FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Posudek oponenta závěrečné práce

Oponent práce:	Ing. Tomáš Pecka
Student:	Bc. Martin Prokopič
Název práce:	Modulární překladač pro TinyC
Obor / specializace:	Systémové programování
Vytvořeno dne:	30. května 2023

Hodnotící kritéria

1. Splnění zadání

- ▸ [1] **zadání splněno**
  [2] zadání splněno s menšími výhradami
  [3] zadání splněno s většími výhradami
  [4] zadání nesplněno

Téma hodnotím jako náročnější. Student ale i tak zadání bez výhrad splnil a implementoval i věci nad rámec zadání.

2. Písemná část práce 95 /100 (A)

Autor popisuje implementaci překladače se zaměřením na middle-end a zvláště backend. Popisuje jednotlivé fáze generování kódu, od návrhu IR až po algoritmy optimalizací. Jednotlivé kapitoly na sebe velmi dobře navazují a práce velmi dobře popisuje jednotlivé kroky prováděné překladačem spolu s potřebnými algoritmy. Teoretická část je proto delší a náročnější na čtení, ale to je vzhledem k tématu v pořádku.

Práce je psána v anglickém jazyce na vynikající úrovni. Po typografické stránce je práce nadprůměrná. Při čtení jsem si všiml pár problémů, jako jsou např. nepřiměřeně dlouhé mezery po "et al.", občasně vyskytující se sirotci, či horizontálně nezarovnané popisky u obrázků v subfigures. Nejsem si jistý, zda se tabulce 2.3 dá říkat tabulka. Nic z toho ale nijak významně neruší čtení a nedostatky se vyskytují jen zřídka.

I přesto, že autor několikrát v textu zmiňuje, že rychlost kompilátoru není problém, tak bych uvítal nějaký alespoň základní benchmarking jak optimalizace a zvolený počet registrů VM ovlivňují rychlost překladu.

3. Nepísemná část, přílohy 94 /100 (A)

Výsledný kompilátor je dodaný s CLI rozhraním, které dovoluje spuštět pouze jednotlivé fáze překladu i kompletní překlad z tinyC do t86. Při průchodu zdrojovými kódy

kompilátoru jsem občas zápasil s některými syntaktickými konstrukcemi jazyka Scala (kód v tomto jazyce vidím prakticky poprvé), ale i tak jsem většinou brzy pochopil, co kód implementuje.

Kompilátor je otestován velmi základními unit testy a velkými "integračními" testy, kde se testuje kompletní průběh včetně spuštění výsledného programu. Myslím ale, že jednotkové testování by mohlo být extenzivnější a rozšířeno i na další třídy.

Z asi 40 dodaných ukázkových kódu tinyC jsem zkoušel některé přeložit (jak kompletně, tak po jednotlivých fázích generování) do t86 a spustit a vše dopadlo podle očekávání až na jeden test, kde se kompilátor nejspíše zacyklil při zvoleném nízkém počtu registrů (2) pro T86. Možná by bylo vhodné rozšířit testy tak, aby nepracovaly vždy pouze s čtyřmi dostupnými registry ve VM.

Předpokládám, že kompilátor může být dále rozšiřován v rámci dalších DP, tak bych uvítal možná lepší historii ve verzovacím nástroji git, z některých commitů mi nebylo jasné co a proč se děje.

4. Hodnocení výsledků, jejich využitelnost 100 /100 (A)

Pro studenty předmětu NI-GEN, kde se implementuje backend kompilátoru pro překlad do T86 architektury, může být tento překladač vítanou pomůckou (např. jako referenční kompilátor).

Celkové hodnocení 97 /100 (A)

Písemné i nepísemné části práce jsou na velmi dobré úrovni. Nemám důvod hodnotit práci jiným stupněm než je A (výborně).

Otázky k obhajobě

- Které části kompilátoru budete dávat k dispozici studentům NI(E)-GEN? Pokud budou mít k dispozici Vaše kompletní řešení, mohou části Vašeho kódu přebrat a tím si zjednodušit implementaci své semestrální práce.
- V závěru píšete, že Váš překladač generuje kratší a efektivnější kód než naivní překladač na obou testovaných případech, ale pouze dva zmíněné testované programy mi přijdou jako malý počet. Zkoušel jste i další programy? Pokud ne, jak byste postupoval ve vyhodnocení na větším vzorku programů?

Instrukce

Splnění zadání

Posuďte, zda předložená ZP dostatečně a v souladu se zadáním obsahově vymezuje cíle, správně je formuluje a v dostatečné kvalitě naplňuje. V komentáři uveďte body zadání, které nebyly splněny, posuďte závažnost, dopady a případně i příčiny jednotlivých nedostatků. Pokud zadání svou náročností vybočuje ze standardů pro daný typ práce nebo student případně vypracoval ZP nad rámec zadání, popište, jak se to projevilo na požadované kvalitě splnění zadání a jakým způsobem toto ovlivnilo výsledné hodnocení.

Písemná část práce

Zhodnoťte přiměřenost rozsahu předložené ZP vzhledem k obsahu, tj. zda všechny části ZP jsou informačně bohaté a ZP neobsahuje zbytečné části. Dále posuďte, zda předložená ZP je po věcné stránce v pořádku, případně vyskytují-li se v práci věcné chyby nebo nepřesnosti.

Zhodnoťte dále logickou strukturu ZP, návaznosti jednotlivých kapitol a pochopitelnost textu pro čtenáře. Posuďte správnost používání formálních zápisů obsažených v práci. Posuďte typografickou a jazykovou stránku ZP, viz Směrnice děkana č. 52/2021, článek 3.

Posuďte, zda student využil a správně citoval relevantní zdroje. Ověřte, zda jsou všechny převzaté prvky řádně odlišeny od vlastních výsledků, zda nedošlo k porušení citační etiky a zda jsou bibliografické citace úplné a v souladu s citačními zvyklostmi a normami. Zhodnoťte, zda převzatý software a jiná autorská díla, byly v ZP použity v souladu s licenčními podmínkami.

Nepísemná část, přílohy

Dle charakteru práce se případně vyjádřete k nepísemné části ZP. Například: SW dílo – kvalita vytvořeného programu a vhodnost a přiměřenost technologií, které byly využité od vývoje až po nasazení. HW – funkční vzorek – použité technologie a nástroje, Výzkumná a experimentální práce – opakovatelnost experimentů.

Hodnocení výsledků, jejich využitelnost

Dle charakteru práce zhodnoťte možnosti nasazení výsledků práce v praxi nebo uveďte, zda výsledky ZP rozšiřují již publikované známé výsledky nebo přinášející zcela nové poznatky.

Celkové hodnocení

Shrňte stránky ZP, které nejvíce ovlivnily Vaše celkové hodnocení. Celkové hodnocení nemusí být aritmetickým průměrem či jinou hodnotou vypočtenou z hodnocení v předchozích jednotlivých kritériích. Obecně platí, že bezvadně splněné zadání je hodnoceno klasifikačním stupněm A.