Primljen: 22.04.2015. Stručni rad
Prihvaćen: 29.04.2015. UDK 004.414.23

Implementacija prototipnog simulatora mikroupravljača tvrtke Microchip familije PIC16

Implementation of prototype simulator of Microchip microcontroller PIC16

[1]Danijel Korent, [2]Mihael Kukec
[1]student Međimurskog veleučilišta u Čakovcu
[2]Međimursko veleučilište u Čakovcu, Bana Josipa Jelačića 22a, 40000 Čakovec, Hrvatska
e-mail: [1] danijel.korent@cetitec.com, [2] mihael.kukec@mev.com

Sažetak: *Ovim radom opisuje se ostvarenje prototipa simulatora temeljenog na mikroupravljaču tvrtke Microchip familije PIC16. Opisana je općenita povijest mikroupravljača, kako i kada su se pojavili, razvoj tržišta, te povijest 8-bitnih mikroupravljača familije PIC. U nastavku rada navedene su osnovne značajke arhitekture PIC16 mikroupravljača te prednosti i nedostaci ove arhitekture u odnosu na arhitekture drugih proizvođača mikroupravljača, poput mikroupravljača 8051 tvrtke Intel. Također, opisane su značajke instrukcijskog skupa, fizička organizacija memorija i sabirnica unutar mikroupravljača te način na koji se pristupa radnoj memoriji, registrima i sklopovima. Implementacija prototipnog simulatora i parsera izrađena je korištenjem objektno orijentirane programske paradigme te realizirana korištenjem programskog jezika Java. Simulator je implementiran na način da je funkcionalnost mikroupravljača podijeljena na niz podfunkcija mikroupravljača te se svaka podfunkcija simulira u zasebnom razredu. Razredi su potom hijerarhijski spojeni u smislenu cjelinu korištenjem kompozicije objekata i nasljeđivanjem razreda.*

Ključne riječi: *mikroupravljač, simuliranje, arhitektura, Java*

Abstract: *This article describes the realization of a prototype simulator based on the microcontroller of Microchip PIC16 family. The article describes general history of the microcontrollers, when and how they emerged, development of the microcontroller market, and history of 8-bit PIC family. In follow-up, the article summarizes the basic features of the PIC16 microcontroller architecture, and the advantages and disadvantages of this*

architecture in relation to the architecture of other microcontroller manufacturers, such as Intel's 8051 microcontroller. The article also describes basic features of instruction set, physical organization of memories and buses in microcontroller, and how the random-access memory, registers and peripherals are accessed. Implementation of a prototype simulator and parser was done using object oriented programming paradigm, and realized using programming language Java. The simulator is implemented in a way that the functionality of the microcontroller is divided into a number of sub-functions of microcontroller, and each sub-function is simulated in a separate class. Classes are then hierarchically connected in a meaningful whole using the composition of objects and class inheritance.

Keywords: *microcontroller, simulation, architecture, Java*

1. Uvod

Ubrzo nakon pojave prvih mikroprocesora javlja se potreba za procesorski jednostavnijim uređajima koji će uz integriran procesor, na istom čipu također imati i integrirane sve potrebne periferije za potpuno funkcionalno računalo. Četrdeset godina kasnije u prosječnom kućanstvu te automobilima može se nalaziti nekoliko desetina takvih uređaja. Trenutno, više od polovice svih prodanih procesora čine 8-bitni mikroupravljači.

Zbog specifičnosti mikroupravljača praćenje tijeka izvršavanja programa, stanja registara i memorijskih lokacija nije moguće vršiti softverski na samom uređaju kao što je to slučaj sa stolnim računalima, nego se koristi simulator na stolnom računalu ili dodatno sklopovlje fizičko povezano na posebne nožice mikroupravljača. Trenutno najpopularniji protokol ove namjene je tzv. JTAG (engl. *Joint Test Action Group*), međutim kod starijih ili jednostavnijih modela mikroupravljača često ne postoji opcija korištenja dodatnog sklopovlja za praćenje rada i ispravljanje grešaka, pa jedina opcija ostaje improvizacija s dodatnim sklopovljem za vizualnu indikaciju i praćenje signala na vanjskim sabirnicama ili korištenje simulatora.

U radu će se prikazati mogućnost izrade jednostavnog simulatora arhitekture Microchip PIC16 u programskom jeziku Java. Analiziranjem arhitekture mikroupravljača pokušat će se odgovoriti kako oblikovati simulator koristeći objektno orijentiranu paradigmu i programski jezik Java.

2. Nastanak i razvoj mikroupravljača

Prvi mikroupravljač počinje se razvijati iste godine kada je napravljen prvi mikroprocesor, 1971. godine. Te godine je dovršen dizajn prvog mikroprocesora tvrtke Intel za potrebe tvrtke Busicom Corp koja proizvodi kalkulatore, te ga ujedno iste godine komercijalno nudi na tržištu. Radi se o vrlo poznatom 4-bitnom procesoru Intel 4004, koji je uz tri dodatna čipa omogućavao implementaciju potpuno funkcionalnog računala. Iste godine Texas Instruments proizvodi još napredniji integriraniji čip, 4-bitni mikroprocesor s integriranom radnom memorijom (engl. *Random Access Memory - RAM*) i stalnom memorijom (engl. *Read-Only Memory*) za pohranu programskog koda te s ulazno-izlaznim (engl. *Input/Output - I/O*) funkcijama, što ga čini potpuno funkcionalnim računalom u jednom čipu te ujedno i prvim mikroupravljačem. Radi se o relativno nepoznatom TMS1000 kojeg Texas Instruments (Texas Instruments, 1976.) ne nudi na tržištu, već ga počinje koristiti u svojim vlastitim kalkulatorima.
(http://www.computerhistory.org/semiconductor/timeline/1974-MCU.html).

Tri godine kasnije Texas Instruments svoj mikroupravljač počinje nuditi kao zaseban proizvod te time TMS1000 ujedno postaje i prvi komercijalno dostupan mikroupravljač. Zbog svoje izuzetno niske cijene i integriranosti TMS1000 (http://smithsonianchips.si.edu/augarten/p38.htm) je imao komercijalni uspjeh te je zaživio je u potrošačkoj elektronici; prodano je preko stotinu milijuna komada ovog čipa. Kao odgovor na tržištu se pojavljuju mikroupravljači drugih proizvođača; General Instrument 1975. godine izdaje prvi 8-bitni mikroupravljač i preteču današnje PIC serije - PIC1640, a 1977. godine pojavljuju se mikroupravljači MC6801 tvrtke Motorola i 8048 tvrtke Intel. (http://www.ukessays.com/essays/information-technology/examining-the-first-microprocessor-chip-devices-information-technology-essay.php).

Ubrzo se tržište procesora specijalizira u dva odvojena smjera. Jedno je tržište procesora opće namjene koji se koriste u stolnim računalima i kompleksnijim sustavima, gdje je potrebna veća računalna snaga te za to vrijeme veliki adresni, poput 8-bitnog Zilog Z80, 16-bitnog Intel 8086 i 32-bitnog Motorola MC68000. Drugo je tržište jeftinijih čipova s jednostavnijim procesorima koji teže prema što boljim I/O mogućnostima i što većoj integriranosti raznih periferija u jednom čipu, poput brojila (engl. *timer*), prekidnog sustava (engl. *interrupt controller*) ili sklopa za asinkronu komunikaciju (engl. *Universal Asynchronous Receiver/Transmitter - UART*), zbog čega postaju idealni u industriji potrošačke elektronike te kao pomoćni čipovi u kompleksnijim računalnim sustavima.

3. Značajke 8-bitnih mikroupravljača arhitekture PIC

Glavna odlika 8-bitne familije PIC mikroupravljača je njihova jednostavnost i niska cijena. PIC mikroupravljači imaju vrlo mali broj procesorskih instrukcija, samo 33 instrukcije kod osnovnih modela do 83 instrukcije kod naprednijih modela. Uz mali instrukcijski skup, same instrukcije su također izrazito ortogonalne, tj. nema mnogo instrukcija koje mogu koristiti samo određene registre ili načine adresiranja, što dodatno pojednostavljuje učenje i pisanje programskog koda. Većina registara posebne namjene koji kontroliraju procesor i periferije memorijski mapirani su u procesorski adresni prostor te im se pristupa jednako kao i ostalim registrima u memoriji, što omogućava jednostavnu kontrolu CPU-u i ostale periferije bez potrebe za dodatnim instrukcijama. Nabrojane značajke znatno pojednostavljuju i olakšavaju učenje ove arhitekture, što je ujedno i jedan od glavnih razloga njihove velike popularnosti (http://ww1.microchip.com/downloads/en/devicedoc/33023a.pdf).

Mane PIC arhitekture su mali nelinearan pristup radnoj memoriji, tj. podjela memorije u tzv. „banke“, malen stog te samo jedna prekidna rutina. Također jedna od mana tako minimalnog instrukcijskog skupa i jednostavne arhitekture je neprilagođenost kompajlerima viših programskih jezika, poput programskog jezika C. Ovaj nedostatak se u novijim PIC mikroupravljačima donekle ublažio proširenjima instrukcijskog skupa i dodatnim načinima adresiranja memorije.

PIC arhitekturu obilježavaju sljedeće značajke:

- RISC arhitektura
- Harvard arhitektura
- Akumulatorska arhitektura
- Nelinearan pristup memoriji
- Izrazito ortogonalan instrukcijski skup.

3.1. RISC arhitektura

PIC mikroupravljači koriste procesor koji ima reducirani instrukcijski skup (engl. *Reduced Instruction Set Computing - RISC*), što znači da imaju mali broj jednostavnih instrukcija. Glavna prednost arhitekture RISC naspram arhitekture s kompleksnim skupom instrukcija (engl. *Complex Instruction Set Computing - CISC*) je u tome što jednostavnije instrukcije omogućuju jednostavniji dizajn procesora i zahtijevaju manji broj taktova za izvršenje pojedinačne instrukcije. Ove značajke omogućile su da se strojni ciklus mikroupravljača

familije PIC16 izvršava u 4 radna takta signala vremenskog vođenja, što je u to vrijeme bilo brzo. Potrebno je napomenuti da se instrukcije grananja izvršavaju u dva strojna ciklusa, dok se sve ostale instrukcije izvršavaju u jednom strojnom ciklusu.

Uslijed ovih značajki, PIC mikroupravljači su u trenutku pojave na tržištu bili iznimno brzi naspram ostale 8-bitne konkurencije. Prvi službeni mikroupravljač arhitekture PIC (PIC16C54) imao je maksimalan radni takt od 40MHz, vrlo visok radni takt čak i za tadašnje procesore za stolna računala, što mu je omogućavalo postizanje iznimno niskih latencija odaziva. Uz četiri radna takta po strojnom ciklusu, ovaj mikroupravljač je postizao brzine do teoretskih 10 MIPS-a (milijuna instrukcija u sekundi). Za usporedbu tadašnji najveći konkurenti, Motorola 68HC11 postiže maksimalno 5Mhz sa svojim CISC procesorom, dok je Intelova 8051 CISC arhitektura radila na 12MHz, s jednim strojnim ciklusom na 12 radna takta. Time se postizala brzina do teoretskih 1MIPS, dok se u praksi radilo o oko 0.6 MIPS-a zbog velikog broja instrukcija koje koriste dva strojna ciklusa, a neke čak i četiri strojna ciklusa. No treba imati na umu da ove brojke nisu direktno usporedive, jer je u mnogo slučajeva potrebno izvršiti od nekoliko pa čak do desetak PIC instrukcija za obavljanje zadaće koja se na 8051 arhitekturi može obaviti jednom instrukcijom. Prednost brzog izvršavanja instrukcija PIC arhitekture izgubila se pojavom mnogih derivata 8051 arhitekture s jednim radnim taktom po oktetu instrukcije, kao i pojavom Atmelove AVR arhitekture koja radi na jednom radnom taktu po strojnom ciklusu.

3.2. Harvardska arhitektura

Kod harvardske arhitekture programska i radna memorija fizički su odvojene pri čemu svaka ima vlastite adresne i podatkovne linije. Konkretno kod arhitekture PIC, programska i radna memorija nalaze se na potpuno odvojenim sabirnicama, koje su neovisne jedna o drugoj. Ovaj odabir arhitekture omogućava dvije bitne stvari za ostale značajke PIC arhitekture: (I) programskoj i radnoj memoriji moguće je pristupiti istodobno, (II) zbog odvojenih sabirnica programska i radna memorija mogu biti različite širine.

Istovremen pristup programskoj i radnoj memoriji iskorišten je implementacijom dvostupanjskog instrukcijskog cjevovoda (engl. *instruction pipelining*). Dvostupanjski cjevovod omogućuje da se izvršavanje trenutne instrukcije i dohvat iduće izvode paralelno; u prvom stupnju se obavlja dohvaćanje iduće instrukcije, dok se u drugom stupnju istodobno izvodi izvršavanje trenutne instrukcije. Ovo ubrzava izvođenje svih instrukcija koje ne

mijenjaju programsko brojilo. Promjenom programskog brojila dohvaćena instrukcija u prvom stupnju cjevovoda više nije valjana, te se u idućem strojnom ciklusu ne izvršava već se samo dohvaća nova instrukcija.
(http://ww1.microchip.com/downloads/en/DeviceDoc/ramrom.pdf)

Kod šire sabirnice programske memorije dolazi do još jedne bitne razlike naspram 8051 arhitekture. Iako obje arhitekture koriste instrukcije duže od svoje podatkovne sabirnice, PIC radi samo jedno čitanje programske memorije po instrukciji, dok 8051 mora raditi i do tri čitanja po instrukciji. Kao i većini CISC arhitektura, instrukcije u arhitekturi 8051 variraju dužinom, od 8 bitova pa sve do 24 bitova dužine, ali se prenose sabirnicom fiksne širine; u slučaju arhitekture 8051 dužina iznosi samo 8 bitova. Za razliku od toga, u arhitekturi PIC sve instrukcije su jednake širine (12, 14 ili 16 bitova) te programska memorija i sabirnica imaju isti broj bitova kao i instrukcije, što omogućava da se cijela instrukcija uvijek prenese u jednom čitanju. Ove razlike u arhitekturi nisu rezultat dobrih ili loših inženjerskih odluka, nego su rezultat potpuno različitih ciljeva u mogućnostima i namjeni sklopa. Naime, cilj arhitekture PIC je da bude što jednostavnija kako bi proizvodnja bila što jeftinija te jednaka dužina svih instrukcija i samo nekoliko vrsta formata instrukcija omogućuju vrlo jednostavnu implementaciju upravljačke jedinice mikroupravljača. S druge strane arhitektura 8051 dizajnirana je da bude što svestranija te da, između ostalog, omogućava dohvaćanje programskog koda s vanjske memorijske jedinice kojima je u to vrijeme standardna širina podatkovne sabirnice bila 8 bitova. Budući da je interna programska sabirnica direktno spojena na vanjsku sabirnicu, interna sabirnica je morala biti jednake dužine kao i vanjska. U slučaju različitih veličina sabirnice bilo bi potrebno posebno sklopovlje za dohvat instrukcije za svaku od sabirnica.
(http://galia.fc.uaslp.mx/~cantocar/microprocesadores/EL_Z80_PDF_S/8051.PDF)

3.3. Akumulatorska arhitektura

Kod svih aritmetičkih i logičkih operacija koje se izvode između dva operanda implicitno se podrazumijeva da je jedan od operanda uvijek radni registar "W". Drugi operand može biti dvije vrste, konstanta "k" koji se u dokumentima naziva "literal" ili vrijednost sa memorijske adrese "f". Rezultat operacije se uvijek sprema u jedan od dva korištena operanda, u radni registar "W" ili u memorijsku lokaciju "f", ukoliko se adresiralo radnu memoriju. Akumulatorsku arhitekturu također koristi i 8051 arhitektura.

U usporedbi s drugim arhitekturama prednost je u tome što omogućava relativno jednostavnu fizičku izvedbu procesora, što znači i nižu cijenu konačnog uređaja. Prednost drugih arhitektura je ta što dva ulazna operanda i izlazni operand mogu biti bilo koji od skupa registara poput registar-registar arhitekture (npr. MIPS i ARM) ili bilo koji registar i memorijska lokacija poput registar-memorija arhitekture (npr. x86). Mana ovoj arhitekturi je ta što je često potrebno izvršiti nekoliko instrukcija da bi se izvela operacija koju registar-registar i registar-memorija arhitekture mogu izvršiti jednom instrukcijom.

3.4. Nelinearan pristup memoriji

Jedna od mana PIC mikroupravljača je ta što nemaju linearni pristup memoriji, već je memorija podijeljena u tzv. "banke", pa se moraju koristi određeni bitovi u registrima posebne namjene kako bi se mogao adresirati cijeli adresni prostor radne ili programske memorije. Prilikom direktnog pristupa radnoj memoriji memorijska adresa je ukodirana u instrukciju koja je uvijek fiksne dužine. Npr. kod PIC mikroupravljača s 12-bitnim instrukcijama aritmetičko-logičke instrukcije imaju 5 bitova namijenjenih za adresu radne memorije. To znači da instrukcija može direktno adresirati samo 32 memorijske adrese. Kako bi se moglo adresirati više memorije, koristi se poseban registar koji kontrolira gornje bitove memorijske adrese prilikom pristupa memoriji. Kod programiranja treba obratiti posebnu pozornost na podjelu memorije jer nepravilno korištenje može uzrokovati greške koje je vrlo teško pronaći. Radna memorija PIC arhitekture je podijeljena na područje registara posebne namjene (engl. *Special Function Registers - SFR*) i područje registara opće namjene (engl. *General Purpose Registers - GPR*). U području registara posebne namjene mapirani su specijalni registri procesora i periferija. Neovisno o kojem području radne memorije se radi, podatku se pristupa istom vrstom instrukcija na jednak način.

PIC arhitektura također podržava i indirektno adresiranje te adresiranje relativno na programski brojač. Kod naprednijih modela mikroupravljača indirektno adresiranje ima linearan pristup radnoj i programskoj memoriji.

3.5. Izrazito ortogonalni instrukcijski skup

Ortogonalni instrukcijski skup je onaj kod kojeg nema posebnih instrukcija koje mogu koristiti samo određene registre ili načine adresiranja. Neortogonalni instrukcijski skup je teže

naučiti i koristiti, jer od programera zahtjeva da pamti i pazi koje operacije može izvršiti nad kojim registrima. Primjer neortogonalnog instrukcijskog skupa je 8051 arhitektura.

PIC arhitektura postiže visoku ortogonalnost zahvaljujući činjenici da koristi samo jedan radni registar W, koji se implicitno podrazumijeva u instrukcijama, dok su skoro svi ostali specijalni registri mapirani u adresni prostor radne memorije te im se pristupa isključivo putem radne memorije, s istim instrukcijama kojima se pristupa i ostalim memorijskim lokacijama. Iznimke su instrukcija WDT koja jedina ima pristup sigurnosnom brojilu (engl. *watchdog timer*) i instrukcija SLEEP koja mikroupravljač postavlja u stanje spavanja.

Instrukcijski skup PIC16 mikroupravljača iznimno je jednostavan i simetričan te se prema formatu može podijeliti u četiri generalne kategorije:

- instrukcije za 8-bitne operacije
- instrukcije za 1-bitne operacije
- instrukcije s 8-bitnim neposrednim vrijednostima
- instrukcije za bezuvjetni skok.

4. Programska implementacija simulatora

Za implementaciju simulatora odabran je programski jezik Java, a biblioteka Swing je odabrana za implementaciju grafičkog sučelja simulatora. Programska implementacija sadrži simulator PIC16F84 mikroupravljača, parser tekstualnog asemblerskog programskog koda u strojni opkod mikroupravljača te grafičko sučelje koje vizualno prikazuje unutarnja stanja mikroupravljača.

4.1. Mikroupravljač PIC16F84A

Kao referenca za oblikovanje aplikacije odabran je mikroupravljač PIC16F84A. Ovaj model pripada sredini skupa familija (engl. *mid-range*) s 35 instrukcija i unutar te familije je jedan od najjednostavnijih modela. Maksimalni radni takt mu je 20MHz, ima 1024x14bit programske memorije, 68B radne memorije, te 64B EEPROM memorije. Podržava prekide s jednom prekidnom rutinom i četiri moguća izvora i ima ugrađen 8-bitni brojač. Također ima ugrađen hardverski stog koji podržava do osam adresa i sigurnosno brojilo (engl. *watchdog timer*) koje pruža zaštitu od beskonačnih petlja.
(http://ww1.microchip.com/downloads/en/devicedoc/35007b.pdf)

Čip u DIP (engl. *Dual In-line Packa*ge) kućištu ima 18 nožica, od čega ih se 13 može koristiti kao ulazne ili izlazne nožice. Nožicama se čita ili mijenja izlazno stanje pomoću specijalnih 8-bitnih registara PORTA i PORTB, svaka nožica mapirana je na jedan od bitova ovih registara. Mijenjanjem sadržaja registara TRISA i TRISB na jednak se način svakoj nožici određuje da li je ulazna ili izlazna. Ako je bit koji kontrolira nožicu postavljen na '0' tada je nožica izlazna, u suprotnom nožica je u ulaznom stanju. Prilikom pokretanja mikroupravljača svi TRIS bitovi postavljeni su na '1', dakle sve nožice postavljene su kao ulazne. Ovaj model također ima i nožicu za vanjski prekid – nožica INT, nožicu za vanjsko resetiranje mikroupravljača – MCLR te jednu nožicu za vanjsku kontrolu 8-bitnog brojača – T0CKI. Od ostalih nožica dvije služe za prihvat radnog takta mikroupravljača, a dvije za napajanje. Neke nožice su multipleksirane, tj. dvije različite funkcije nalaze se na jednoj nožici, stoga nije moguće istovremeno koristiti svih 13 I/O nožica i sve opisane mogućnosti. Pomoću registara posebne namjene određuje se funkcija pojedinih nožica. Kao dio srednje familije proizvoda, ovaj mikroupravljač koristi 13-bitni programski brojač i može adresirati do 8K x 14bit memorije, no samo 1K je fizički implementirano. Mikroupravljač ignorira tri bita najviše vrijednosti u programskom brojaču; npr., ako se napravi skok na adresu 1025, procesor će nastaviti izvođenje programa od adrese br. 2. Potrebno je dodati da se prekidna rutina, ukoliko postoji u programu, smješta na adresu 04h u programskoj memoriji.

Memorijski prostor je podijeljen na dvije banke od 128 lokacija. PIC16F84 ima 16 registra posebne namjene i 68 registara opće namjene. Iako se tih 84 registara može smjestiti u samo jednu banku, to nije napravljeno zbog kompatibilnosti s ostalim mikroupravljačima iz familije. Većina registara mapirano je u obje banke i u tom se slučaju pristupa istom fizičkom registru neovisno aktivnoj adresnoj banci, uz pet iznimaka – registri TMR0/OPTION, PORTA/TRISA, PORTB/TRISB, EEDATA/EECON1, EEADR/EECON2. Na prvih 12 adresa nalaze se mapirani registri posebne namjene, kojima upravljaju procesor i periferije. Na adresama od 12 do 79 nalaze se registri opće namjene, na slici žuto obojano. Ostatak dostupnog adresnog prostora, od adresa 80 do 127, nije fizički implementiran te čitanje tih registara uvijek vraća nulu.

PIC16F84 ima 16 registara posebne namjene mapiranih u adresni prostor radne memorije, pomoću kojih se kontrolira rad procesora i dodatne periferije. Registri PORT i TRIS u ranijem su poglavlju objašnjeni kao registri vezani uz kontrolu fizičkih nožica mikroupravljača. Registri koji upravljaju integriranom EEPROM memorijom su EEDATA, EEADR, EECON1 i EECON2. Registar TMR0 sadrži trenutnu vrijednost integriranog 8-bitnog brojača, dok se u OPTION registru može podesiti djelitelj kako bi se povećao vremenski opseg brojača. U

registru INTCON se može uključiti ili isključiti mogućnost prekida, a svaki se izvor prekida može pojedinačno kontrolirati.

Osim direktnog adresiranja radne memorije, procesor ima mogućnost indirektnog adresiranja. Adresa kojoj se želi indirektno pristupiti upiše se u registar FSR te se čitanjem i pisanjem u registar INDF čita ili mijenja lokacija na koju pokazuje registar FSR. Budući da je FSR registar 8-bitni, a adresni prostor ovog mikroupravljača sastoji se od dvije banke po 128 adresa, indirektnim adresiranjem može se linearno pristupiti cijelom adresnom prostoru mikroupravljača.

Druga dva para adresnih registara su PCL i PCLATH. U registru PCL mapirano je donjih osam bitova programskog brojila, u koji se može čitati i pisati, čime se mijenja tok programa. PCLATH predstavlja gornje bitove programa i nije dio direktno adresiranog programskog brojila, već je zasebni registar koji služi kao međuspremnik. Kada se u PCL zapisuje neka vrijednost, istodobno se sadržaj međuspremnika PCLATH upisuje u gornje bitove programskog brojila. Iz tog se razloga prilikom svakog upisivanja u PCL registar treba pobrinuti da se PCLATH registru nalazi ispravna vrijednost. Čitanje i pisanje PCL registra omogućava adresiranje relativno s obzirom na programsko brojilo, što znači da je moguće unutar koda konstruirati tablicu podataka iz koje se uzimaju podaci na temelju indeksa. Upravo za ovu svrhu dodana je instrukcija RETLW koja prilikom povratka iz potprograma u W registar upiše vrijednost k te time omogućava efikasnu implementaciju tablica s manje potrebnog koda.

No ovdje se i krije potencijalna greška koju je lako napraviti. Prilikom planiranja tablice treba imati na umu početnu poziciju tablice u programskom adresnom prostoru, jer ukoliko dođe do preljeva registra PCL prilikom računanja adrese, procesor će napraviti skok na krivu adresu i izvršiti instrukcije različite od željenih.

4.2. Opis implementacije

U ovom poglavlju opisani su glavni dijelovi implementacije, način na koji su ostvareni opći i posebni registri, hardverski stog i instrukcije mikroupravljača te opis razreda CPU koji sve navedene razrede povezuje u jednu funkcionalnu cjelinu.

Implementacija je fokusirana oko dvije glavne apstrakcije; registri mikroupravljača - realizirani s apstraktnim razredom Registar8_Base koji implementira 8-bitne registre i ALU, te instrukcije mikroupravljača - realizirani s razredom "Instruction" te razredima koji nasljeđuju ovaj razred, a koji implementiraju instrukcije i upravljačku jedinicu. Iako su

registri i aritmetičko-logička jedinica (engl. *Arithmetic Logic Unit*) unutar arhitekture mikroupravljača jasno odvojeni konstrukti, kao također i instrukcije i upravljačka jedinica, implementirani simulator neće simulirati hardversku arhitekturu mikroupravljača, već je izrađen na višem stupnju apstrakcije, tj. na programskom modelu mikroupravljača (engl. *Instruction Set Architecture - ISA*) te se u obzir uzimaju samo dvije važne apstrakcije - instrukcija i registar.

Radna memorija mikroupravljača implementirana je kao polje tog razreda. Iako se u tu svrhu moglo koristiti polje nekog primitivnog podatkovnog tipa, poput tipa int ili byte, apstraktni razred koristi se da bi se mogla prema potrebi pozvati dodatna logika prilikom operacije s tim vrijednostima. Razlog tome je što se za određene operacije moraju postavljati razne zastavice mikroupravljača, a također i što zadani mikroupravljač ima nekoliko vrsta registara, koji instrukcije tretiraju potpuno identično, iako su oni interno potpuno različiti. Iz tog razloga potrebno je nekoliko razreda koji imaju zajedničko sučelje, ali djelomično različitu implementaciju. Takvim dizajnom omogućeno je da razred "Instruction" i naslijeđeni razredi ne moraju pratiti kakvom vrstom registra manipuliraju te da li moraju pokrenuti kakve dodatne radnje, nego sve registre tretiraju na potpuno jednak način, dok se implementacija tih registara brine o dodatnim radnjama, baš kao i u fizičkoj implementaciji mikroupravljača.

Razred "Instruction" je bazni razred za četiri dodatna razreda, modelirana prema četiri binarna formata instrukcija mikroupravljača iz familije PIC16. Razred "Instruction" ima implementiran samo "kustur" razreda, osnovne metode, dok je u naslijeđenim razredima implementirana logika instrukcija. Da bi ovi razredi mogli pristupiti registrima koristi se sučelje "InternalCpuInterface" koje daje pristup svim registrima, stogu i potrebnim informacijama za uspješno izvršavanje svih instrukcija.

Kako bi razredi na bazi "Instrukcija" mogli izvršiti sve zadaće koje mogu izvršitit mikroupravljači PIC16, potrebna su još dva dodatna razreda - "RegisterPC" i "StackMemory". Razred "RegisterPC" predstavlja 13-bitni programski brojač mikroupravljača, koji je djelomično memorijski mapiran. Razred "StackMemory" predstavlja hardversku stog memoriju koju mikroupravljač koristi za proceduralne pozive i prekide.

Razred "Cpu" je kompozicija svih objekata potrebnih za simulaciju mikroupravljača. To uključuje razrede koji simuliraju programsku memoriju, radnu memoriju, hardverski stog, module poput brojača i prekidni sustav te njima pripadajuće registre posebne namjene. Ovaj razred implementira dva sučelja, jedno koje razred "Instruction" koristi za pristup svim podacima i metodama potrebnim za izvršavanje instrukcija mikroupravljača, dok drugo

sučelje koriste razredi grafičkog sučelja za dohvat svih podataka koje grafičko sučelje prikazuje.

Programska memorija je implementirana kao polje primjeraka razreda "Instruction". Razred također stvara primjerak razreda "StackMemory" za simulaciju hardverskog stoga, "RegistarFile" za simulaciju radne memorije te razrede InterruptController, TimerController i EepromController za simulaciju modula mikroupravljača.

Pozivanjem metode "executeInstruction" izvršava se jedna instrukcija. Ova metoda ažurira module mikroupravljača, provjerava da li je došlo do prekida te izvršava instrukciju koja se nalazi na adresi na koju pokazuje programsko brojilo.

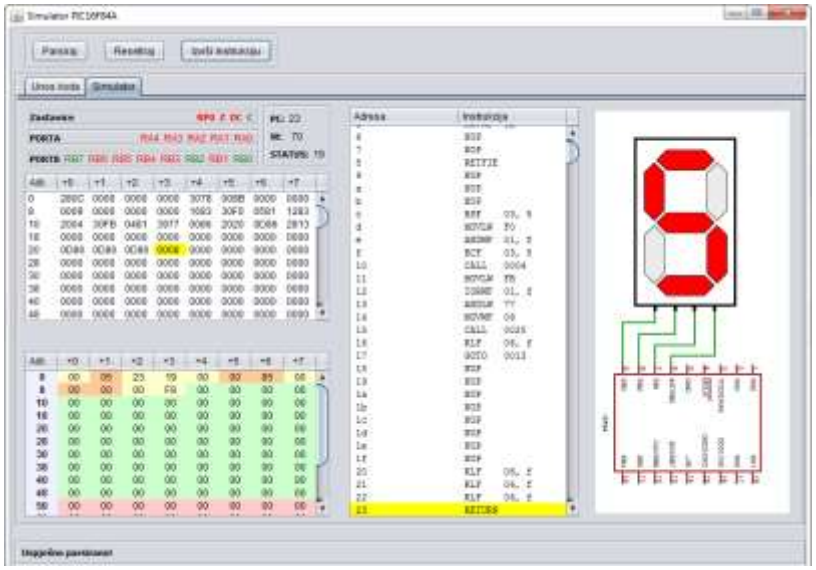
Radna memorija mikroupravljača simulirana je s razredom "RegistarFile". Razred ima jednu javnu metodu pomoću koje se dohvaća registar sa zadane adrese, pri čemu automatski provjerava s koje banke treba dohvatiti registar te odrađuje funkcionalnost indirektnog adresiranja. Uz jednu javnu metodu, primjerci važnih registara posebne namjene dostupni su pomoću javnih varijabli poput radnog registra W, programskog brojila, registra STATUS i ostalih važnih registara. Javne varijable su konstantnog tipa te nije moguće promijeniti primjerak razreda na koji se varijabla referencira. Implementiran je pomoću dva polja primjeraka razreda Register8b_Base, svako polje predstavlja jednu banku memorije. U polja se stvaraju svi potrebni primjerci razreda kao što su registri opće namjene i registri posebne namjene, dok se fizički neimplementirani adresni prostor popunjava s posebnim razredom koji implementira ponašanje tog prostora. Konstruktor ovog razreda prima primjerke razreda svih modula budući da su potrebne za stvaranje registara posebne namjene.

Razred "StackMemory" implementira stog memorijsku strukturu. Struktura prima 13-bitne vrijednosti budući da služi isključivo za pamćenje memorijskih adresa iz programskog brojila. Ovaj razred je nužan za simuliranje instrukcija "call" i "ret" te mehanizma prekida. Memorija je implementirana kao polje varijabli int tipa te s jednom int varijablom koja se koristi kao kružni pokazivač na stog.

Razred Register8b_Base simulira rad registra i aritmetičko-logičke jedinice. Implementiran je kao apstraktni razred s djelomičnom implementacijom, stoga nije moguće stvoriti primjerak ovog razreda, već on služi kao baza za razrede koji nasljeđuju ovaj razred i u potpunosti implementiraju metode. Sve aritmetičke i logičke operacije implementirane su u ovom razredu, ali metode za dohvaćanje i postavljanje same vrijednosti tog registra nisu implementirane, već su apstraktne. Ovim se na jednostavan način postiže da registri svih vrsta imaju zajedničko sučelje te ih instrukcije tretiraju na jednak način.

Postoji nekoliko vrsta registara u adresnom prostoru PIC16F84: standardni koji služe za memoriranje općih podataka, neimplementirani - memorijske adrese koje instrukcije mogu adresirati, ali se na njima ne nalaze fizički implementirani registri i memorijski - mapirani registri modula poput brojila i prekidnog sustava. Svi navedeni registri implementirani su nasljeđivanjem i implementiranjem ovog razreda.

Slika 1. Grafičko sučelje simulatora



Logika upravljačke jedinice implementirana je u ovom razredu. Svaki primjerak ovog razreda predstavlja jednu instrukciju u programskoj memoriji mikroupravljača. Četiri dodatna razreda nasljeđuju ovaj razred, svaki je odgovoran za određeni format instrukcije. Razredi prilikom stvaranja primjeraka primaju strojni kod instrukcije te ju u konstruktoru dekodira na tip instrukcije i operande prema određenom formatu.

Razredi imaju nekoliko javnih metoda, uključujući metodu "execute" koja izvršava logiku procesorske instrukcije. Metoda "execute" je implementirana na način da sve potrebne informacije i objekte dohvaća pomoću sučelja "CpuInternalInterface". Nakon što dohvati potrebne registre, npr. registar čija je adresa ukodirana u instrukciju, radni registar W, programsko brojilo i sl., metoda uspoređuje tip instrukcije i na temelju vrste odlučuje koje metode poziva nad dohvaćenim registrima. Također, ukoliko je potrebno radi operacije nad stog memorijom ili prekidnim sustavom.

Iako programska implementacija simulatora sadrži mnogo više razreda od spomenutih i opisanih u ovom radu, detaljan opis svih razreda i sučelja nadilazi opseg ovog rada. U ovom radu opisani su samo razredi nužni za simuliranje izvršavanja osnovnih instrukcija, dok

kompletna implementacija simulatora, parsera i grafičkog sučelja (slika 1) sadrži 37 razreda i 2 sučelja.

5. Zaključak

Arhitektura mikroupravljača PIC16 tvrtke Microchip može se ubrojiti u manje složene arhitekture mikroprocesora te je stoga ona dobro polazište za usvajanje prvih znanja u području ostvarivanja programske podrške za ugrađene računalne sustave. Nakana ovog rada je prikazati napor prema ostvarivanju jezgre simulatora navedene arhitekture. Implementirani simulator je moguće koristiti u daljnjem razvoju sustava za potporu procesa učenja u domeni arhitekture računala te ugrađenih računalnih sustava.

Literatura

1. Examining The First Microprocessor Chip Devices.
2. General-Purpose Microcontroller Family is Announced.
3. Getting Started with On-chip Memory.
4. Heffernan, D. 8051 TUTORIAL. http://galia.fc.uaslp.mx/~cantocar/microprocesadores/EL_Z80_PDF_S/8051.PDF (14.01.2015)
5. http://smithsonianchips.si.edu/augarten/p38.htm (18.10.2014)
6. http://ww1.microchip.com/downloads/en/devicedoc/33023a.pdf (10.12.2014)
7. http://ww1.microchip.com/downloads/en/DeviceDoc/ramrom.pdf (14.12.2014)
8. http://www.computerhistory.org/semiconductor/timeline/1974-MCU.html (18.10.2014)
9. http://www.ukessays.com/essays/information-technology/examining-the-first-microprocessor-chip-devices-information-technology-essay.php (18.10.2014)
10. Microchip PIC16F84A Data Sheet.http://ww1.microchip.com/downloads/en/devicedoc/35007b.pdf (25.11.2014)
11. PICmicro™ Mid-Range MCU Family Reference Manual.
12. Texas Instruments. (1976). TMS 1000 Series Data Manual.
13. The Most Widely Used Computer on a Chip The TMS 1000.