RETARGETABLE SOURCE-LEVEL DEBUGGER

Pavol Korvas
Master Degree Programme (2), FIT BUT
E-mail: xkorva01@stud.fit.vutbr.cz

Supervised by: Jakub Křoustek
E-mail: ikroustek@fit.vutbr.cz

Abstract: This paper deals with the design of a retargetable source-level debugger. The paper describes functionalities of debugger such as stepping, inserting and processing breakpoints, visualization of information about memory, stack, and variables. Future research is discussed at the end of the paper.

Keywords: debugging, debugger, DWARF, debugging information, Lissom, stepping, breakpoint

1 ÚVOD

Táto práca sa zaoberá návrhom rekonfigurovateľného ladiaceho nástroja na úrovni zdrojového kódu. Takýto typ ladiaceho nástroja sa využíva pri súčasnom návrhu modelov procesorov a aplikácií pre tieto procesory (*hardware/software codesign*). V tejto oblasti je možné jednotlivé modely procesorov simulovať, čo nám umožní odladiť každý model procesoru bez nutnosti jeho fyzického vytvorenia. Rekonfigurovateľnosť nám zaistí fungovanie ladiaceho nástroja nezávisle na cieľovej architektúre, čím dosiahneme toho, že použitím jedného nástroja dokážeme odladiť veľké množstvo rôznych modelov procesorov a aplikácií [1].

2 PRINCÍP LADIACICH NÁSTROJOV

Ladenie môžeme definovať ako proces diagnostiky chýb nachádzajúcich sa v programe a určenie spôsobu ich opravy. Chyby sa objavujú v rôznych formách, napríklad chyby v kóde, chyby návrhu, chyby užívateľského rozhrania alebo tiež systémové chyby. Samotné ladenie je riadené pomocou krokovania a nastavovania rôznych typov bodov prerušenia (*breakpointov*). Ladiaci nástroj obsahuje aj ďalšie dôležité funkcionality ako zobrazovanie aktuálne ladeného kódu, získavanie informácií o premenných, vyhodnocovanie výrazov a vizualizáciu stavu zásobníka, pamäte a registrov [2].

3 NÁVRH REKONFIGUROVATEĽNÉHO LADIACEHO NÁSTROJA NA ÚROVNI ZDROJOVÉHO KÓDU

Tento ladiaci nástroj vzniká ako súčasť projektu Lissom a využíva nástroje vytvorené vrámci tohoto projektu. Ladiace informácie, ktoré sa využívajú pre zaistenie všetkých funkcií vyvíjaného ladiaceho nástroja, sa získavajú z formátu ladiacich informácii DWARF. Formát DWARF je spracovávaný pomocou knižnice *dwarfapil*. Ladiace informácie sú generované pomocou experimentálneho prekladača jazyka C. Oba tieto nástroje sú vyvíjané vrámci projektu Lissom.

3.1 RIADKOVÉ INFORMÁCIE ZDROJOVÉHO KÓDU

Najzákladnejšou funkcionalitou ladiaceho nástroja je možnosť krokovania, nastavovania bodov prerušenia a zobrazovania aktuálne ladeného kódu. Tieto všetky funkcionality sa týkajú riadkových informácií získavaných z ladiacich informácií.

Bod prerušenia je špeciálny kód vložený do programu, ktorý pri vykonávaní vyvolá výnimku a odovzdá riadenie ladiacemu nástroju s tým, že program sa zastaví. Body prerušenia môžu byť nastavované na úrovni zdrojového kódu pomocou špecifikácie súboru a čísla riadku alebo na úrovni strojových inštrukcií priamo zadaním adresy. Môžeme ich rozdeliť na logické, fyzické, dočasné, interné a dátové (*watchpointy*) [3].

Vyvíjaný ladiaci nástroj umožňuje nastavovať a zachytávať všetky zmienené body prerušenia. Taktiež umožňuje krokovanie na troch úrovniach a to na úrovni inštrukcií mikroarchitektúry, na úrovni strojových inštrukcii a na úrovni zdrojového kódu. Inštrukcia zdrojového kódu môže zahŕňať viacero strojových inštrukcií, preto sa vytvorilo mapovanie medzi týmito dvoma úrovňami inštrukcií. Toto mapovanie sa uskutočňuje pomocou adries jednotlivých inštrukcií. Inštrukcia zdrojového kódu určená rozsahom adries a všetky strojové inštrukcie spadajúce do tohoto rozsahu sa namapujú na danú inštrukciu zdrojového kódu.

Krokovanie využíva tri príkazy a to `step`, `next` (*step over*) a `finish` (*step return*). Pre zaistenie správnej funkcionality príkazov `next` a `finish` bolo nutné implementovať hierarchiu vykonávania funkcií a možnosť nastavovať a zachytávať interné body prerušenia. Pri oboch príkazoch je potrebné získať informáciu o návratovej adrese funkcie, ku ktorej sa tieto príkazy vzťahujú. Návratová adresa danej funkcie je získaná z informácií, ktoré poskytuje hierarchia volania funkcií. Konkrétna realizácia tejto hierarchie je popísaná v časti o vizualizácii stavu zásobníka.

3.2 Informácie o premenných

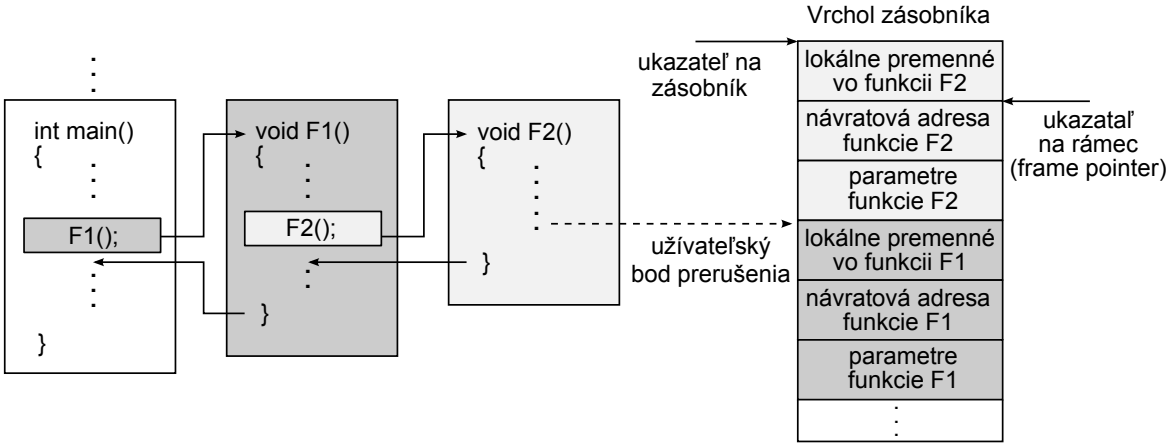
Ďalšou funkcionalitou, ktorú ladiaci nástroj poskytuje je získavanie informácii o globálnych a lokálnych premenných ako sú napríklad názov, typ, hodnota, bitová veľkosť a umiestnenie. Informácie o premenných sa získavajú z ladiacich informácií. Hodnota každej premennej je uložená na 32 bitoch preto je potrebné previezť túto hodnotu na odpovedajúci typ premennej. Prevod sa uskutočňuje pomocou bitových operácií. Pri 64-bitových dátových typoch sa spájajú dve po sebe idúce 32-bitové hodnoty. Ladiaci nástroj umožňuje nielen čítať hodnoty premenných, ale je taktiež možné tieto hodnoty nastavovať. Pri nastavovaní je zase naopak potrebné previesť hodnotu daného dátového typu na 32-bitovú hodnotu.

3.3 Vizualizácia stavu zásobníka

Vizualizácia stavu zásobníka (*stack back-trace*) zobrazuje informácie v akej časti programu resp. v akej funkcii sa práve nachádzame a ako sme sa tam dostali. Pri vizualizácii stavu zásobníka sa využívajú informácie získané pri vytváraní hierarchie funkcií. Samotné vytváranie hierarchie sa vykonáva nasledujúcim spôsobom. Na počiatočné adresy funkcií sa nastavia interné body prerušenia. Pri zachytení interného bodu prerušenia na začiatku funkcie sa získa adresa volania z predchádzajúcej hodnoty programového čítača. Pomocou nej získame návratovú adresu, na ktorú sa nastaví interný bod prerušenia. Pri zachytení interného bodu prerušenia na návratovej adrese vieme, že sme opustili danú funkciu a aktualizujeme hierarchiu funkcií. Na samotnú vizualizáciu stavu zásobníka sa využívajú informácie o lokálnych premenných, návratovej adrese a parametroch príslušných funkcií ako je vidieť na obrázku 1.

3.4 Vyhodnocovanie výrazov na úrovni jazyka C vrámci ladenia

Pri vyhodnocovaní jednoduchých výrazov na úrovni jazyka C sa v rámci ladenia použijú iba číselné a logické hodnoty a premenné. Pre vyhodnotenie konkrétneho výrazu je potrebné najskôr výraz analyzovať a oddeliť jednotlivé operandy a operátory výrazu. Pre zaistenie tejto funkcionality bude potrebné implementovať jednoduchý analyzátor výrazov. Taktiež je potrebná typová kontrola, aby sa zamedzilo vyhodnocovaniu vzájomne neprekonvertovateľných typov.



Obrázok 1: Vizualizácia stavu zásobníka.

3.5 POKRAČOVANIE PRÁCE

Predmetom ďalšieho rozširovania funkcionalít ladiaceho nástroja bude podpora pre získavanie informácii o zložitejších dátových štruktúrach. Pre implementáciu tejto funkcionality bude potrebné dodatočné spracovanie informácii z týchto dátových štruktúr, napríklad pre podporu ukazateľov bude potrebný mechanizmus, ktorý sa postupne prepracuje až k hodnote odkazovanej premennej.

Po úprave experimentálneho prekladača bude možné generovať ladiace informácie nazývané *Call Frame Information* (CFI), vďaka ktorým sa celý proces vytvárania hierarchie vykonávania funkcií zrýchli a zefektívni. Terajší stav prekladača však generovanie týchto informácií nepodporuje.

4 ZÁVER

Pri neustálom zvyšovaní počtu rôznych typov architektúr a modelov procesorov začína byť súčasný návrh hardvéru a softvéru dôležitým odvetvím, ktoré pri použití simulátoru a rekonfigurovateľného ladiaceho nástroja urýchli a zefektívni celý proces vývoja procesorov a aplikácií.

Táto práca predstavuje možnú realizáciu takého ladiaceho nástroja a po implementácii ďalších rozšírení by mohol tento ladiaci nástroj tvoriť plnohodnotnú súčasť produktu, ktorý bude poskytovať efektívny a rýchly spôsob vývoja procesorov a aplikácií.

POĎAKOVANIE

Tento príspevok vznikol vďaka podpore výskumného zámeru MSM0021630528 a grantu TAČR TA01010667.

REFERENCE

[1] J. Křoustek, Z. Přikryl, D. Kolář, and T. Hruška. Retargetable Multi-level Debugging in HW/SW Codesign. In 23rd International Conference on Microelectronics (ICM'11), page 6. Institute of Electrical and Electronics Engineers, 2011. ISBN 978-1-4577-2209-7.

[2] Rosenberg, B. J.: How Debuggers Work? Algorithms, Data Structures, and Architecture. Wiley Computer Publishing, 1996.

[3] Wilczák, M.: Ladicí nástroj generických simulátorů procesorů. Diplomová práce, Fakulta informačních technologií, Vysoké Učení Technické v Brně, 2010.