PREVAZILAŽENJE OGRANIČENJA SITEMA ZA RAZVOJ PREVODILACA PRILIKOM RAZVOJA PREVODIOCA ZA APX PROCESOR DIGITALNIH SIGNALA

Zoran Jovanović, *Fakultet Tehničkih Nauka u Novom Sadu*, Aleksandar Simeonov, Dejan Mišković, Miroslav Živković, *MicronasNIT, Novi Sad*

Sadržaj – *U radu je opisanan način prevazilaženja ograničenja sistema za razvoj prevodioca prilikom razvoja C prevodioca za APX procesor digitalnih signala. Opisane optimizacije omogućavaju iskorišćenje nekih od specifičnosti ciljne platforme.*

1. UVOD

Procesori digitalnih signala (DSP) svoju primenu pre svega nalaze u audio i video aplikacijama, kao i u oblasti telekomunikacija. Oni se najčešće koriste kao platforma za realizaciju algoritama koji zahtevaju obimna numerička izračunavanja u realnom vremenu. U velikom broju slučajeva predstavljaju osnovu rešenja u oblastima potrošačke elektronike, što za sobom povlači stroge zahteve u vezi cene takvih uredjaja. Ukoliko se radi o prenosivim uređajima sa sopstvenim napajanjem potrošnja, koja je direktno povezana sa radnom učestanošću, predstavlja podjednako bitan faktor, s obzirom da od nje zavisi dužina autonomnog rada uređaja. Sve navedeno dovodi do toga da ovakve platforme, u donosu na alogoritme koji se na njima realizuju, imaju veoma ograničene resurse, kako u vezi količine memorije, tako i u vezi radne učestanosti. Tradicionalno, programi za ovakve platforme se pišu u asemblerskom jeziku, jer on omogućava programeru da do maksimuma iskoristi memorijske resurse i procesorsku snagu ciljne platforme. Ovakav pristup u razvoju programske podrške, pored navedenih prednosti, ima i značajne nedostatke koji se pre svega ogledaju u količini vremena potrebnog za razvoj i kasnije održavanje tako napisanih programa. Rešenje navedenih problema se ogleda u procesu razvoja programa koji podrazumeva upotrebu prevodilaca za više programske jezike, a zatim pisanje kritičnih delova koda u asembleru. Zahtevi koji se postavljaju pred prevodioce za procesore digitalnih signala, pre svega proizilaze iz ograničenih resursa ovih platformi i odnose se kako na broj generisanih instrukcija, tako i na broj ciklusa koji je neophodan za njihovo izvršavanje. Da bi se ovi zahtevi ispunili neophodno je da prevodilac generiše program čiji broj instrukcija neće biti značajno veći od broja instrukcija u ručno pisanom asemblerskom kodu, a jedan od preduslova da se to postigne je maksimalno iskorišćenje specifičnosti ciljne platforme. U radu je opisana realizacija dve optimizacione tehnike koje prevazilaze pojedina ograničenja korišćenog sistema za razvoj prevodilaca i time omogućavaju iskorišćenje određenih specifičnosti ciljne platforme.

2. OPIS CILJNE PLATFORME

Ciljna platforma za koju je razvijen prevodilac je APX (*Advanced Processor with eXpandable architecture*) procesor digitalnih signala razvijen od strane nemačke firme Micronas. Arhitektura procesora omogućava da se RISC delu koji čini osnovu, doda i određen broj jedinica specifične namene. U osnovnoj konfiguraciji pored RISC dela nalazi se i jedinica za rad sa brojevima u pokretnom zarezu. Ove dve jedinice imaju mogućnost paralelnog rada. Osnovne karakteristike RISC jedinice su:

- protočna struktura vidljiva za programera,
- celobrojna aritmetika,
- 32 akumulatora širine 32 bita, od čega je u jednom trenutku moguće pristupiti do 16 akumulatora
- ALU jedinica,
- 16 adresnih registara širine 32 bita,
- samouvećanje i samoumanjenje sadržaja adresnih registara,
- podrška za kružne bafere,
- postojanje većeg broja memorijskih banaka uz linearni adresni prostor,
- upravljač resursima koji omogućava korišćenje raspoloživih resursa procesora na optimalan način, odnosno alokaciju željenog broja registara uz njihovo eventualno smeštanje na stek korišćenjem samo jedne instrukcije,
- skup instrukcija sadrži podršku za uslovno izvršenje instrukcija,
- paralelene instrukcije za istovremeno izvršenje aritmetičkih operacija i instrukcija smeštanja rezultata u memoriju ili učitavannje operanada iz memorije.

Osnovne karakteristike jedinice za rad sa brojevima u pokretnom zarezu:

- osam akumulatora širine 32 bita namenjenih čuvanju brojeva u standardnom IEEE Std-754 formatu,
- osam akumulatora koji čine proširenje osnovnog skupa akumulatora i koriste se za predstavljanje brojeva u proširenoj preciznosti, pri čemu se koriste 3 dodatna bita za eksponent i 6 dodatnih bita za mantisu,
- ALU jedinica,
- MAC jedinica,
- aritmetičke instrukcije koje uzimaju oba operanda iz memorije,
- mogućnost paralelnog rada sa RISC jedinicom.

Osobine APX procesora koje su bitne za optimizacije opisane u ovom radu biće detaljnije opisane u poglavljima posvećenim tim optimizacijama.

3. OPIS SISTEMA SA RAZVOJ PREVODIOCA

Prilikom realizacije prevodioca za APX familliju procesora korišćen je sistem za razvoj prevodioca CoSy (Compiler System) holandske firme ACE.

Ovaj sistem omogućava brz razvoj C prevodioca za veoma širok skup ciljnih platformi, što je omogućeno visokim stepenom konfigurabilnosti sitema. Prevodilac razvijen uz pomoć ovog sistema sačinjavaju moduli (engines) koji obavljaju relativno nezavisne zadatke. Komunikacija između modula obavlja se preko zajedničkih struktura podataka. Osnovna struktura podataka koja sadrži sve informacije o izvornom programu koji se prevodi je međujezik srednjeg nivoa (Intermediate representation), CCMIR. Modul anc0 koji realizuje prednji kraj prevodioca formira međujezik, a zatim svi ostali moduli koriste ovu strukturu. Svi CoSy moduli mogu se podeliti u tri osnovne grupe:

- moduli za prevođenje složenih konstrukcija u jednostavnije (lowering engines) – uloga ovih modula je da konstrukcije koje odslikavaju koncepte programskog jezika prilagode arhitekturi ciljne platforme,
- analizatorski moduli – proširuju međujezik informacijama neophodnim za rad drugih modula prevodioca,
- optimizacioni moduli – obavljaju transformacije međujezika u cilju generisanja boljeg izlaznog koda

Osim međujezika srednjeg nivoa postoji i međujezik niskog nivoa (Low Level Intermediate Representation). Ovaj međujezik se formira nakon faze izbora instrukcija i najčešće se koristi pri realizaciji mašinski zavisnih optimizacija i za naknadno poboljšanje generisanog koda. Skup instrukcija ciljne platforme opisan je pravilima koja se sastoje od šablona koji opisuje instrukciju i klauzula za definisanje uslova pod kojima pravilo može biti primenjeno, cene primene pravila koja se uzima u obzir prilikom izbora instrukcija, kao i drugih specifičnosti. Rezultat primene nekog od pravila je kreiranje odgovarajućeg čvora u međujeziku niskog nivoa.

4. KOMBINOVANJE VIŠE OD DVA ČVORA MEĐUJEZIKA NISKOG NIVOA

Jedan od mehanizama za poboljšanje generisanog koda, koje obezbeđuje sistem za razvoj prevodilaca CoSy, je i kombinovanje dva čvora međujezika niskog nivoa u cilju spajanja dve operacije u jednu složeniju u slučaju da skup instrukcija to dozvoljava. Tipičan primer ovakve transformacije je iskorišćenje mogućnosti samouvećanja sadržaja adresnih registara prilikom pristupa memoriji. Na slici 1.a dat je primer generisanog koda pre primene opisane optimizacije, a na slici 1.b je prikazan kod nakon optimizacije.

```
ZR(0) = YM[P0 + 0]
ZR(STATICZERO) = YM[P0 += 1]
        a.
ZR(0) = YM[P0 (+= 1)]
        b.
```

Slika1. *a) pre primene optimizacije b) nakon primene optimizacije*

Kao što se može videti sa slike 1.a prvo se generiše instrukcija koja čita sadržaj memorijske lokacije, a zatim instrukcija koja uvećava sadržaj adresnog registra čitanjem sadržaja memorijske lokacije u registar u koji nije moguće pisati (STATICZERO), samo da bi se promenila vrednost adresnog registra. Na slici 1.b prikazana je instrukcija emitovana nakon spajanja čvorova za čitanje sadržaja memorijske lokacije i instrukcije uvećanja sadržaja adresnog registra. Na nivou međujezika ne događa se fizičko spajanje, nego se samo dodaje veza između dva kombinovana čvora.
Kao što je već rečeno, APX procesor poseduje aritmetičke instrukcije koje uzimaju oba operanda direktno iz memorije. Primer aritmetičke instrukcije sa dva memorijska operanda prikazan je na slici 2. U primeru je data instrukcija sabiranja sadržaja dve memorijske lokacije uz uvećanje sadržaja adresnog registra P0 i umanjenje sadržaja adresnog registra P8.

```
CF1 = (AF0 = ZM[P0 + = 1 ]) +  (AF0 = ZM[P8 - = 1 ])
```

Slika 2. *Primer instrukcije sa dva memorijska operanda*

Prepreku u generisanju ovakve instrukcije od strane prevodioca predstavlja činjenica da CoSy omogućuje spajanje samo dva čvora međujezika niskog nivoa, što dovodi do toga da je moguće modifikovati sadržaj samo jednog od adresnih registara. Na slici 3 prikazan je generisan kod pre primene optimizacije.

```
ZR(STATICZERO) = YM[P0 += 1]
ZR(STATICZERO) = YM[P8 -= 1]
CF1 = (AF0 = ZM[P0 + 0 ]) +  (AF0 = ZM[P8  + 0])
```

Slika 3. *Kod pre primene optimizacije spajanja čvorova međujezika niskog nivoa*

Primenom optimizacije podržane od strane sistema za razvoj prevodioca dobija se kod sa slike 4.

```
ZR(STATICZERO) = YM[P8 -= 1]
CF1 = (AF0 = ZM[P0 + 0 ]) +  (AF0 = ZM[P8  + 0])
```

Slika 4. *Kod nakon primene standardne optimizacije spajanja čvorova međujezika niskog nivoa*

U standardnoj konfiguraciji prevodioca modul za kombinovanje, combine se poziva samo jednom i svaki čvor može biti kombinovan samo sa jednim čvorom međujezika niskog nivoa. U cilju prevazilaženja ovog ograničenja razvijen je nov modul nazvan coalescecombine. Uloga ovog

modula je da dva čvora koja su kombinovana nakon prvog poziva combine modula zameni jednim novim, koji zamenjuje kompletnu funkcionalnost originalnih čvorova. Osim novog modula svakom od pravila koje može postati predmet optimizacije je dodat korisnički prolaz (user pass) unutar koga se izvršava akcija formiranja novog čvora, kao i nova pravila na osnovu kojih se formiraju čvorovi koji zamenjuju funkcionalnost dva kombinovana čvora.
Primeri originalnih pravila dati su na slikama 4.a i 4.b, dok je novo pravilo koje ih zamenjuje dato na slici 4.c. Sa slike se može videti da je modifikacija sadržaja adresnog registra predstavljena uvođenjem novog operatora xirContentIncr.

```
o:mirPlus(o1:mirContent(a1:adr), o2:mirContent(a2:adr))
-> rd:Nt_r32;
                    (a)

o:mirAddrPlus (s:adr, c:mirIntConst) -> d:adr;
                    (b)

mirAddRULE  o:mirPlus (o1:xirContentIncr(a1:adr),
o2:mirContent(a2:adr)) -> rd:Nt_r32;
                    (c)
```

Slika 5. *Primeri originalnih kombinovanih pravila a) i b) i pravilo koje ih zamenjuje c)*

Nakon zamene dva originalna čvora, od strane coalescecombine modula, sa jednim novim ponovo se poziva standardni modul za kombinovanje čvorova čime se otvara mogućnost spajanja novouvedenog sa nekim od postojećih čvorova. Konačan rezultat rada navedenih modula je oblik instrukcije prikazan na slici 2, čime je postignuto maksimalno iskorišćenje mogućnosti koje pruža skup instrukcija APX procesora.

5. SPREČAVANJE TRANSFORMACIJE ADRESNIH IZRAZA

Adresni generatori APX procesora podržavaju i indeksno adresiranje. Primer instrukcije čitanja sadržaja memorijskih lokacije sa indeksnim adresiranjem dat je na slici 6.

```
ZM[R8 + 5] = YR(R6)
ZM[R8 + 6] = YR(R1)
ZM[R8 + 7] = YR(R1)
```

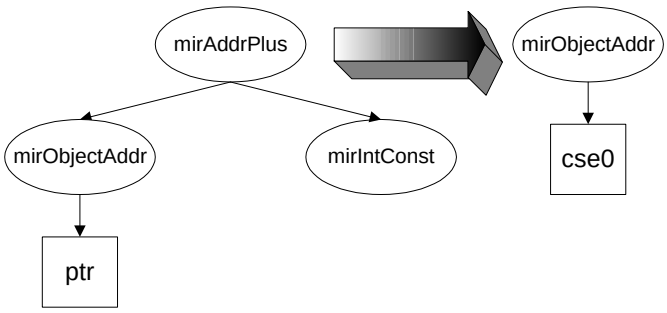
Slika 6. *Primer instrukcije sa indeksnim adresiranjem*

Jedan od modula koji predstavljaju deo sistema za razvoj prevodilaca je i modul za eliminaciju zajedničkih podizraza.
Problem koji se javlja prilikom upotrebe ovog modula je da njegovom primenom dolazi do optimizacije adresnih izraza. Ova optimizacija nije poželjna s obzirom na postojanje indeksnog adresnog moda i činjenicu da se na taj način povećava broj generisanih instrukcija. Primer koda generisanog nakon transformacije međujezika od strane modula za eliminaciju zajedničkih podizraza dat je na slici 7.

```
YC0 = XC8 + 20
ZM[R0 + 0] = YR(R6)
ZM[R0 + 1] = YR(R1)
ZM[R0 + 2] = YR(R1)
```

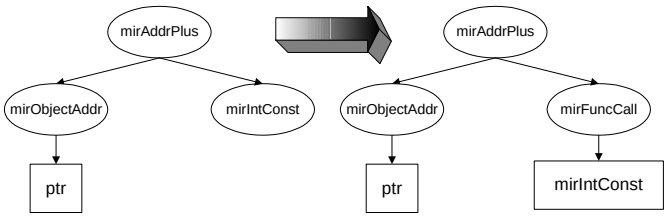
Slika 7. *Primer generisanog koda nakon transformacije od strane modula za eliminaciju zajedničkih podizraza*

Izgled međujezika srednjeg nivoa koji je predmet optimizacije prikazan je na slici 8. Sa slike se može videti da je u ovom slučaju emitovana nepotrebna instrukcija kojom se inicijalizuje registar C0, a zatim za pristup memoriji koriste indeksi 0, 1 i 2, umesto da se direktno koriste odgovarajući indeksi. Prilikom rada modula za eliminaciju zajedničkih podizraza celokupno podstablo sa korenom u čvoru mirAddrPlus zamenjuje se pristupom privremenoj promenljivoj uvedenoj od strane tog modula, u ovom slučaju cse0.



Slika 8. *Transformacija adresnog izraza od strane modula za eliminaciju zajedničkih podizraza*

Osnovna ideja u sprečavanju ovih transformacija je da se pre poziva modula za eliminaciju zajedničkih podizraza svi čvorovi celobrojnih konstanti unutar adresnih izraza zamene čvorovima poziva funkcija, a da se po završetku rada optimizacionog modula izvrši inverzna transformacija. Na slici 9 dat je izgled dela stabla međujezika pre (a) i posle transformacije (b). Vrednost celobrojne konstante čuva se kao parametar poziva funkcije što omogućava inverznu transformaciju nakon završetka rada modula za eliminaciju zajedničkih podizraza.



(a) (b)

Slika 9. *Transformacija adresnog izraza u cilju sprečavanja optimizacije od strane modula za eliminaciju zajedničkih podizraza*

6. ZAKLJUČAK

U radu su prikazane dve transformacije koje prevazilaze ograničenja, odnosno sprečavaju neželjene optimizacije sistema za razvoj prevodilaca CoSy. Navedene optimizacije u značajnoj meri doprinose iskorišćenju specifičnosti APX

procesora, što je od velikog značaja pri realizaciji aplikacija za digitalne signalne procesore.

LITERATURA

[1] ACE Associated Compiler Experts bv. *Beg-CoSy Reference Manual.* Ref. CoSy-8005-beg, 2001.

[2] ACE Associated Compiler Experts bv. *The CoSy Framework*, *A compiler construction system*. Ref. CoSy-8006-fw, 2001.

[3] Steven S. Muchnick, "*Advanced Compiler design & Implementation*", Morgan Kaufmann Publishers, 1997

[4] Aho, Sethi, Ullman, "*Compilers Principles, Techniques, and Tools*", Addison Wesley, 1985

Abstract – This paper describes the way to overcome restrictions of compiler development system during the implementation of C compiler for APX digital signal processor. The described optimizations enable usage of some specific features of target platform.

RESTRICTION OVERCOMING OF COMPILER DEVELOPMENT SYSTEM FOR THE DEVELOPMENT OF APX DIGITAL SIGNAL PROCESSOR COMPILER

Zoran Jovanovic, Aleksandar Simeonov, Dejan Miskovic, Miroslav Zivkovic