog izbornika polja *Debug*:

Copy	Ctrl+C
Select all	Ctrl+A
Zoom in	Ctrl++
Zoom out	Ctrl+-
Reset zoom level	Ctrl+0

- *Copy*: kopiranje selektiranog teksta (kratica Ctrl+C)
- *Select All*: selektiranje cijelog teksta (kratica Ctrl+A)
- *Zoom in*: povećanje prikaza (kratica Ctrl+'plus')
- *Zoom out*: smanjivanje prikaza (kratica Ctrl+'minus')
- *Reset zoom level*: postavljanje veličine prikaza na *defaultnu* vrijednost (kratica Ctrl+0)

Da bi se pojednostavnili kontekstni izbornici, zadnje tri stavke vezane za *zoom* mogu se sakriti. Istovjetne stavke postoje i u poljima *Preview* i *Debug* pa sakrivanje vrijedi za sve tri vrste polja odjednom. Sakrivanje se ostvaruje pomoću opcija u izborniku *Options*.

Izravne funkcionalnosti polja *Console* su ponovno slične polju *Debug*:

- Selektiranje teksta mišem
- Povećavanje i smanjivanje prikaza kombinacijom tipke Ctrl i kotačića na mišu

7.2 Polje Navigator

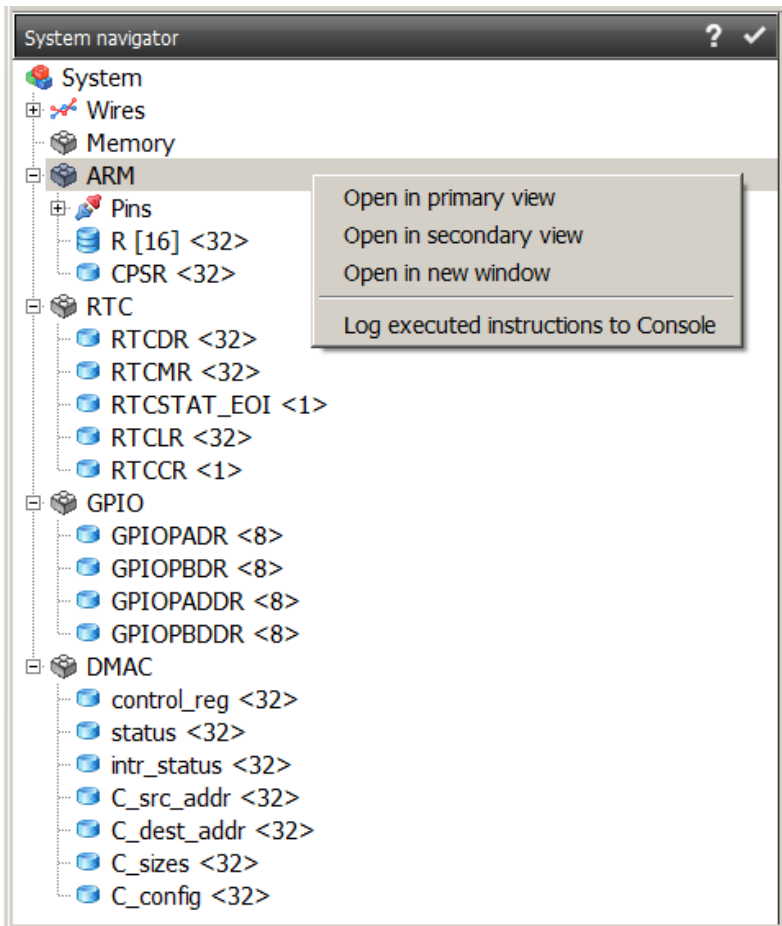
Glavna namjena polja *Navigator* je **prikazivanje svih dijelova modela u hijerarhijskom**, odnosno stablastom **obliku**. Osim toga na prikazanim dijelovima moguće je otvaranje kontekstnih izbornika.

Navigator može biti praktičan za složenije modele koji imaju više hijerarhijskih razina - komponente koje u sebi imaju podkomponente koje opet u sebi imaju druge podkomponente itd.

U slučaju relativno jednostavnog modela kao što je ARM postoji samo sistemska komponenta koja u sebi ima druge komponente, koje pak nemaju daljnjih podkomponentata. Zato je korisnost *Navigator*a umanjena, tim više što su iste podkomponente dostupne u polju *Preview*.

Navigator prikazuje sve komponente i dijelove kao čvorove hijerarhijskog stabla. Svaki čvor ima ikonu i ime te kraći opis. Postoje više vrsta čvorova, a mogu se podijeliti na dvije grupe:

- **komponente**
- **lokacije**



Navigator ne prikazuje sve dijelove modela nego samo one za koje je u modelu zadano da su vidljivi. To je učinjeno tako da se smanji i pojednostavni prikaz, jer već i relativno jednostavni model poput ARM-a ima stotine dijelova od kojih većina krajnjem korisniku uopće nije zanimljiva.

Veličina prikaza u polju *Navigator* ne može se izravno mijenjati, ali je pomoću opcija u izborniku *Options* moguće odabrati veličinu fonta (koji onda vrijedi i za neka druga polja kao što su *Display* i *Dump* te *StackView* i *MemView*).

7.2.1 Komponente u polju Navigator

Komponente u polju *Navigator* su **čvorovi koji u sebi sadrže druge čvorove, tj. podkomponente i lokacije**. Specijalni slučaj komponente je *System* ili korijenska komponenta koja predstavlja model i koja nema roditelja. Prikazana je na vrhu polja *Navigator* kao tri šarene lego kockice. Sve ostale komponente imaju ikonu sive lego kocke i mogu se otvarati/zatvarati na uobičajeni način pomoću znaka *+/-* pokraj ikone.

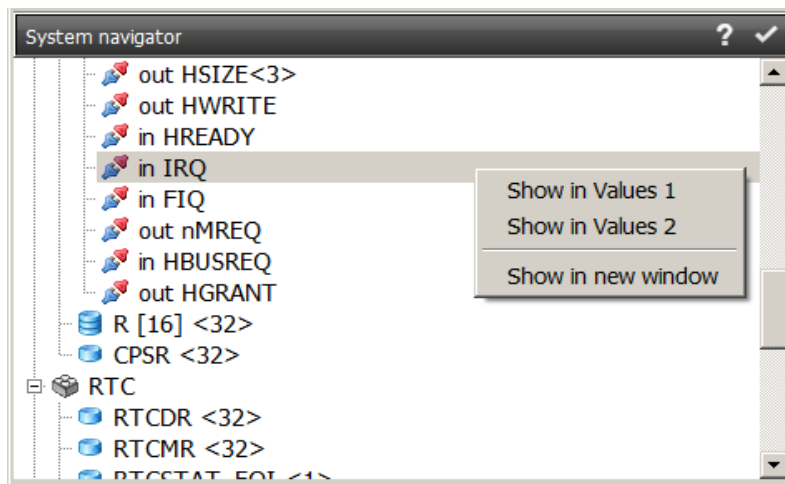
Sve komponente imaju kontekstni izbornik posve identičan kontekstnim izbornicima tih istih komponenata prikazanih u polju *Preview*. Na gornjoj slici je prikazan kontekstni izbornik procesora *ARM* (usporedite s istim izbornikom u potpoglavlju 6.1.2.1). Također je moguće napraviti **dvostruki klik na komponentu** što će automatski prikazati tu komponentu u glavnom polju *Preview*.

7.2.2 Lokacije u polju Navigator

Lokacije u *Navigatoru* su dijelovi modela koji sadrže/pamte neki podatak. Lokacije se uvijek nalaze **u jednoj od komponenata** i nemaju daljnjih dijelova, tj. lokacije su vršni čvorovi stablastog prikaza. Postoje sljedeće vrste lokacija:

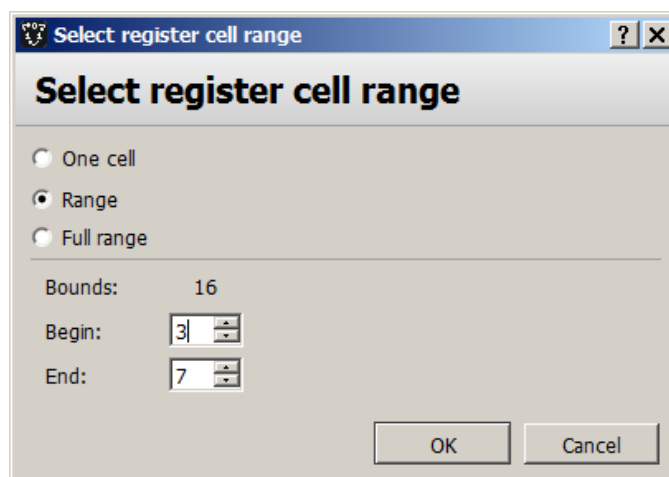
- **žice:** prikazane ikonom plave i crvene žice
 - uvijek su grupirane u dodatnom čvoru *Wires* kako bi se smanjila duljina prikaza
 - uz svaku žicu piše njena širina u bitovima ili ništa ako je žica jednobitna
 - uz svaku žicu stoji naznaka *wire* da se radi o žici ili *w-or* i *w-and* koje označavaju dvije specijalne vrste jednobitnih žica: *wired-or* i *wired-and*
- **priključci:** prikazane ikonom plave i crvene utičnice
 - uvijek su grupirani u dodatnom čvoru *Pins* kako bi se smanjila duljina prikaza
 - uz svaki priključak piše njegova širina u bitovima ili ništa ako je priključak jednobitni
 - uz svaki priključak piše naznaka *"in"*, *"out"* ili *"inout"* kao naznaka smjera priključka
- **registri:** prikazani svijetloplavim ikonama raznih oblika
 - nisu nikada grupirani
 - uz svaki registar piše njegova širina u bitovima ili ništa ako je priključak jednobitni
 - postoje sljedeće vrste registara:
 - bezdimenzionalni /obični registri: prikazani niskim/jednostrukim valjkom
 - jednodimenzionalni registri: prikazani višestrukim/visokim valjkom (služe kao jednodimenzionalna polja (engl. *array*) primjerice kao memorija ili skup registara opće namjene)
 - uz njih je navedeno koliko elemenata imaju
 - punjivi (engl. *loadable*) jednodimenzionalni registri: prikazani višestrukim/visokim valjkom s crvenom strelicom (specijalni slučaj memorije u koju se puni asemblerski program, obično su sakriveni iz prikaza u polju *Navigator*)
 - uz njih je navedeno koliko elemenata imaju
 - višedimenzionalni registri, prikazani matricom (u praksi se rijetko koriste u modeliranju)
 - uz njih je navedeno koliko elemenata imaju za svaku dimenziju zasebno

S lokacijama je moguće napraviti samo jednu akciju, a to je prikazivanje u poljima *Display* (objašnjeni su u potpoglavlju 7.3). Za to se koristi ili dvostruki klik na lokaciji ili njen kontekstni izbornik. Jedine dodatne akcije imaju punjivi jednodimenzionalni registri, a one odgovaraju akcijama prikazanim za kontekstni izbornik memorijske komponente koji je objašnjen u 6.1.2.2. No, kako su ovi registri obično sakriveni, a iste akcije dostupne su u memorijskoj komponenti u poljima *Navigator* i *Preview*, ovdje nije prikazan njihov izgled. Općeniti kontekstni izbornik za lokacije izgleda ovako.



Na slici je prikazan slučaj kad su već otvorena dva polja *Display* s nazivima *Values 1* i *Values 2*. Desni klik na lokaciju otvorit će kontekstni izbornik u kojem se nudi da se dotična lokacija prikaže u jednom od postojećih polja *Display* ili da se otvori novo *Display* polje (*Show in new window*). Dvostruki lijevi klik na lokaciji će automatski prikazati lokaciju u prvom polju *Display* iz kontekstnog izbornika.

Ako se u *Navigatoru* odabere polje registara, onda se ono ne prikazuje odmah u polju *Display*, nego se prethodno otvara dijaloški prozor u kojemu treba zadati elemente polja koje se žele prikazivati. Prikazan je primjer kad je odabrano polje općih registara R0-R15 procesora *ARM* koji su jednodimenzionalno polje veličine 16.



U gornjem dijelu prozora su tri gumba koji tvore tzv. *radio-buttons* grupu, što znači da se mora odabrati točno jedan gumb. U donjem dijelu prozora zadaju se indeksi elemenata koje se želi prikazati. Moguće je odabrati:

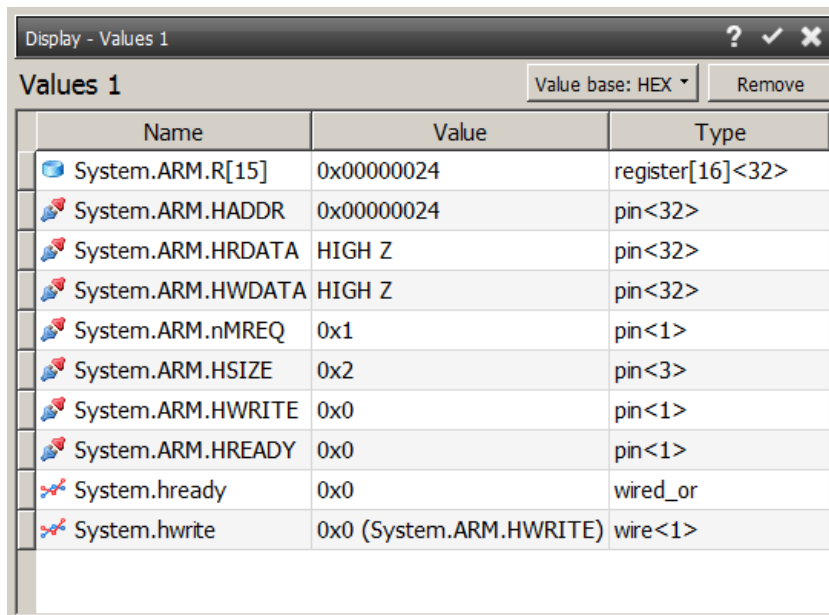
- prikaz samo jednog elementa polja (*One cell*) čiji indeks onda treba zadati
- prikaz niza elemenata od početnog do završnog indeksa (*Range*) i onda treba zadati oba indeksa
- prikaz svih elemenata u polju (*Full range*) i tada ne treba ništa dodatno zadati

Kod odabira u velikim poljima postoji ograničenje broja elemenata koji se mogu zadati (500) jer se inače jako usporava rad sa simulatorom.

7.3 Polje Display

Polje **Display** prikazuje statička stanja (vrijednosti) u lokacijama. Statičko stanje znači stanje tijekom pauzirane simulacije. To je ujedno i glavni nedostatak polja **Display**: u tijeku simulacije u njemu nema prikaza.

Također, većina važnih lokacija trebala bi imati animaciju stanja u poljima **Preview**, **MemView** i **StackView**, tako da polje **Display** ima prvenstvenu uporabnu vrijednost kod modeliranja, kada treba pratiti stanja svih lokacija i kada animacije u modelu još nisu implementirane. Zbog svega toga će **uporaba polja Display** biti rijetka za krajnje korisnike. Polje **Display** se koristi zajedno s poljem **Navigator** jer se samo u polju **Navigator** mogu zadati lokacije koje će se prikazivati u polju **Display**. Korisnik može stvarati željeni broj polja **Display**, uklanjati ih iz simulatora, te ih sakrivati i prikazivati. Primjer jednog polja **Display** prikazan je slikom.



Name	Value	Type
System.ARM.R[15]	0x00000024	register[16]<32>
System.ARM.HADDR	0x00000024	pin<32>
System.ARM.HRDATA	HIGH Z	pin<32>
System.ARM.HWDATA	HIGH Z	pin<32>
System.ARM.nMREQ	0x1	pin<1>
System.ARM.HSIZE	0x2	pin<3>
System.ARM.HWRITE	0x0	pin<1>
System.ARM.HREADY	0x0	pin<1>
System.hready	0x0	wired_or
System.hwrite	0x0 (System.ARM.HWRITE)	wire<1>

Polje **Display** sastoji se od već objašnjene naslovne trake ispod koje je dodatna traka s naslovom prikazanim većim slovima (**Values 1**) i dva gumba te glavni dio prozora - tablica s prikazom vrijednosti odabranih lokacija.

Veliki naslov može se kliknuti i tada se na tom mjestu otvara maleno *editabilno* polje u kojemu se može zadati neki drugi naslov. Naslovi se koriste samo za identifikaciju polja **Display**, s obzirom da ih može biti više i ako se ne žele koristiti *defaultno* dodijeljeni naslovi čiji oblik je **Values N**.

Gumb **Value base** omogućuje odabir brojevnice baze (binarna, dekadna i heksadekadna) kojima se ispisuju vrijednosti lokacije. Baza vrijedi za sve lokacije u pripadnom polju **Display** - nije moguće odabrati drugu bazu za pojedinu lokaciju.

Gumb **Remove** omogućuje uklanjanje pojedine lokacije iz polja **Display** i to tako da se prvo mišem odabere jedna ili više lokacija, a zatim se klikne na gumb **Remove**. Inače jedna lokacija može biti prikazana u više polja **Display**.

Tablica s lokacijama ima tri stupca i proizvoljan broj redaka. Stupcima se može mijenjati redoslijed i širina uobičajenim povlačenjem zaglavlja stupaca ili granica između stupaca. Redovima se može mijenjati redoslijed tako da se mišem uhvati malo sivo polje na lijevoj strani svakog reda i onda se opet uobičajenim povlačenjem redak odvuče na željeni položaj. Stupci su sljedeći:

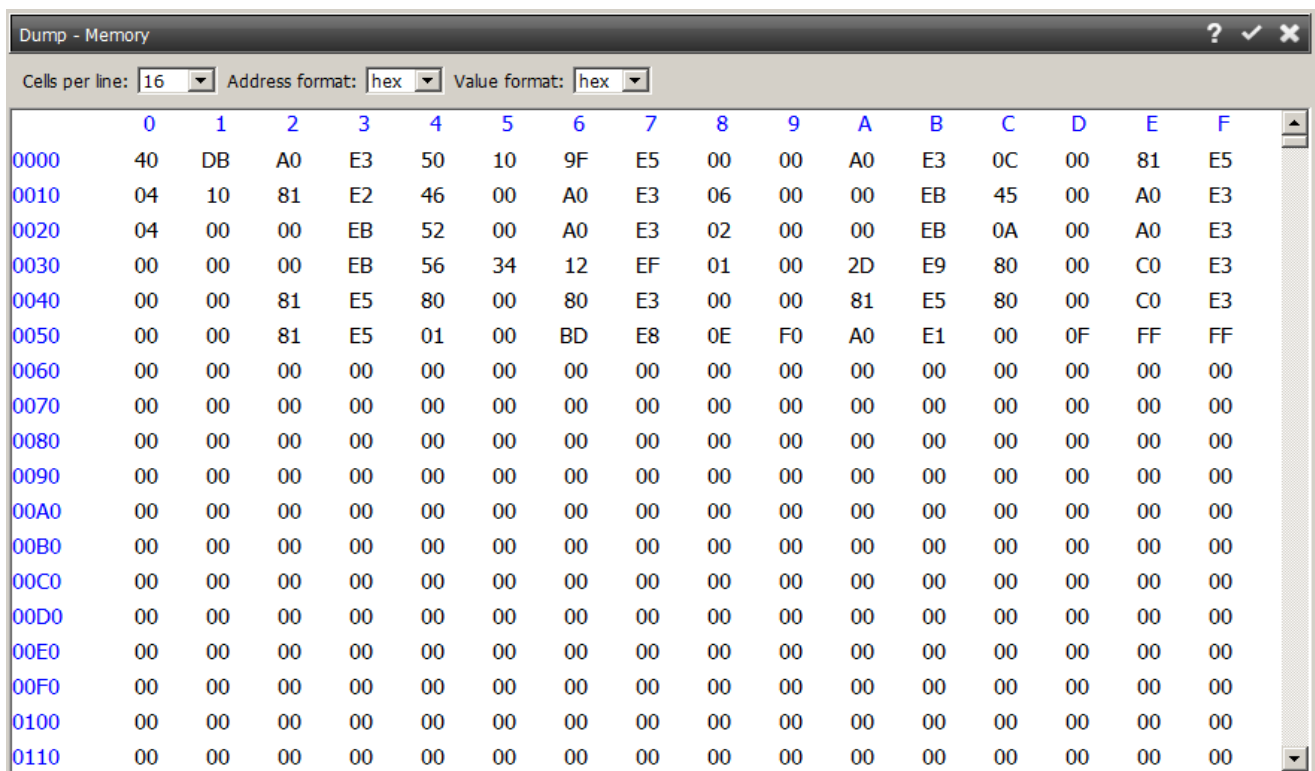
- Stupac **Name** prikazuje ikonu i ime lokacije (uključujući i imena svih nadkomponenti).
- Stupac **Value** prikazuje vrijednost lokacije. Za priključke se zapravo radi o vrijednosti žice/sabirnice na koju je priključak spojen.
 - Žice osim normalne numeričke vrijednosti može imati i specijalnu vrijednost "HIGH Z" kada njome nitko ne upravlja.
 - Za žice se osim vrijednosti ispisuje i ime vlasnika tj. priključka koji upravlja tom žicom.
 - Za *wired-and* i *wired-or* žice se vlasnik ne ispisuje jer njima može upravljati proizvoljan broj priključaka.
- Stupac **Type** prikazuje kratki opis lokacije sa širinom u bitovima, donekle slično kao i u **Navigatoru**.

Veličina prikaza u polju **Display** ne može se izravno mijenjati, ali je pomoću opcija u izborniku **Options** moguće odabrati veličinu fonta (koji onda vrijedi i za neka druga polja kao što su **Navigator** i **Dump** te **StackView** i **MemView**).

7.4 Polje Dump

Polje **Dump** prikazuje statička stanja (vrijednosti) u lokacijama memorije. Statičko stanje znači stanje tijekom pauzirane simulacije. To je ujedno i glavni nedostatak polja **Dump**: u tijeku simulacije u njemu nema prikaza. Također, za memorijske lokacije pogodnija je animacija stanja u poljima **MemView** i **StackView**, tako da polje **Dump** ima prvenstvenu uporabnu vrijednost kod modeliranja, kada treba pratiti stanja svih lokacija i kada animacije u modelu još nisu implementirane. **Zbog svega toga će uporaba polja **Dump** biti rijetka za krajnje korisnike.**

Dump prikazuje stanje **svih** memorijskih lokacija (što znači da to može biti vrlo veliko polje) i to u sirovom obliku, bez grupiranja i bez prikazivanja vrste podatka u lokacijama. Polje **Dump** otvara se iz kontekstnog izbornika memorijske komponente (6.1.2.2) - bilo iz polja **Preview** ili iz **Navigatora**. Korisnik može stvoriti najviše jedno polje **Dump** po jednoj memoriji i može ga sakrivati i prikazivati. Primjer jednog polja **Dump** prikazan je slikom.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	40	DB	A0	E3	50	10	9F	E5	00	00	A0	E3	0C	00	81	E5
0010	04	10	81	E2	46	00	A0	E3	06	00	00	EB	45	00	A0	E3
0020	04	00	00	EB	52	00	A0	E3	02	00	00	EB	0A	00	A0	E3
0030	00	00	00	EB	56	34	12	EF	01	00	2D	E9	80	00	C0	E3
0040	00	00	81	E5	80	00	80	E3	00	00	81	E5	80	00	C0	E3
0050	00	00	81	E5	01	00	BD	E8	0E	F0	A0	E1	00	0F	FF	FF
0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Polje **Dump** sastoji se od već objašnjene naslovne trake ispod koje je dodatna traka koja omogućuje minimalni utjecaj na format prikaza. Ispod toga se nalazi područje koje zauzima većinu površine i koje prikazuje podatke.

Prikaz je zapravo tabličnog formata i organiziran je po adresama koje rastu po stupcima, a zatim sljedeća dresa prelazi u sljedeći redak. **Plavom bojom** prikazani su dijelovi **adrese**: u najgornjem retku su niži dijelovi adrese, a u lijevom stupcu su viši dijelovi adrese. Podatci su prikazani u tablici - **svaka memorijska lokacija (uobičajeno je to bajt) prikazana je zasebno.**

U traci iznad prikaza se može birati:

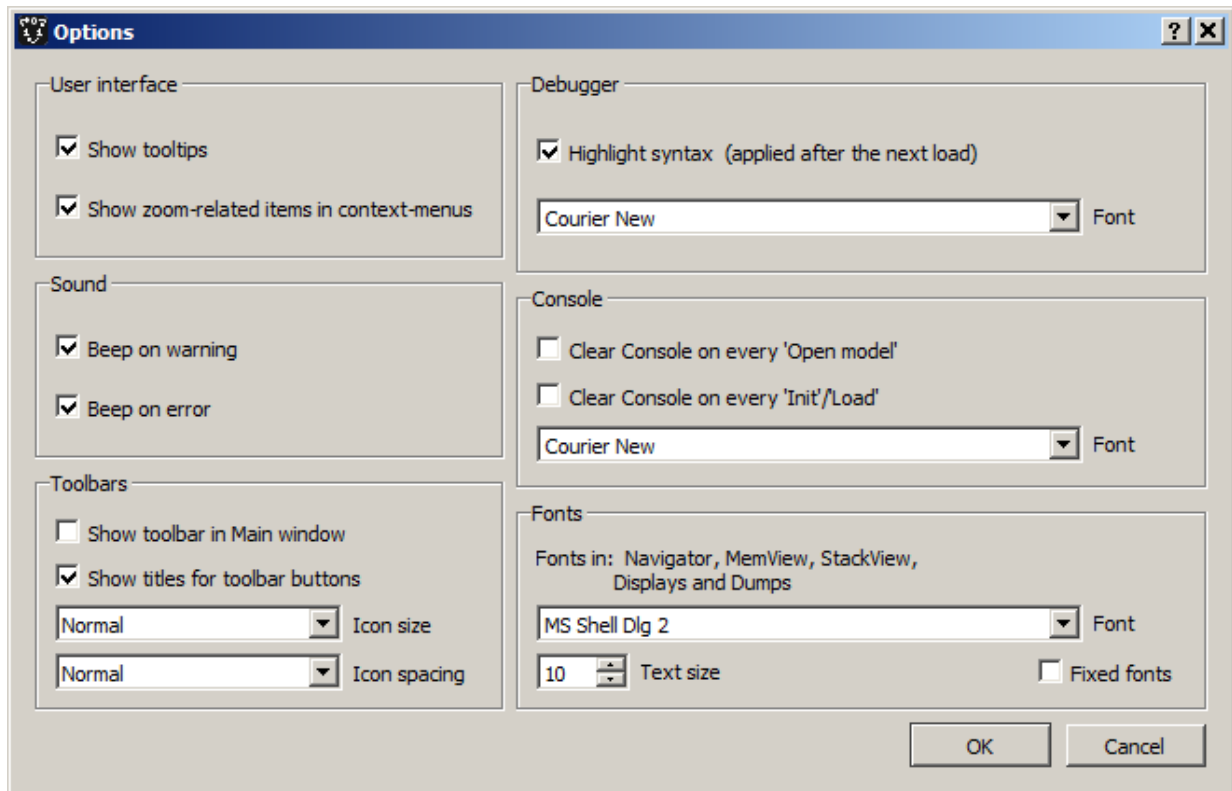
- Broj podataka u retku (**Cells per line**)
- Brojevnja baza plavog prikaza adresa (**Address format**)
- Brojevnja baza prikaza podataka (**Value format**)

Veličina prikaza u polju **Dump** ne može se izravno mijenjati, ali je pomoću opcija u izborniku **Options** moguće odabrati veličinu fonta (koji onda vrijedi i za neka druga polja kao što su **Navigators** i **Display** te **StackView** i **MemView**).

8 Opcije

8.1 Raspoložive opcije

Sve postojeće opcije dostupne su iz **izbornika** *Options* u kojemu se nalazi samo jedna opcija istog imena *Options*. Stavka *Options* otvara posebni **prozor** *Options* za zadavanje svih opcija.



Sve **opcije** služe za **namještanje izgleda i ponašanja sučelja** koje vrijedi za cijelu ili za veći dio aplikacije pa možemo reći da su to "**općenite postavke**". Kako je u prethodnim poglavljima objašnjeno, mnoge "**specifične**" **postavke** zadaju se **izravno u poljima** pomoću kontekstnih izbornika. Postavke koje se zadaju izravno, kao i opcije u prozoru *Options* uglavnom imaju efekt koji je odmah vidljiv, uz par iznimaka.

Ako korisnik odabere gumb *OK* na dnu prozora *Options*, onda sve zadane postavke postaju "potvrđene". Ako se odabere gumb *Cancel*, onda sve opcije zadane u prozoru *Options*, bez obzira što su bile odmah vidljive, bivaju "poništene" i vraćaju se na stanje prije otvaranja prozora *Options*. Šest grupa opcija u prozoru *Options* su:

Opcije *User interface* omogućuju odabir nekih općenitih ponašanja sučelja:

- *Show tooltips*: dozvoljava ili zabranjuje prikaz tzv. *tooltipova* koji se otvaraju uglavnom na raznim gumbima/ikonama i sličnim elementima (ne na svima).
- *Show zoom-related items in context-menus*: dozvoljava ili zabranjuje prikaz stavki za *zoomiranje* (*zoom in*, *zoom out* i *reset zoom*) u kontekstnim izbornicima polja *Preview* i *Debug* te *Console*.

Opcije *Sound* reguliraju zvukove u aplikaciji:

- *Beep on warning*: dozvoljava ili zabranjuje zvučni signal u slučaju ispisa upozorenja
- *Beep on error*: dozvoljava ili zabranjuje zvučni signal u slučaju ispisa greške

Opcije *Toolbars* reguliraju izgled i ponašanje alatnih traka u glavnom prozoru i *Debug* poljima:

- *Show toolbar in Main window*: dozvoljava ili zabranjuje prikaz alatne trake u glavnom prozoru. Alatna traka od *Debug* polja sakriva se izravno u polju *Debug*. Ove dvije trake su jednake (osim u slučaju modela s više

procesora). Za modele s jednim procesorom prirodnije je koristiti traku u polju [Debug](#), a za više procesora može biti zgodno koristiti i jedne i druge alatne trake.

- [Show title for toolbar buttons](#): Ispod svakog gumba može biti prikazan njegov naziv. Ova stavka prikazuje ili sakriva taj naziv. Naziv može biti zgodan korisnicima koji su tek počeli koristiti simulator, a kasnije ih je bolje isključiti jer samo povećavaju dimenzije alatne trake.
- [Icon size](#) i [Icon spacing](#): odabire se jedna od pet razina veličine ikona i razmaka između ikona u alatnoj traci

Opcije [Debugger](#) namještaju ponašanje i izgled polja [Debug](#):

- [Highlight Syntax](#): zadaje da li u polju [Debug](#) treba asemblerski program prikazivati sa ili bez bojanja sintakse. Ovo je jedna od rijetkih opcija čiji efekt nije vidljiv odmah, nego tek kasnije kad se učita novi asemblerski program (ili ako se ponovno učita stari).
- [Font](#): odabire vrstu slova/font za [Debug](#) prozor, pri čemu se nude samo fontovi koji imaju fiksnu širinu. Preporučljivi su fontovi koji nisu preširoki i koji imaju različit prikaz velikog slova **O** od **znamenke ništice**, jer je to banalan ali čest izvor grešaka i zabuna u asemblerskom programiranju. Autor ovog dokumenta osobno preferira ☺ font losevka koji je besplatno dobavljiv s weba.

Opcije [Console](#) namještaju ponašanje i izgled polja [Console](#):

- [Clear Console on every "Open model"](#): zadaje automatsko brisanje polja [Console](#) na svako otvaranje modela. krajnjim korisnicima ova opcija nije od prevelike koristi, jer je ona namijenjena uglavnom onima koji sami izrađuju svoje modele i višekratno ih učitavaju u simulator pri traženju grešaka u modelima. Krajnji korisnici će jednom na početku rada učitati model i nakon toga do kraja rada stalno koriste isti model.
- [Clear Console on every "Init/load"](#): zadaje automatsko brisanje polja [Console](#) na početku svake runde simuliranja. Ova opcija je korisna ako model dio stanja prikazuje u [Console](#) ili je korisnik uključio prikaz dijela stanja u polju [Console](#). Budući da su ispisi u [Console](#) obično opsežni, onda se na ovaj način oni čine preglednijima i kraćima, jer se barem obrišu ispisi od stare/prethodne runde simuliranja.
- [Font](#): odabire vrstu slova/font za polje [Console](#) - isto kao što je opisano za fontove prethodnoj grupi opcija.

Opcije [Fonts](#) odabiru font i njegovu veličinu za polja: [MemView](#), [StackView](#), [Navigator](#), [Display](#) i [Dump](#):

- [Font](#): odabire vrstu slova/font za nabrojena polja - slično kao što je iznad opisano za odabir fonta u poljima [Debug](#) i [Console](#). Razlika je što se ovdje nude **svi** fontovi. Ako se želi filtrirati fontove tako da se u izborniku prikazuju samo fontovi fiksne širine, onda se uključi tzv. *checkbox Fixed fonts*.
- [Text size](#): omogućuje zadavanje veličine slova (izražena u točkama) za nabrojena polja.

8.2 Spremanje i obnavljanje opcija i postavki

Opcije se automatski spremaju i obnavljaju kod izlaska i pokretanja programa, kao što je spomenuto u 2.2.1. Osim opcija iz prozora [Options](#), pamti se još i cijeli niz drugih postavki, od kojih su ovdje nabrojene neke:

- Položaj i veličina glavnog prozora i svih polja u njemu te sakrivenost svih polja
- Uključenost animacije i brzina simulacije
- Sakrivenost alatne trake te veličina prikaza u glavnom i pomoćnom polju [Preview](#) i zadnjem dodatnom polju [Preview](#)
- Sakrivenost alatne trake, stupca s prekidnim točkama i stupca s adresama u polju [Debug](#), kao i brojevne baze za adrese te veličina prikaza u polju [Debug](#)
- Sakrivenost i redoslijed stupaca i zadnja baza u polju [MemView](#)
- Veličina prikaza u polju [Console](#)
- Zadnjih deset modela koji su bili korišteni