EMULATION OF CPU ARCHITECTURES

Lukáš Charvát
Bachelor Degree Programme (1), FIT BUT
E-mail: xcharv03@stud.fit.vutbr.cz

Supervised by: Aleš Smrčka
E-mail: smrcka@fit.vutbr.cz

ABSTRACT

The paper discusses the design of a CPU architecture emulator aimed to assembly languages course. While most of nowadays emulators are architecture specific, this paper describes an approach for creating an emulator, which allows users to easily set up their own architecture, to perform operations upon it, and to display its current state.

1 ÚVOD

Cílem projektu je vyvořit emulátor s možností modifikace architektury nebo její instrukční sady pomocí jednoduchého jazyka a tím jej učinit vhodným i vyúku asemblerů.
Pokud se podíváme na zástupce současných emulátorů, většina z nich postrádá možnost jednoduše měnit emulovanou architekturu nebo její instrukční sadu. Ze zástupců emulátorů schopných emulovat více architektur jmenujme např. ZX32 [1], emulátor několika verzí počítače Sinclair ZX Spectrum, jejichž výčet ale nelze nijak rozšiřovat. Jiným zástupcem je např. projekt PearPC [2]. Emulovaná architektura PowerPC je zde modifikovatelná pouze omezeně pomocí nastavení v konfiguračních souborech. Projekt MAME [3], zaměřující se na emulaci arkádových herních strojů, dovoluje vytvářet uživateli vlastní architektury pomocí popisu jednotlivých komponent na úrovni číslicových obvodů. Nabízí tak naopak velkou modularitu ovšem za cenu náročného popisu komponent.

2 POŽADAVKY NA SYSTÉM

Ze základního požadavku jednoduchosti můžeme vytvořit další vlastnosti výsledného programu:

- schopnost emulovat co největší množství počítačových architektur
- možnost definovat si vlastní funkce měnící stav emulované architektury nebo architekturu samotnou
- jednoduché, pro začátečníky vhodné, názorné a přehledné uživatelské rozhraní schopné pracovat pod více operačními systémy
- schopnost grafického rozhraní názorně zobrazovat logické vazby mezi prvky, sdružovat je do skupin a pak tyto skupiny zobrazovat

3 NÁVRH

S ohledem na požadavky kladené na systém můžeme návrh projektu rozdělit na tři části. V návrhu jazyka pro popis jsou rozebrány společné vlastnosti architektur, a na jejich základě je pak vystavěna gramatika jazyka. Dále je navržen vzhled a schopnosti grafického uživatelského rozhraní. Poslední část návrhu se zaměřuje na princip činnosti emulátoru.

3.1 JAZYK PRO POPIS ARCHITEKTUR A OPERACÍ NAD NIMI

První dva požadavky vedou k vytvoření jednoduchého jazyka pro popis architektury. Pro tento účel byl navrhnut jazyk inspirovaný jazykem Python, nabízející stručné a jednoduché vyjadřování. Pro dosažení schopnosti emulovat co nejvíce počítačových architektur, obsahuje jazyk příkazy pro tvorbu základních prvků architektur. Z pohledu identifikace daných prvků je možné rozdělit je na dva druhy:

- jedinečné, v architektuře jednoznačně identifikované (např. akumulátor)
- skupiny prvků, kde jsou jednotlivé prvky identifikované názvem a počtem prvků (např. paměť), k jednotlivým prvkům se pak přistupuje pomocí indexace

Bitová šířka obou typů prvků může nabývat libovolného rozsahu (vývoj budoucích architektur nesmí být omezen architekturami současnými). Některé virtuální architektury uvažují ve své specifikaci i nekonečný počet prvků skupiny (např. `heap` u Java Virtual Machine [4]). Zmíněné prvky je emulátor schopen dynamicky vytvářet nebo je naopak z architektury odebírat (např. `stack frame` u Java Virtual Machine). Dalším častým jevem v počítačových architekturách, který je schopen jazyk popsat, je dynamické vytváření prvků, které sice v architektuře vystupují samostatně, ale ve skutečnosti jsou částí jiného prvku (např. registr `ax` vystupuje jako část registru `eax` u architektury Intel x86 [5]).
Pro reprezentaci přesunů hodnot mezi jednotlivými prvky architektury jazyk disponuje prostředky pro vyjádření operace přiřazení a to nejen nad celými prvky (u jedinečných prvků), ale i nad jejich částmi tzv. řezy. Dále je jazyk schopen vyjádřit nad prvky základní matematické operace (sčítání, odčítání, násobení, dělení, bitové posuny). Jazyk také umožnuje seskupování výše uvedených sledů operací do funkcí.
Pro potřeby grafického rozhraní jazyk poskytuje možnost seskupovat prvky do kolekcí určených k zobrazení uživateli. Pro vyjádření logických vazeb mezi prvky umožňuje jazyk vytvářet tzv. ukazatele (např. pro zvýraznění vrcholu zásobníku u architektury Intel x86).
Na základě výše uvedených požadavků na schopnosti jazyka byla vytvořena gramatika, dostupná k nahlédnutí na stránkách projektu [6]. Na následujících řádcích je naznačena ukázka popisu architektury Intel x86.

```
object eax = bits[0:31]; // Vytvoření 32bitového registru EAX
symbol ax = eax[0:15];   // Registr AX je částí registru EAX

array<bits[0:7]> memory(1048576);          // Vytvoření paměti
pointer<memory> s_pointer = ss << 4 + sp; // Logická vazba
```

3.2 GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ

Grafické uživatelské rozhraní poskytuje možnost komfortního psaní kódu v asembleru zvolené architektury, tak i zobrazování jejího aktuálního stavu. Jelikož je emulátor vyvíjen primárně pro výuku asemblerů, rozhraní zvýrazňuje efekt jednotlivých instrukcí na emulovanou architekturu vyznačením právě změněných prvků a nabízí možnosti zobrazení hodnot prvků v hexadecimálním, desetinném a binárním kódování. Pro začátečníky je také neméně důležité zvýraznění jednotlivých logických vazeb mezi prvky pomocí ukazatelů. Rozhraní také poskytuje možnost krokování kódu v asembleru s možnostmi krokovat kód manuálně, za pomocí časovače nebo s použitím bodů přerušení (angl. breakpoints).

3.3 INTERPRETACE

Interpret je navržen jako dynamický—změny v architektuře se projevují po jednotlivých událostech (tj. provedených příkazech jazyka pro popis architektury).
Po vzniku uživatelem generované události v grafickém uživatelském rozhraní (např. příkaz zpracování další instrukce) se volá interpret instrukcí [7], jehož cílem je převádět právě prováděné instrukce na příkazy jazyka pro popis architektury. Po jejich provedení je nový stav architektury zobrazen uživateli.

4 ZÁVĚR

Na základě výše uvedeného návrhu byla sestavena gramatika, s níž je možné generovat jazyk pro popis architektury. Pomocí jazyka byla popsána architektura procesoru Intel x86, která byla následně zprovozněna v prototypové verzi emulátoru [6].
Po dokončení implemetace by měl vzniknout nástroj schopný emulovat vetšinu dnešních a díky své univerzálnosti i budoucích architektur se zaměřením na výuku asemblerů.

REFERENCE

[1] ZX32, emulátor počítače Sinclair ZX Spectrum, dostupné online na http://www.geocities.com/SiliconValley/Bay/9932/

[2] PearPC, emulátor počítače PowerPC, dostupné online na http://pearpc.sourceforge.net/

[3] MAME - Multiple Arcade Machine Emulator, dostupné online na http://mamedev.org

[4] Lindholm, T., Yellin F.: The Java Virtual Machine Specification, Second Edition, Prentice Hall PTR, 1999. ISBN 978-0201432947. dostupné online na http://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html

[5] Intel 64 and IA-32 Architectures Software Developer's Manuals, Vol. 1 Basic Architecture, dokument dostupný na http://download.intel.com/design/processor/manuals/253665.pdf

[6] Stránky projektu, dostupné online na http://www.stud.fit.vutbr.cz/~xcharv03/projects/IBP

[7] Kovács Z.: Interpret instrukcí emulátoru CPU, diplomová práce, plánované odevzdání 2009