Posudek oponenta bakalářské práce

Student: Adam Svoboda
Název: Simulátor procesoru podorujícího MIPS ISA

Znalost principu činnosti procesorů je důležitá ne jen pro ty, kdo nové procesory navrhují, ale i pro jejich uživatele, programátory. Umožňuje jim pochopit jakým způsobem je algoritmus realizovaný na úrovni hardware a jaká omezení pro propustnost z toho vyplývají a které operace a jejich kombinace jsou časově nákladné. Tato znalost pak umožňuje optimalizovat jak algoritmy tak slouží jako vodítko při výběru datových struktur. Pro seznámení s principy činnosti procesoru je pro studenty přínosné použít simulátor, který stav procesoru a vykonávání instrukcí zobrazuje. Vytvoření takovéhoto simulátoru s názorným zobrazením stavů a signálů bylo zadáním hodnocené bakalářské práce.

Vybraná procesorová architektura MIPS je dostatečně jednoduchá pro seznámení s problematikou a přitom umožňuje demonstrovat již základní techniky pro zvýšení propustnosti systému. Práce realizuje procesor s pětistupňovým zpracováním instrukcí včetně grafického zobrazení hodnot signálů na výstupu jednotlivých funkčních jednotek a registrů a stavů vyrovnávacích pamětí.

Práce začíná popisem architektury MIPS. V další kapitole je popsaná implementace emulátoru procesoru. Následuje srovnání s dalšími, v současné době dostupnými, simulátory architektury. Výběr simulátorů je omezený, nevyskytují se v něm simulátory určené pro běh rozsáhlých programů a operačních systémů s maximální efektivitou interpretace/překladu za běhu (QEMU, MAME-MESS). Zaměření projektu i výběr alternativ pro srovnání je orientovaný na simulátory, které jsou určeny pro demonstraci činnosti procesoru a výuku, a tak je možné omezený výběr považovat za přijatelný. Textová část práce je zakončena kapitolou popisující kompletní navržený program včetně pokynů pro jeho použití.

Seznam použité literatury odkazuje pouze na sekundární zdroje popisu architektury MIPS. Zásadní autoritativní/kompletní zdroje informací chybí. Při popisu a především implementaci by měl být návrh emulace instrukcí porovnaný s popisem v oficiálním dokumentu MIPS32® Architecture For Programmers, především se jedná o svazek Volume II: The MIPS32® Instruction Set. Originálně vydaný MIPS Technologies Inc. Nyní stejně jako celá architektura MIPS spravovaný Imagination Technologies LTD.

Vytvořený zdrojový kód je rozdělený na vlastní emulátor procesoru a grafické rozhraní. Vlastní procesor je pak realizovaný modulárně pospojováním menších stavebních bloků. Toto řešení je vhodné pro vizualizaci průchodu instrukcí procesorem. Emulátor také modeluje činnost vyrovnávací paměti (cache). V grafické reprezentaci vyrovnávacích pamětí však není implementované zobrazení statistiky přístupů do paměti obsloužených z vyrovnávací paměti. Činnost programu jsem ověřil krátkými testy. Program se jevil jako stabilní avšak jeho použitelnost je limitovaná množstvím nedostatků.

Zpracování vstupního souboru obsahujícího kód v assembleru vyžaduje přesné vkládání znaku mezera. Před instrukcí mezera být nesmí, za instrukcí musí být právě jedna mezera, stejně tak operandy (registry, konstanty) musí být od sebe odděleny mezerou. I když je zvykem mezery do kódu v assembleru psát, tak často slouží pro pohodlné čtení a zarovnání instrukcí a operandů pod sebou. Program nejen že vynucuje použití právě jedné mezery v místech, kde není v běžné notaci zápisu assembleru povinná, ale při použití mezery na začátku řádku celý řádek ignoruje. Chybě je implementované i zpracování operandů v instrukci „lui“, kdy je vyžadovaný zdrojový registr, což odporuje oficiální specifikaci instrukce. Vlastní diagnostika chyb při načítání zdrojového kódu je spíše nedostačující.

V provádění vlastních instrukcí je také množství závažných chyb. Cyklus vytvořený zápisem „loop:beq $0, $0, loop“ nevedl k provedení skoku. Instrukce „ori“ (pravděpodobně i „ani“ a další instrukce) nesprávně interpretuje/vyžaduje přímý operand jako znaménkový. Při implementaci měl být konzultovaný primární popis MIPS ISA. Tato chyba je o to závažnější, že doporučená sekvence

instrukcí „lui“ a „ori“ používaná pro naplnění registru hodnotou v kompletním 32-bitovém rozsahu bude pro hodnoty s jedničkovým bitem 15 vykazovat chybu. Přitom tato sekvence je použitá i pseudoinstrukcemi „la“ a „li“.

Návrh oken a formulářů pro zobrazení stavu procesoru je zafixovaný na pevné rozměry jak prvků tak oken a neumožňuje přizpůsobení rozlišení obrazovky nebo požadavkům uživatele. Předpokládám, že i změna velikosti použitých fontů povede k chybám v zobrazení. Přitom v dnešní době je škála rozlišení monitorů a zařízení velmi široká a na moderním laptopu s 4K rozlišením bude simulátor použitelný jen s obtížemi. Z tohoto pohledu je i nevhodné použití rastrového obrázku pro zobrazení diagramu procesoru. Použití vektorového formátu (například SVG) by bylo vhodnější a šlo by i v některých případech s využitím parametrů použít přímo pro zobrazení hodnot a třeba i barvení spojů. Ještě výhodnější by bylo přímé navázání prvků modelu na systém jejich zobrazování a propojení.

Integrace překladu přímo do simulátoru sice zjednodušuje instalaci, ale volání externího assembleru a načítání binárního spustitelného formátu ELF by možná nebylo složitější na realizaci a umožnilo by pozorovat činnost kódu zkompilovaného z vyšších jazyků (například „C“). I kompletní podpora assembleru s bohatou sadou pseudoinstrukcí, preprocesorem a spojováním více objektových modulů znamená ve vlastní režii extrémní množství práce.

Práci **doporučuji k obhajobě**. Práci však považuji spíše za experiment/prototyp než použitelný program a hodnotím ji klasifikačním stupněm **uspokojivě (D)**. I přes toto hodnocení student prokázal schopnost složitější program v jazyce Java naprogramovat a věřím, že by při dostatečném množství dalšího věnovaného úsilí bylo možné program upravit a rozšířit do podoby, kdy by byl přínosnou pomůckou pro studenty seznamující se s procesorovými architekturami.

V Praze, dne 15. 6. 2016

Ing. Pavel Píša, Ph.D.
Katedra řídicí techniky
Fakulta elektrotechnická
České vysoké učení technické