# USAGE OF A DWARF FORMAT IN DEBUGGER FOR GENERIC MICROPROCESSOR SIMULATORS

**Pavel Janečka**

Bachelor Degree Programme (3), FIT BUT

E-mail: xjanec11@stud.fit.vutbr.cz

Supervised by: Tomáš Hruška

E-mail:hruska@fit.vutbr.cz

**Abstract**: This paper describes basic facts about DWARF debugging format. Further it aids information on using it in order to debug simulator of generic microprocessors in project Lissom. The goal is to develop architecture independent source-level debugger.

**Keywords**: debugger, dwarf, source-level debugger

## 1. ÚVOD

Tato práce vznikla v rámci projektu Lissom, který se zabývá vývojem jazyka a nástrojů pro popis mikroprocesorů.

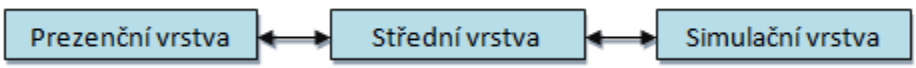
Při vývoji software a programování se uplatňuje proces ladění, který urychluje odhalování a odstraňování chyb. Obecně jej můžeme rozdělit na dvě kategorie: ladění na úrovni strojového kódu a zdrojového kódu. V této práci se budu zabývat druhou možností, protože ladicí nástroj (dále jen debugger) pro úroveň strojového kódu již v projektu Lissom existuje. Důležité je rozšíření v podobě ladění bez závislosti na architektuře procesoru.

## 2. ROZBOR

V rámci projektu Lissom existují nástroje pro vývoj procesorů a aplikací pro ně. Při ladění těchto aplikací je třeba použít takového formátu ladicích informací, který by mohl zaručit jejich dostupnost nezávisle na architektuře procesoru. Z exitujících používaných formátů této specifikaci nejlépe vyhovuje formát DWARF (podrobný popis formátu v [1]).

### 2.1. LADICÍ NÁSTROJ

Současný debugger používaný v rámci projektu Lissom pracuje jako aplikace na prezenční vrstvě architektury tohoto projektu (obr. 1).




**Obrázek 1:** Architektura projektu Lissom. Prezenční vrstva zasílá požadavky střední vrstvě. V případě debuggeru jsou to např. žádost o nastavení bodu přerušení, provedení jednoho kroku nebo výpis obsahu registrů. Střední vrstva odpovídá, mj. dále řídí komunikaci se simulátory. Ty v našem případě představují procesory, na kterých ladíme aplikace. Komunikace probíhá zasíláním zpráv podobajících se zjednodušenému XML dokumentu.

Zásadní rozdíl oproti klasickému debuggeru je zejména ten, že ladění probíhá vzdáleně. Tvorba ladicího nástroje na úrovni zdrojového kódu spočívá v nahrazení aplikace v prezenční vrstvě, úpravě protokolu a případných drobných změnách ve střední vrstvě.

# 3. FORMÁT LADICÍCH INFORMACÍ DWARF

Formát DWARF byl navržen tak, aby nebyl závislý na architektuře procesoru nebo programovacím jazyku. Teoreticky je tedy možné jej použít s libovolným formátem objektového souboru. V praxi se ovšem DWARF používá nejčastěji s formátem objektového souboru ELF a nástroje pro zpracování těchto ladicích informací jsou dostupné pouze pro tuto kombinaci.

## 3.1. ORGANIZACE LADICÍCH INFORMACÍ

Vnitřní reprezentaci ladicích informací tvoří záznamy (Debugging Information Entry, dále jen DIE), ty jsou dále uspořádány hierarchicky do stromové struktury. Kořenem je zpravidla zvláštní DIE - kompilační jednotka (Compilation Unit Entry - CUE) reprezentující objektový soubor. Potomci CUE nebo DIE záznamů jsou vnořené DIEs představující entity ladicích informací.

Fyzicky jsou ladicí informace uloženy ve spustitelných souborech v pojmenovaných sekcích začínajících prefixem **.debug_**. (např. .debug_info, .debug_lines, .debug_pubtypes).

## 3.2. POUŽITÍ LADICÍCH INFORMACÍ DWARF

Formát DWARF poskytuje širokou škálu informací o zkoumaném programu. Lze získat samozřejmě obsah proměnných (data object entry), rozsahy jejich platnosti, informace o typu (type entry) a také informace nižší úrovně. Užitečným přínosem DWARFu je také způsob, jakým reprezentuje zdroje konkrétního procesoru – jeho registry jsou mapovány na vnitřní reprezentaci registrů.

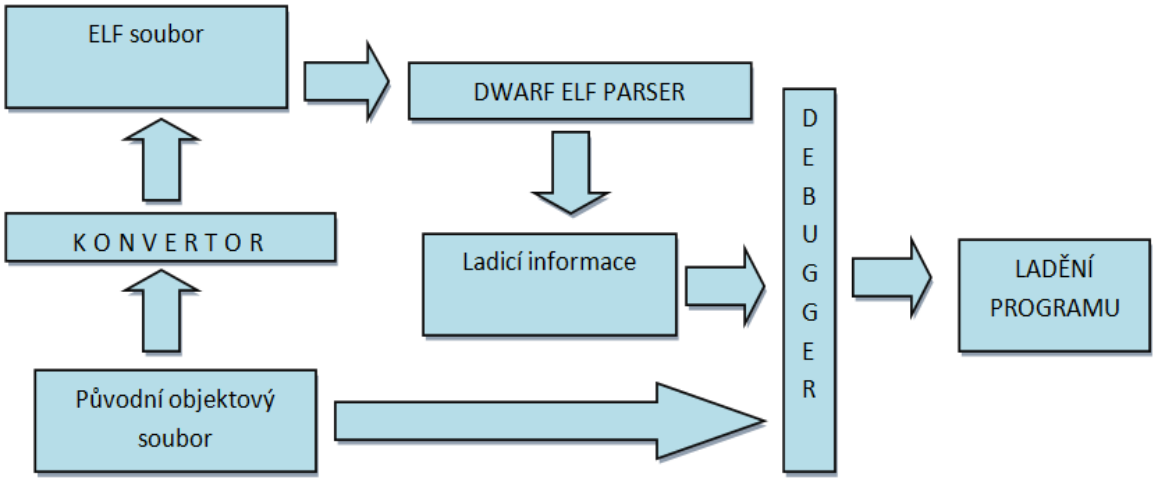
## 3.3. ZPRACOVÁNÍ FORMÁTU DWARF

Na interpretaci ladicích informací ve formátu DWARF existuje několik nástrojů. Projekt Lissom nicméně používá svůj vlastní formát objektového souboru, pro který se mají generovat ladicí informace. Ten se od ELFu odlišuje a existující nástroje nelze proto použít. Vyvstávají následující možnosti:

- Implementace vlastního nástroje pro zpracování formátu DWARF a spustitelného souboru (toto řešení se z časových důvodů jeví značně nereálné)
- Konverze spustitelného souboru na formát ELF, jehož kombinace s DWARFem je podporována existujícími nástroji. Předpokladem je existence takovéhoto konvertoru.
- Upravení části zpracovávající spustitelný ELF soubor tak, aby byla schopna zpracovat požadovaný spustitelný soubor.

# 4. ŘEŠENÍ

Vyzkoušel jsem nejprve poslední navrhované řešení, které se ale ukázalo jako neefektivní. Poté jsem se pokusil o realizaci druhé možnosti, tedy konverze spustitelného souboru na formát ELF (konvertor byl vyvinut v rámci projektu Lissom). Debugger tedy nyní pracuje tak, že je původní spustitelný soubor konvertován a pomocí nástrojů na zpracování DWARF a ELF formátů jsou posléze extrahovány ladicí informace a převedeny do vnitřní reprezentace. Nakonec jsou společně s původním spustitelným souborem předány jádru debuggeru. Proces lze znázornit tímto obrázkem:




**Obrázek 2:** Schéma debuggeru s využitím konvertoru

Současný stav debuggeru je ve stádiu, kdy je funkční mapování instrukcí na řádky zdrojového souboru v jazyce C. Tato skutečnost si vyžádala drobné rozšíření komunikačního protokolu, kde se při zaslání zprávy o bodu přerušení přenáší informace o umístění tohoto bodu ve všech souborech, kde se nachází. Dále jsem implementoval nastavování bodů přerušení ve zdrojovém souboru v jazyce C. Tyto funkce jsou dostupné přes rozhraní příkazové řádky.

Zamýšlenými rozšířeními jsou poskytování dalších informací, jako například obsahy registrů, paměti, callback stack, sledování proměnných, zisk jejich obsahu a další. Bude vhodné pokusit se zajistit mapování konkrétního sloupce na řádku zdrojového souboru v jazyce C na instrukce. Dále je v plánu přenesení funkcionality příkazové řádky také do vývojového prostředí.

## 5. ZÁVĚR

Cílem práce bylo seznámit čtenáře s možným postupem při tvorbě debuggeru na úrovni zdrojového kódu nezávislého na architektuře procesoru. Zároveň bude výsledek této práce použit v rámci projektu Lissom.

## PODĚKOVÁNÍ

Tento příspěvek vznikl za podpory grantů MPO FR-TI1/038, TAČR Alfa TA01010667 a výzkumnému záměru MSM 21630528.

## REFERENCE

[1] Unix International: DWARF Debugging Information Format [online]. [cit. 2011-03-03]. Dostupné z WWW: <http://dwarfstd.org/doc/dwarf-2.0.0.pdf>

[2] *Lissom Project Introduction* [online]. [cit. 2011-03-03]. <http://www.fit.vutbr.cz/research/groups/lissom/index.html>.

[3] Rosenberg, B., J.: How Debuggers Works, Wiley, 1996, ISBN 0-471-14966-7.

[4] Milan Wilczák: Ladicí nástroj generických simulátorů procesorů, diplomová práce, Brno, FIT VUT v Brně, 2010