Prilagođavanje uvezivača GOLD za arhitekturu MIPS64

Vladimir Radosavljević, Miroslav Popović, Petar Jovanović, Strahinja Petrović

Uvezivač predstavlja alat koji spaja objektne datoteke i od njih pravi izvršnu ili deljenu datoteku. Cilj ovog rada je realizacija podrške za uvezivanje programa za arhitekturu MIPS64 u okviru uvezivača GOLD. U radu je realizovana podrška za verzije binarne sprege programske podrške N32 i N64.

***Ključne reči*—Uvezivač, GOLD, MIPS64**

I. UVOD

Uvezivači (*eng. linkers*) i punjači (*eng. loaders*) postoje kao deo programskih alata skoro onoliko dugo koliko postoje računari, jer predstavljaju bitne komponente programskih alata koje omogućavaju da se programi prevode iz više a ne samo iz jednog modula.

Uvezivač je alat koji spaja objektne datoteke dobijene prevođenjem ili asembliranjem, i od njih pravi izvršnu ili deljenu datoteku. Rukuje sa malim brojem osnovnih tipova podataka koji su definisani u ulaznim objektnim datotekama: simboli, relokacije i podaci.

Simbol je u osnovi ime i vrednost (adresa u memoriji). U objektnim datotekama koji su generisani od koda koji je napisan u programskom jeziku C, postoje simboli za svaku funkciju i za svaku globalnu ili statičku promenljivu. Takvi simboli se nazivaju definisani simboli. Simboli se takođe koriste da označe reference na ime nekog simbola koji je definisan u nekoj drugoj objektnoj datoteci. Takve reference se nazivaju i nedefinisani simboli. Tokom procesa uvezivanja, uvezivač dodeljuje adrese za svaki definisan simbol, i razrešava svaki nedefinisan simbol tražeći definisan simbol sa istim imenom.

Relociranje je proces koji se izvršava nad podacima. Većina relokacija se odnosi na simbole kao i na odstojanja u okviru podataka. Vrste računanja koje relokacije vrše zavise direktno od arhitekture. Relokacije u objektnoj datoteci mogu da se odnose na nedefinisane simbole. Ako uvezivač ne može da razreši taj simbol, on će u većini situacija prijaviti grešku (postoje simboli za koje se ne prijavljuje greška ako nisu definisani).

Podaci predstavljaju niz bajtova koji imaju svoju veličinu i tip. Oni sadrže mašinski kod generisan od strane prevodioca i asemblera (*text* sekcija). Takođe sadrže i inicijalizovane promenljive (*data* sekcija). Mogu sadržati i konstantne promenljive i konstantne tekstove (*read-only* sekcija). U njima se nalaze i neinicijalizovane promenljive za koje se podrazumeva vrednost nula (*bss* sekcija). Uvezivač čita podatke iz svake ulazne datoteke, spaja ih zajedno, primenjuje relokacije i upisuje ih u izlaznu datoteku.

Osnovni koraci kroz koje svaki uvezivač prolazi:

- Čitanje ulaznih objektnih datoteka
- Određivanje veličine i tipova podataka koje se nalaze u ulaznim datotekama
- Čitanje simbola iz ulaznih datoteka
- Konstruisanje tabele simbola koja sadrži sve simbole iz ulaznih objektnih datoteka i uvezivanje nedefinisanih simbola sa njihovim definicijama
- Određivanje gde će se podaci nalaziti u izlaznoj datoteci, tj. određivanje gde će se oni nalaziti kada se program pokrene
- Čitanje sadržaja podataka i relokacija
- Primeniti relokacije na podatke u izlaznoj datoteci
- Po potrebi, upsati kompletnu tabelu simbola sa njihovim vrednostima u izlaznu datoteku.

II. UVEZIVAČ GOLD

Razvoj uvezivača GOLD[1] započet je od strane Jan Lens Tejlora i malog tima u kompaniji Gugl. Ideja je bila da se realizuje uvezivač koji će biti brži od već postojećeg uvezivača GNU (*eng. GNU LD*). U programske alate GNU, uvezivač GOLD je dodat 2008. godine.

Uvezivač GOLD je napisan u programskom jeziku C++ i podržava samo format datoteka ELF (*eng. Executable and Linkable Format*). Glavni razlog zašto je uvezivač GOLD napisan u programskom jeziku C++ je mogućnost korišćenja šablona (*eng. template*) kao i jednostavnije korišćenje različitih tipova struktura podataka.

Osnovni koraci kroz koje GOLD prolazi su:

- Prolazak kroz ulazne datoteke radi identifikacije simbola i ulaznih sekcija
- Ponovni prolazak kroz ulazne datoteke, radi čitanja relokacija i pravljenja tabele globalnih odstojanja (*eng. Global Offset Table*) i tabele proceduralnih uvezivanja (*eng. Procedure Linkage Table*)
- Dodeljivanje izlaznih sekcija izlaznim segmentima i dodeljivanje adresa izlaznim segmentima
- Prolazak kroz ulazne datoteke, radi prepisivanja ulaznih sekcija u izlaznu datoteku i radi razrešavanja relokacija

Glavne razlike u odnosu na uvezivač GNU jesu u drugom

Vladimir Radosavljević - Istraživačko-razvojni institut RT-RK, Narodnog fronta 23a, 21000 Novi Sad, Srbija (e-mail: vladimir.radosavljevic@rt-rk.com)

Miroslav Popović - Istraživačko-razvojni institut RT-RK, Narodnog fronta 23a, 21000 Novi Sad, Srbija (e-mail: miroslav.popovic@rt-rk.com)

Petar Jovanović - Istraživačko-razvojni institut RT-RK, Narodnog fronta 23a, 21000 Novi Sad, Srbija (e-mail: petar.jovanovic@rt-rk.com)

Strahinja Petrović - Istraživačko-razvojni institut RT-RK, Narodnog fronta 23a, 21000 Novi Sad, Srbija (e-mail: strahinja.petrovic@rt-rk.com)

prolazu u kojem se čitaju relokacije kao i izostanak podrazumevane skripte koju koriste uvezivači (*eng. Linker Script*). Drugim prolazom u kojem se čitaju relokacije, pojednostavljuje se rukovanje sa simbolima i izbegava se prolazak kroz tabelu simbola. Izostanak podrazumevane skripte koju koriste uvezivači omogućava da se ulazne sekcije dodeljuju izlaznim sekcijama prilikom njihovog čitanja. Time se izlazne sekcije lako mogu grupisati u izlazne segmente na osnovu njihovog tipa ili indikatora (samo za čitanje, upisive i/ili izrvršne), a ne prema adresama koje su dodeljene u skriptama koje koriste uvezivači. Uvezivač GOLD takođe ima podršku za skripte koje koriste uvezivači iz kompatibilnih razloga.

Ispitivanjem performansi[1], utvrđeno je da je uvezivač GOLD od dva do pet puta brži od uvezivača GNU.

III. ARHITEKTURA MIPS64

Arhitektura MIPS64[3] se koristi u igračkim konzolama, u set-top boksovima, a danas održava popularnost u umrežavanju i u telekomunikacionim infrastrukturnim programima. Uključivanjem moćnih funkcionalnosti, arhitektura MIPS64 pruža solidnu osnovu visokih performansi za budućnost razvijanja arhitekture MIPS.

Arhitekture MIPS32 i MIPS64 uključuju važne tehnologije kao što su vektorska obrada podataka (*eng. Single Instruction Multiple Data*) i virtualizacija. Ove tehnologije, u saradnji sa tehnologijama kao što su paralelizacija (*eng. multi-threading*), ekstenzije na procesorima za digitalnu obradu signala i EVA (*eng. Enhanced Virtual Addressing*), obogaćuju arhitekturu da podržava savremene programe koji zahtevaju veće količine memorije, povećane računarske operacije i okruženja za bezbedno izvršavanje programa.

Arhitektura MIPS64 se zasniva na fiksnim širinama instrukcijske reči. Aritmetičke i logičke operacije koriste format sa tri operanda (*eng. three-operand*), omogućavajući prevodiocima da optimizuju složene izraze. Dostupnost trideset i dva opšte namenska registra, omogućava prevodiocu bolju optimizaciju koda, držeći podatke kojima se često pristupa u registrima.

Postoje dve binarne sprege programske podrške na arhitekturi MIPS64: N64 i N32 ABI (*eng. Application binary interface*). N64 ABI se koristi u 64-bitnom režimu adresiranja, dok radi sa 64-bitnim podacima. Kako bi se omogućio lak prelazak na 32-bitnu familiju, arhitektura MIPS64 podržava 32-bitni kompatibilni režim (N32 ABI) u kojoj su svi registri i adrese širine 32 bita i sve instrukcije koje su prisutne na arhitekturi MIPS32, mogu da se izvršavaju.

IV. REALIZACIJA REŠENJA

Kako bi se dodala podrška unutar uvezivača GOLD za uvezivanje programa za arhitekturu MIPS64 bilo je neophodno proći kroz nekoliko koraka.

A. *Podrška za sekciju .MIPS.options*

U ovoj sekciji se nalaze razne informacije koje se odnose na objektnu datoteku. Uvezivač iz ove sekcije koristi informacije o upotrebi registara, koje se nalaze u deskriptoru ODK_REGINFO.

Polja koje se nalaze u deskriptoru ODK_REGINFO su:

- Maska registara opšte namene koji se koriste
- Maska koprocesorskih registara koji se koriste
- Inicijalna vrednost globalnog pokazivača

Za ovu sekciju je bilo potrebno dodati podršku za čitanje informacija iz ulaznih datoteka, kao i ažuriranje inicijalne vrednosti globalnog pokazivača u svim deskriptorima ODK_REGINFO koje se nalaze u izlaznoj datoteci.

B. *Podrška za sekciju .MIPS.abiflags*

U zaglavlju ELF se nalazi polje *e_flags* u kojem se smeštaju informacije o specifičnostima nekog procesora. Zbog nedovoljne veličine tog polja, kao i zbog razloga što to polje nije dostupno posle učitavanja programa, bilo je potrebno napraviti sekciju u kojoj će se čuvati sve neophodne informacije koje omogućavaju programskom punjaču da odredi zahteve programa.

Informacije koje se nalaze u ovoj sekciji su:

- Verzija strukture radi budućih ekstenzija (inicijalna vrednost je 0)
- Maksimalna veličina registara potrebna za svaku klasu registara
- Binarna sprega programske podrške za jedinicu za operacije u aritmetici pokretnog zareza (*eng. Floating point ABI*)
- Ekstenzije instrukcijskog seta
- Verzija instrukcijskog seta
- Indikator da li se koriste neparni registri jednostruke preciznosti

Za ovu sekciju je bilo potrebno dodati podršku za čitanje informacija iz ulaznih datoteka, spajanje informacija iz ulaznih datoteka na osnovu unapred određenih pravila i upisivanje spojenih informacija u izlaznu datoteku.

C. *Podrška za čitanje informacija iz relokacionih sekcija*

Relokaciona sekcija sadrži sve neophodne informacije koje opisuju kako se modifikuje sadržaj neke sekcije, omogućavajući time da izvršne i deljene datoteke mogu pravilno da se izvršavaju.

N64 i N32 ABI koriste tip relokacione sekcije SHT_RELA. Za svaku relokaciju je potrebno pročitati iz strukture na kom mestu se relokacija primenjuje (r_offset), indeks simbola iz tabele simbola (r_sym) i tip relokacije (r_type) koji se nalaze u r_info polju kao i dodatnu informaciju koja je potrebna da bi rezultat relokacije bio tačan (r_addend).

N64 ABI ima specifično polje r_info koje sadrži 3 tipa relokacija (r_type, r_type2, r_type3), specijalan simbol (r_ssym) i indeks simbola iz tabele simbola (r_sym). Polje r_info na arhitekturi MIPS64 N64 ABI koja radi u režimu little endian (sa manjeg kraja), umesto jednog 64-bitnog broja little endian, sadrži jedan 32-bitni broj little endian praćen sa 32-bitnim brojem big endian (sa većeg kraja).

Field Name	Type	Comments
r_offset	**Elf64_Addr**	Where to apply relocation: relocatable: byte offset in section executable: virtual address
r_sym	**Elf64_Word**	Symbol index
r_ssym	**Elf64_Byte**	Special symbol — see Table 30
r_type3	**Elf64_Byte**	Relocation type — see Table 32
r_type2	**Elf64_Byte**	Relocation type — see Table 32
r_type	**Elf64_Byte**	Relocation type — see Table 32
r_addend	**Elf64_Sxword**	Explicit addend for relocation operation (*Elf64_Rela* only)

Sl. 1. Izgled strukture relokacija za N64 ABI[3]

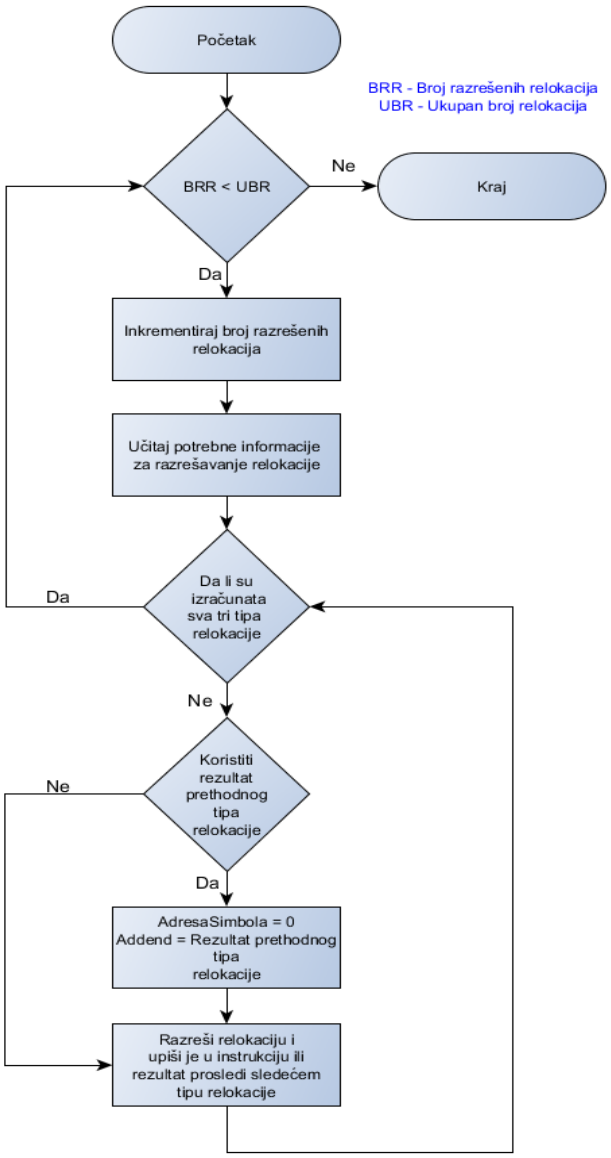
Iz navedenih razloga, bilo je potrebno dodati podršku za čitanje informacija iz relokacionih sekcija i to samo za N64 ABI.

D. Dodavanje algoritma za razrešavanje relokacija

Razrešavanje relokacija je proces povezivanja referenci na simbole sa definicijom tih simbola.

Na početku razrešavanja relokacije, potrebno je za svaku učitati osnovne informacije na osnovu kojih se izvršava relociranje.

N64 ABI je specifičan u razrešavanju relokacija iz razloga što se u jednoj relokaciji nalaze tri tipa relokacija koje se razrešavaju jedna za drugom. Kada se relokacija računa za prvi tip relokacije (r_type), računa se na osnovu informacija koje su dobijene iz strukture relokacija ili iz prethodno izračunate relokacije. Na osnovu toga da li je drugi tip relokacije (r_type2) R_MIPS_NONE odlučuje se da li se rezultat relokacije upisuje u instrukciju ili se šalje za računanje sledećeg tipa relokacije. Ako je drugi tip relokacije R_MIPS_NONE, onda se rezultat računanja u prvom tipu relokacije upisuje u instrukciju i prelazi se na razrešavanje sledeće relokacije. U slučaju da nije, onda se prilikom računanja drugog tipa relokacije umesto dodatne informacije (r_addend) koristi rezultat računanja iz prvog tipa relokacije i adresa simbola se određuje na osnovu specijalnog simbola (r_ssym). Na kraju računanja drugog tipa relokacije, ako je potrebno, isto se dešava kao i kod računanja prvog tipa relokacije. Za razliku od prethodnih tipova relokacija, na kraju računanja trećeg tipa relokacije (r_type3), ako je potrebno, ispituje se da li se sledeća relokacija odnosi na istu instrukciju. Ako se odnosi, onda se rezultat posle trećeg tipa relokacije koristi kao dodatna informacija u prvom tipu relokacije u sledećoj relokaciji i adresa simbola se stavlja na nulu. Ovaj postupak se ponavlja za sve relokacije ako se odnose na istu instrukciju. Ako se sledeća relokacija ne odnosi na istu instrukciju, rezultat posle trećeg tipa relokacije se upisuje u instrukciju i računanje nove relokacije se radi na osnovu informacija iz strukture relokacija.



Sl. 2. Algoritam razrešavanja relokacija za N64 ABI

Za razliku od N64 ABI koji ima tri tipa relokacija, N32 ABI ima samo jedan tip relokacije. Na kraju računanja relokacije, ako se sledeća relokacija ne odnosi na istu instrukciju, rezultat relokacije se upisuje u instrukciju. U slučaju da se sledeća relokacija odnosi na istu instrukciju, onda se rezultat relokacije koristi kao dodatna informacija u sledećoj relokaciji i adresa simbola sledeće relokacije se postavlja na nulu i tako za sve relokacije dok se one odnose na istu instrukciju.

V. TESTIRANJE

Provera ispravnosti uvezivača GOLD koji je prilagođen arhitekturi MIPS64, vršena je uz pomoć dva skupa testova. Za potrebe verifikacije korišćena je platforma Cavium Octeon CNG / EP6300C sa procesorom MIPS64R2.

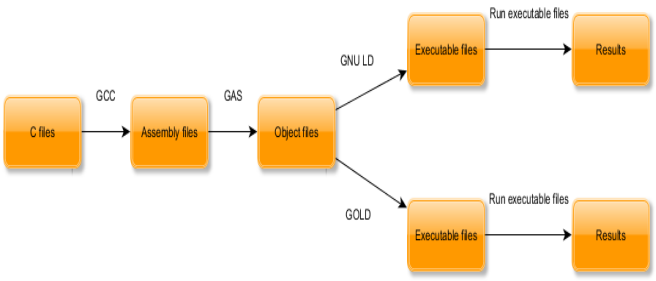
Takođe, uvezivač GOLD je korišćen za uvezivanje pretraživača Chromium i kernela Linux. U oba slučaja su

programi uspešno uvezani i uspešno pokrenuti na platformi sa procesorom MIPS64.

A. *Skup testova DejaGNU*

Iz ovog skupa testova, iskorišćeno je 1600 testova, koji se inače koriste za ispitivanje rada programskih prevodioca.

Testiranje je vršeno tako što se na početku uzima svih 1600 testova koji su napisani u programskom jeziku C i alatom GCC (*eng. GNU Compiler Collection*) se dobijaju asemblerske datoteke. Od tih asemblerskih datoteka su dalje pravljene objektne datoteke pomoću alata GAS (*eng. GNU Assembler*). Tako dobijene objektne datoteke su prosleđivane uvezivaču GNU i uvezivaču GOLD koje su od njih pravile izvršne datoteke. Upoređivanje rezultata je vršeno pokretanjem tih izvršnih datoteka, i na osnovu dobijenih rezultata utvrđeno je da se rezultati poklapaju.



Sl. 3. Postupak testiranja

B. *Skup testova prevodioca LLVM*

Ovaj skup testova sadrži kompletne programe koji se mogu prevesti, uvezati i izvršiti. Izvorni kod ovih testova je uglavnom pisan u programskim jezicima C i C++. Testovi se prvo prevode za arhitekturu MIPS64, zatim se uvezuju pomoću uvezivača GOLD, i na kraju se upoređuju rezultati izvršenih testova sa referentim rezultatima. Na osnovu dobijenih rezultata, utvrđeno je da se svi testovi uspešno izvršavaju.

VI. ZAKLJUČAK

U ovom radu realizovana je podrška za uvezivač GOLD koji omogućava uvezivanje programa za arhitekturu MIPS64 i to za N64 i N32 ABI. Podržavanje uvezivača GOLD za arhitekturu MIPS64 bilo je neophodno zato što se ovaj uvezivač koristi u značajnim projektima koji pripadaju zajednici otvorenog koda (*eng. Open source community*). Neki od tih projekata su: pretraživač Chromium, matični klijent (*eng. Native client*) i operativni sistem Android. Prikazano rešenje nije predlagano Binutils zajednici do sada. Rešenje je prihvaćeno i nalazi se u zvaničnom kodu uvezivača GOLD.

Dalji razvoj podrazumeva dodavanje podrške za postepeno uvezivanje (*eng. incremental linking*), kao i unapređenje algoritma za spajanje identičnih sekcija *(eng. identical code folding*) za arhitekturu MIPS64.

ZAHVALNICA

Ovaj rad je delimično finansiran od strane Ministarstva za nauku i tehnologiju Republike Srbije, na projektu tehnološkog razvoja broj TR32031.

LITERATURA

[1] Ian Lance Taylor, “A New ELF Linker”, Proceedings of 2008 the GCC Developers' Summit, 2008 (dostupno online na: https://research.google.com/pubs/pub34417.html, pristupljeno 28.04.2017.)

[2] J. R. Levine, "Linkers and loaders", 1st ed. San Francisco, Calif. [u.a.]: Morgan Kaufmann, 2010.

[3] Imagination technologies, MIPS64 architecture (dostupno online na: https://www.imgtec.com/mips/architectures/mips64/, pristupljeno 28.04.2017.)

[4] Jim Dehnert, "64-bit ELF Object File Specification", MIPS Technologies Silicon Graphics Computer Systems (dostupno online na: ftp://www.linux-mips.org/pub/linux/mips/doc/ABI/elf64-2.4.pdf, pristupljeno 28.04.2017.)

ABSTRACT

Linker is a computer program that combines object files into a single executable file or a shared library. This paper describes effort to port GOLD linker for the MIPS64 architecture for both N64 and N32 ABI.

Porting GOLD linker to MIPS64 architecture

Vladimir Radosavljević, Miroslav Popović, Petar Jovanović, Strahinja Petrović