MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Optimalizace aplikací distribuce teTEX

BAKALÁŘSKÁ PRÁCE

Marek Lipovčan

Brno, jaro 2003

Prohlášení

Prohlašuji, že tato bakalářská práce práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: RNDr. Petr Sojka

Shrnutí

teTEX je jednou z nejrozšířenějších distribucí TEXu a přidružených programů. Některé jeho součásti jsou poměrně výpočetně náročné (např. generování vzorů pro dělení slov). Cílem tohoto projektu je analyzovat možnosti optimalizace těchto programů a tyto poznatky prakticky ověřit. Z teTEXu byly pro tyto účely vybrány programy PatGen, TEX, eTEX, pdfTEX, pdfeTEX, Omega a dvipdfm. Hlavním zájmem bylo optimalizovat programy pro generování vzorů. Kromě programu PatGen byl analyzován také program OPatGen.

Klíčová slova

TEX, teTEX, Optimalizace, Generování vzorů, PatGen, OPatGen, UPatGen

Předmluva

Většina dokumentů je v dnešní době vytvořena a zpracovávána v elektronické formě. Mezi nejpoužívanější programy pro tvorbu dokumentů patří bezesporu TEX (obzvláště v akademické sféře). Vzhledem k době vzniku a díky kvalitnímu zpracování nepatří TEX mezi programy, které by kladly vysoké nároky na použitý hardware, obzvláště dnes, kdy jsou vysoce výkonné počítače dostupné i běžnému uživateli. Stále ale existují úkoly, jejichž zpracování je časově náročnější. Proto je velmi užitečné vědět, jestli lze tyto aplikace optimalizovat a zdali to vůbec má smysl.

Pro kvalitní a uhlazenou sazbu textů je nutné použít kvalitní algoritmus pro dělení slov. TEX tento problém vyřešil vysoce efektivní a elegantní cestou – pomocí vzorů. Prvním programem pro generování těchto vzorů byl PatGen. Ten má ale několik významných omezení. Mezi největší problémy patří bezesporu omezení na 8-bitové kódování (obzvláště ve spojení se systémem Omega). V nedávné době vznikl na Fakultě informatiky projekt zabývající se tímto problémem [Antos]. Výsledkem je program OPatGen. Vedlejším produktem čistého návrhu tohoto programu je ale výrazné snížení rychlosti generování (zhruba 10 $\times$ oproti PatGenu). Zde se jeví jako velmi praktické zjištění, lze-li tento program optimalizací urychlit.

V praktické části jsem pomocí testů zjistil, jaký vliv na rychlost uvedených programů mají parametry překladu a pokusil se nalézt nejvhodnější z nich. Dále jsem se pokusil dokázat, že lze vytvořit program pro generování vzorů, který by měl stejné schopnosti jako OPatGen, ale přitom by nebyla degradována jeho výkonnost. Výsledkem je návrh a částečná implementace programu UPatGen.

Obsah

1 **Úvod do optimalizace** . . . 5
- 1.1 *Výhody optimalizace* . . . 5
- 1.2 *Nevýhody optimalizace* . . . 5
- 1.3 *Jak na to* . . . 5
  - 1.3.1 Optimalizace hardware . . . 6
  - 1.3.2 Optimalizace kompilací . . . 6
  - 1.3.3 Optimalizace použitím rychlejších knihoven . . . 7
  - 1.3.4 Optimalizace zdrojového kódu. . . . 7

2 **Úvod do systému teTEX** . . . 9
- 2.1 *Hlavní komponenty* . . . 9
  - 2.1.1 Kpathsea . . . 9
  - 2.1.2 Web2c . . . 9
  - 2.1.3 Rozšíření TEXu . . . 9
  - 2.1.4 Další podpůrné nástroje a skripty . . . 9
- 2.2 *Adresářová struktura teTEXu* . . . 10
- 2.3 *Jak vypadají zdrojové kódy* . . . 10
- 2.4 *Jak se překládá teTEX pod Linuxem* . . . 10
- 2.5 *Konfigurace* . . . 11

3 **Optimalizace aplikací distribuce teTEX pomocí kompilace** . . . 13
- 3.1 *Hardware použitý pro testy* . . . 13
- 3.2 *Aplikace vybrané pro testy* . . . 14
- 3.3 *Testovací soubory* . . . 15
- 3.4 *Parametry překladů* . . . 15
- 3.5 *Struktura adresáře s testy* . . . 16
- 3.6 *Postup testování* . . . 16
- 3.7 *Výsledky* . . . 17
  - 3.7.1 Stroj č. 1 (Athlon XP 1700+) . . . 18
  - 3.7.2 Stroj č. 2 (Intel Mobile Pentium III 800) . . . 18
  - 3.7.3 Stroj č. 3 (Athlon XP 1600+) . . . 19
  - 3.7.4 Stroj č. 4 (2 × Intel Xeon 2,2) . . . 19
- 3.8 *Doporučení* . . . 19

4 **Profilování** . . . 21
- 4.1 *Vytvoření a analýza profilu* . . . 21
- 4.2 *Ukázky profilování* . . . 22
  - 4.2.1 Profily aplikace PatGen . . . 22
  - 4.2.2 Profily aplikace OPatGen . . . 22

5 **Generování vzorů dělení slov** . . . 23
- 5.1 *Vzory* . . . 23
- 5.2 *Generování vzorů* . . . 24
- 5.3 *Přehled existujících programů pro generování vzorů* . . . 24
  - 5.3.1 PatGen . . . 24
  - 5.3.2 OPatGen . . . 24

5.4 *UPatGen* 25
5.4.1 Hlavní cíle 26
5.4.2 Návrh 26
5.4.3 Implementace 27
6 **Závěr** 29
A **Parametry překladu** 30
B **Kompletní výsledky testů** 35
C **Kompletní ukázky profilů** 48
D **Obsah CD** 50

Kapitola 1

Úvod do optimalizace

Doby, kdy optimalizace byla nezbytná kvůli výkonnosti počítačů, jsou už dávno pryč. Současné počítače jsou již natolik rychlé, že pro většinu běžných uživatelských programů to nemá smysl. Nicméně stále existuje spousta speciálních aplikací, pro které je optimalizace více než vhodná. Nejprve bych se pokusil nastínit důvody, proč a kdy optimalizovat a proč to někdy není vhodné. Dále uvedu metody, jak to provést.

1.1 Výhody optimalizace

Jedním z hlavních důvodů pro optimalizaci je samozřejmě urychlení programu. Toto má smysl pouze pro programy, jejichž rychlost je podstatná. Často jsou důležitější spíše paměťové nároky než výpočetní výkon. V takovém případě má význam optimalizovat program pro úsporu paměti. Někdy je také nutné provést optimalizaci kvůli specifickému hardware.

1.2 Nevýhody optimalizace

Mezi hlavní nevýhody patří časová náročnost – optimalizovat program není triviální. Je důležité si rozmyslet, jestli má vůbec cenu strávit nad tím další čas. Optimalizace navíc v drtivé většině přidá další složitost a i přehledný program se může lehce změnit v programátorskou noční můru. Častým produktem optimalizace je také velké kvantum nově vzniklých chyb, jejichž odstranění si vyžádá další čas navíc. Mnohé optimalizace mohou způsobit problémy s portabilitou. Je dobré vyhnout se „nečistým trikům“, které sice program urychlí, ale způsobí pozdější problémy se stabilitou a přenositelností.

1.3 Jak na to

Někteří programátoři zastávají názor, že pokud chceme rychlost, musíme to napsat v assembleru. Tento přístup ale mnohdy vede do pekel. Kód v assembleru se velmi špatně čte, nehledě k tomu, že současné překladače generují často lepší kód než ten ručně psaný. Jak tedy optimalizovat? Každému kódu by měl předcházet návrh programu. Když vytvoříme dobré programové schéma včetně popisu datových struktur, při kterém budeme myslet na pozdější optimalizaci, vše půjde rychleji. V prvé řadě musíme mít program plně funkční. Nejprve tedy napíšeme program bez velkých optimalizací, řádně ho odladíme, ověříme správnost a teprve poté přijde na řadu optimalizace. Je vhodné použít ladící a pomocné nástroje jako jsou gdb[1], lclint[2], Electric Fence[3] a další.

1. http://www.gnu.org/directory/gdb.html
2. http://www.splint.org/
3. ftp://ftp.perens.com/pub/ElectricFence/

Optimalizace většinou udělá program hůře čitelným, proto je skutečně nutné odstranit všechny chyby ještě před ní.

Dalším krokem by mělo být určení slabých míst – částí kódu, jejichž zpracování zabere nejvíce systémových prostředků. Tyto prostředky dělíme do 3 hlavních skupin: procesorový čas, použitá operační paměť, I/O operace. V případě optimalizace zdrojových kódů je důležité najít vhodné místo pro optimalizaci. Jestliže například optimalizujeme funkci zabírající 50 % celkového času a podaří se nám ji dvakrát zrychlit, pak výsledný program poběží o 25 % rychleji. Jestliže totéž uděláme s funkcí, která zabere pouze 10 % celkového času, pak výsledkem bude zrychlení o zhruba 5 %. Je zřejmé, že nalezení vhodného místa je důležité pro úspěšnou optimalizaci. K tomuto účelu se používá technika zvaná *profilování* (viz Kapitola 4, *Profilování*). Příkladem aplikací pro vytváření profilů je např gprof. Abychom mohli takto program analyzovat, musíme jej přeložit s podporou pro profilování.

Existuje několik základních metod pro optimalizaci v závislosti na dostupných prostředcích, předpokladech a požadavcích. Lze je rozdělit na 2 nejvýznamnější skupiny:

- optimalizace bez modifikace zdrojových kódů;
- optimalizace modifikací zdrojových kódů.

Následuje popis těch nejdůležitějších z obou skupin (je zcela zřejmé, do které skupiny daná metoda patří). Všechny zde uvedené lze různě kombinovat.

1.3.1 Optimalizace hardware

Jedná se o nejjednodušší metodu. Vzhledem k rychlému vývoji výpočetní techniky je velmi pravděpodobné, že výměnou příslušného hardware můžeme vyřešit náš problém. Pro urychlení výpočtu algoritmů je vhodné použít rychlý procesor, v případě nedostatku odkládacího prostoru pak paměť, případně pevný disk. Lze ale také pořídit speciální obvody, které budou na úrovni hardware provádět některé funkce našeho programu. Typickým příkladem takového obvodu je grafický akcelerátor na grafické kartě.

1.3.2 Optimalizace kompilací

Většina současných překladačů nabízí možnosti pro vyladění překladu formou specifikování optimalizačních parametrů. Existují jak obecné parametry, tak speciální pro konkrétní účely. Tyto parametry se liší v závislosti na použitém překladači. Pokud jich máme více na výběr, je vhodné použít ten, který bude pro náš program vytvářet nejrychlejší kód. Uvedu popis nejpoužívanějších parametrů pro překladač gcc verze 3.2.

–g Přidá informace nutné pro ladění.

–Wall Zapne hlášení nejběžnějších chyb.

–O0 Vypne optimalizaci.

–O1, –O2, –O3 Zapne optimalizaci (O1 nejnižší optimalizace, O3 nejvyšší optimalizace). Tento parametr pouze vyvolá předdefinovanou množinu optimalizačních parametrů.

–Os Zapne optimalizaci pro co nejmenší velikost výsledného kódu.

–march, –mcpu Produkuje kód vyladěný pro danou architekturu/procesor (i386, i686, pentiumpro, k6, athlon–xp). Umožní použití specifických instrukcí pro danou architekturu. Může produkovat kód, který nepoběží na nižší platformě.

–fomit–frame–pointer Uvolní jeden registr pro jiné použití. Na některých platformách tento parametr znemožňuje ladění.

–ffloat–store Tento parametr zabraňuje ukládání některých proměnných s pohyblivou desetinnou čárkou do registrů. Použije se v případě, že daná architektura poskytuje větší přesnost než je potřeba.

–fforce–mem Před provedením operace zkopíruje operand do registru.

–fforce–addr Před provedením operace zkopíruje adresu paměti do registru.

–finline–functions Vloží jednoduché funkce přímo tam, odkud se volají (inline).

–ffast–math Urychlí matematické funkce. Může vygenerovat kód produkující špatné výsledky (u programů vyžadujících striktní dodržování IEEE/ISO pravidel). Na druhou stranu může program výrazně urychlit.

–fstrength–reduce, –frerun–loop–opt, –funroll–loops Pokusí se optimalizovat smyčky.

–fthread–jumps Optimalizuje skoky.

–fexpensive–optimizations Provede více menších optimalizací, které ale mohou být být časově náročnější.

Podrobnější informace k těmto i dalším parametrům lze nalézt v manuálové stránce překladače gcc, případně v článku [Myth].

1.3.3 Optimalizace použitím rychlejších knihoven

Jestliže náš program využívá nějaké staticky nebo dynamicky linkované knihovny, je vhodné se podívat, zda neexistují ekvivalentní knihovny, které jsou více optimalizované, případně si tyto knihovny přeložit.

1.3.4 Optimalizace zdrojového kódu.

Je důležité uvědomit si, co daný úsek kódu vlastně dělá a jestli toto nelze řešit lépe. To platí zejména o výběru vhodných algoritmů pro třídění (bubble sort $\times$ quick sort, merge sort) nebo vyhledávání (sekvenční, indexované vyhledávání, hešování, KMP, ...). Podstatného zrychlení lze také dosáhnout použitím vhodných datových struktur. Užitečná je analýza všech cyklů a míst větvení programu. Výhodně optimalizovat se také dají rekurzivní funkce. Vhodné je také minimalizovat počet proměnných (nemělo by to ale být na úkor přehlednosti programu).

Kromě optimalizace algoritmů je také důležité optimalizovat přístup k paměti. Výrazného urychlení je možné dosáhnout zajištěním co nejlepší lokality odkazů. Tím je myšlena schopnost programu používat adresy, které jsou blízko sebe (v čase a umístění). Lze toho dosáhnout např. rozdělením používaných datových struktur na více (méně) používané a podle toho pro ně alokovat paměť. Většina platforem se ale liší

ve velikosti použité cache a stránek, proto je tato optimalizace obecně obtížná. Není vhodné předávat velké objemy dat hodnotou, lepší je použití odkazu. Při použití dynamicky alokované paměti je praktické předpovídat potřebné objemy dat a paměť alokovat případně předem.

Výrazné zpomalení programu mohou způsobit časté I/O operace (program musí čekat). To se dá vylepšit pomocí sdružení těchto operací nebo použitím vyrovnávací paměti. Pokud je to možné, je vhodné používat asynchronní I/O operace.

Vysoce užitečné je psát kód s použitím více vláken („Multithreading“), kdykoli je to vzhledem k povaze programu možné. Významně tak lze urychlit zpracování na strojích s více procesory. Tato technika je ale užitečná i na běžných jednoprocesorových strojích – některé současné procesory obsahují vylepšenou podporu pro vícevláknové zpracování (např. technologie Intel HyperThreading©), lze to také využít pro urychlení I/O operací (program může mezitím provádět jiné části kódu) atd. Velkým problémem je ale zatím neexistence univerzálního multiplatformního rozhraní pro vícevláknové zpracování. Nicméně vzhledem k výhodám je vhodné si rozmyslet, jestli to nelze v našem programu využít.

Kapitola 2

Úvod do systému teTEX

teTEX [Tetex] je distribuce TEXu a dalších programů (pdfTEX, Omega, LATEX, dvips, . . .) udržovaná Thomasem Esserem. Základ tvoří distribuce Web2c autorů Karla Berryho a Olafa Webera. Jestliže používáte Linux, pak s nejvyšší pravděpodobností právě teTEX tvoří základ podpory TEXu ve vašem systému. V současné době je aktuální stabilní verzí systém teTEX 2.0.

2.1 Hlavní komponenty

2.1.1 Kpathsea

Kpathsea je základní knihovna pro celou distribuci. Poskytuje aplikacím funkce pro jednoduché a rychlé hierarchické vyhledávání souborů podle jejich jména. Dále jsou s ní spojeny některé další nástroje (kpsewhich, . . .).

2.1.2 Web2c

Web2c je implementace TEXu pro různé platformy (Unix, MS Windows™, Mac, DOS, . . .). Kromě samotného TEXu obsahuje ještě programy METAFONT, METAPOST, BibTEX, nástroje pro konverzi mezi obrazovými formáty, nástroje pro práci s formátem DVI, program pro generování vzorů PatGen a také programy pro generování kódu a dokumentace tangle a weave.

2.1.3 Rozšíření TEXu

Kromě standardního TEXu obsahuje teTEX také další jeho rozšíření. Zejména se jedná o tyto programy: pdfTEX – program pro přímé vytváření souborů ve formátu Adobe™ PDF, eTEX – rozšíření TEXu o další primitiva, Omega (Ω) – TEX s podporou Unicode.

2.1.4 Další podpůrné nástroje a skripty

Zde patří další programy pro práci s formátem DVI – dvips, dvilj, xdvi, dvipdfm. Dále v této distribuci nalezneme spoustu podpůrných skriptů a nástrojů pro dokumentaci, konfiguraci a práci s fonty – programy texdoc, texdoctk, texconfig, allcm, allec, allneeded, fontimport. Podrobný popis těchto nástrojů lze získat pomocí příkazu **texdoc** <**jméno_programu**>.

2.2 Adresářová struktura teTEXu

Stromová struktura teTEXu odpovídá standardu TDS (TEX Directory Structure)[1]. Základní struktura vypadá následovně:

tex/<**formát**>**/**<**balík**>**/** makra

font/<**typ**>**/**<**dodavatel**>**/**<**druh_písma**> fonty

metafont/<**dodavatel**>**/** soubory METAFONTu

doc/<**balík**>**/** dokumentace

source/<**balík**>**/** zdrojové kódy

bibtex/{**bst,bib**}**/**<**balík**>**/** soubory BibTEXu

Proměnné zde znamenají:
<`formát`> — název formátu TEXu (`latex, amstex, ...`)
<`balík`> — název příslušného balíku (`babel, seminar, ...`)
<`typ`> — název typu fontu (`pk, tfm, afm, ...`)
<`dodavatel`> — název dodavatele fontu (`adobe, urw, ...`)
<`druh_písma`> — název druhu písma (`times, cm, ...`)

2.3 Jak vypadají zdrojové kódy

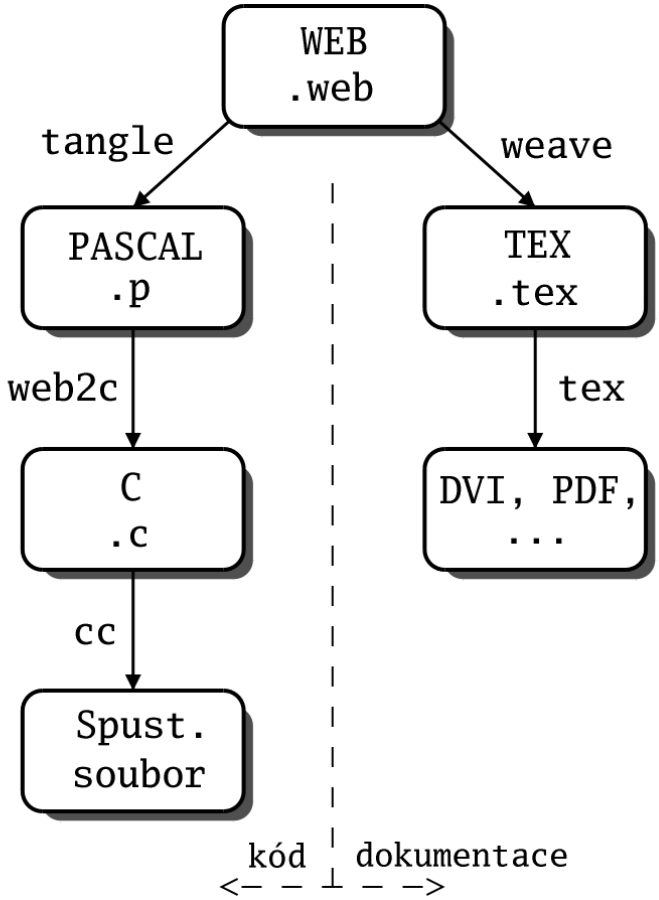
Většina programů z této distribuce je napsána pomocí nástrojů literárního programování. Tento název se používá pro programové nástroje, které v sobě spojují psaní kódu spolu s jeho dokumentací. V případě TEXu se nejvíce používají WEB [Web] nebo CWEB [CWeb]. Z takto napsaného programu se vygeneruje dokumentace ve formátu TEX (pomocí programu weave) a zdrojový kód pomocí programu tangle (ctangle pro CWEB). WEB umožňoval psát a generovat kód v jazyce PASCAL, CWEB používá C nebo C++. Existuje také možnost jednoduše vytvářet patche ve formátu `changefile (.ch)`. Pokud jich je více, lze je pomocí programu tie aplikovat na zdrojový soubor a vytvořit nový aktuální soubor ve formátu WEB. Vygenerovaný kód v PASCALu se nejprve převede do C pomocí nástroje web2c, respektive jeho obslužného skriptu convert. V případě CWEBU se přímo generuje kód v C (C++). Takto vytvořený kód se poté přeloží systémovým překladačem jazyka C. Vygenerovanou dokumentaci lze přeložit pomocí TEXu. Názorně tyto transformace popisuje následující obrázek (2.1).

2.4 Jak se překládá teTEX pod Linuxem

Do dočasného adresáře se rozbalí zdrojové kódy samotného teTEXu (balík `tetex-src`). Potom se spustí **configure**, který vytvoří soubory `Makefile` pro překlad. Lze mu zadat parametry pro úpravu distribuce. Lze zvolit, které balíky se mají překládat (`--without-BALÍK`), umístění souborů (`--prefix=CESTA, --[.*]dir=CESTA`), zda se mají vytvářet spustitelné soubory v adresáři `PREFIX/bin/PLATFORMA/` nebo přímo v adresáři `PREFIX/bin/` a spousta dalších. Poté se příkazem **make** spustí samotný překlad.

Postup kompilace:

1. http://www.tug.org/tds/



Obrázek 2.1: Transformace z formátu WEB

1. Jako první se přeloží vyžadované knihovny (`libwww, libncurses, libz, libpng, libxpdf, libGoo, libt1, libtype1, libtxi`).
2. Poté se přeloží nástroje pro dokumentaci ve formátu `info` a nástroj pro zobrazování dialogů (`dialog`).
3. Následují knihovna a nástroje kpathsea.
4. Nyní se přeloží nástroje web2c. Nejprve samotná aplikace web2c (konvertor PASCALU do C), poté nástroje tie, tangle a ctangle, aplikace pro filtrování dokumentace weave a cweave. Dále následují také ostatní aplikace z tohoto balíku (samotný TEX, eTEX, METAFONT, nástroje pro DVI, pdfTEX, PatGen atd.).
5. Následuje zbytek aplikací z adresáře `texk` (dvips, dvipdfm atd.).

Následuje případná instalace daných souborů. Nejprve se do požadovaného adresáře (`PREFIX/share/`) rozbalí strom `texmf` (z balíku `tetex-texmf`). Poté se provede instalace aplikací teTEXu pomocí příkazu **make install**.

2.5 Konfigurace

Počáteční poinstalační konfigurace se provede příkazem **texconfig confall**. Pro další nastavení lze použít program texconfig (bez parametrů se spustí v interaktivním režimu – využívá knihovnu ncurses).

Konfigurační soubory jsou umístěny v adresáři `texmf/web2c/`. Dva nejdůležitější soubory jsou:

- `texmf.cnf` – zde se konfigurují cesty pro vyhledávání souborů, vstupní soubory TEXu, adresáře s fonty, další důležité cesty a také hlavní nastavení pro různé součásti TEXu (např. limity pro datové struktury)
- `fmtutil.cnf` – zde se nastavují formáty pro dané aplikace (např. se zde dá zapnout podpora pro csTEX, csLATEX atd.).

V případě jakékoli změny v adresářové struktuře souborů je nutné k jejich zavedení spustit příkaz **texhash** (znovu vygeneruje informace v souborech `ls-R`).

Kapitola 3

Optimalizace aplikací distribuce teTEX pomocí kompilace

Z této distribuce byly pro účely testu vybrány aplikace PatGen, pdfTEX, pdfeTEX, TEX, eTEX, Omega, dvipdfm a dále ještě program OPatGen. Tyto programy byly přeloženy s různými volbami překladače na několika strojích a poté byly spuštěny nad testovacími daty. Výsledky prakticky ověřují, jakých optimalizací lze dosáhnout pomocí kompilace.

3.1 Hardware použitý pro testy

Celkem byly využity 4 stroje:

1. Prvním strojem je můj stolní počítač (`angel.local`).

   - HARDWARE:

     CPU: AMD Athlon XP™ 1700+
     Základní deska: MSI KT3V
     Paměť: 256 MB
     HDD: 40 GB 7200 ot./min U–DMA 100

   - SOFTWARE:

     Operační systém: RedHat Linux© 8.0
     Jádro: 2.4.21–rc1
     Překladač C: gcc verze 2.96 (compat–gcc–7.3–2.96.110) a 3.2 (GCC 3.2 20020903 Red Hat Linux 8.0 3.2–7)
     Překladač C++: g++ verze 2.96 (compat–gcc–g++–7.3–2.96.110) a 3.2 (GCC 3.2 20020903 Red Hat Linux 8.0 3.2–7)
     Standardní knihovna C: glibc verze 2.3.2 (Red Hat Linux 8.0 2.3.2–4.80.6)
     Standardní knihovna C++: libstdc++ verze 3.2 (Red Hat Linux 8.0 3.2–7)

2. Druhým strojem je můj přenosný počítač HP Omnibook 6000 (`archangel.local`).

   - HARDWARE:

     CPU: Intel Mobile Pentium III™ 800 MHz
     Základní deska: HP (Intel PIIX4M)
     Paměť: 256 MB
     HDD: 30 GB 5400 ot./min U–DMA 33

   - SOFTWARE:

     Operační systém: RedHat Linux© 8.0
     Jádro: RedHat 2.4.18–24.8.0

Překladač C: gcc verze 2.96 (compat–gcc–7.3–2.96.110) a 3.2 (GCC 3.2 20020903 Red Hat Linux 8.0 3.2–7)
Překladač C++: g++ verze 2.96 (compat–gcc–g++–7.3–2.96.110) a 3.2 (GCC 3.2 20020903 Red Hat Linux 8.0 3.2–7)
Standardní knihovna C: glibc verze 2.3.2 (Red Hat Linux 8.0 2.3.2–4.80.6)
Standardní knihovna C++: libstdc++ verze 3.2 (Red Hat Linux 8.0 3.2–7)

3. Třetím strojem je stolní počítač v kanceláři RNDr. Petra Sojky (`daeron.fi.muni.cz`).

   - HARDWARE:
     CPU: AMD Athlon XP™ 1600+
     Základní deska: MSI KT2 Pro
     Paměť: 512 MB
     HDD: 2 × 60 GB 7200 ot./min U–DMA 100 (software RAID 1)
   - SOFTWARE:
     Operační systém: RedHat Linux© 7.3
     Jádro: RedHat 2.4.18–17.7.x
     Překladač C: gcc verze 2.96
     Překladač C++: g++ verze 2.96
     Standardní knihovna C: glibc verze 2.2.5–42 (Red Hat Linux 7.3)
     Standardní knihovna C++: libstdc++ verze 2.96–110 (Red Hat Linux 7.3)

4. Posledním strojem je server používaný v Laboratoři zpracování přirozeného jazyka NLP (`aurora.fi.muni.cz`).

   - HARDWARE:
     CPU: 2 × Intel Xeon™ HT 2,2 GHz (virtuálně 4 procesory)
     Základní deska: logika Intel
     Paměť: 4 GB
     HDD: více SCSI disků (> 500 GB)
   - SOFTWARE:
     Operační systém: Debian Linux 3.0
     Jádro: 2.4.20–1–pentium4–smp
     Překladač C: gcc verze 2.95.4 a 3.0.4
     Překladač C++: g++ verze 2.95.4 a 3.0.4
     Standardní knihovna C: glibc verze 2.3.1–14 (Debian Linux)
     Standardní knihovna C++: libstdc++ verze 3.0.4–7 (Debian Linux)

3.2 Aplikace vybrané pro testy

Jak již bylo napsáno, pro test byly vybrány programy PatGen, TEX, eTEX, pdfTEX, pdfeTEX, Omega, dvipdfm a OPatGen. Použita byla distribuce teTEX 2.0. Zdrojové kódy těchto aplikací byly získány ze souborů WEB pomocí nástrojů této distribuce. Zároveň s těmito programy byly také z distribuce použity knihovny, které jsou u těchto programů vyžadovány (zejména knihovna `kpathsea`). OPatGen byl stažen z jeho domovské WWW stránky[1], použita byla verze 1.0 (PatLib v. 1.0).

1. http://www.fi.muni.cz/~xantos/patlib/

3.3 Testovací soubory

Pro testování aplikací TEX, eTEX, pdfTEX, pdfeTEX a Omega byl použit zdrojový kód knihy The TEXbook [Texbook], jejíž autorem je Donald E. Knuth. Tento zdrojový kód ve formátu TEX (`texbook.tex`) je možné nalézt např. v distribuci TEXLive.

TEXbook byl použit také pro testování aplikace dvipdfm. Nejdříve byl přeložen ze zdrojového kódu (`texbook.tex`) do formátu DVI (`texbook.dvi`) pomocí aplikace eTEX. Soubor `texbook.dvi` byl poté použit jako vstup pro tuto aplikaci.

Aplikace PatGen a OPatGen byly testovány nad výběrem dat ze souborů, které pro svou diplomovou práci použil David Macháček [Mach]. Z těchto souborů bylo vybráno 25768 rozdělených slov (víceméně náhodné posloupnosti znaků anglické abecedy) délky nejvýše 20 znaků. OPatGen byl spuštěn s parametrem `-u8` (používá UTF–8). Generování bylo provedeno s následujícími parametry:

`hyph_start` = 1
`hyph_finish` = 2
`pat_start` = 1
`pat_finish` = 3
`good weight` = 1
`bad weight` = 1
`threshold` = 1

3.4 Parametry překladů

Pro testování bylo vybráno 35 různých kombinací parametrů překladu. Přehled použitých parametrů je uveden v následující tabulce (tabulka č. 3.1). Podrobnější popis jednotlivých voleb je v sekci Oddíl 1.3.2. Ne všechny testovací stroje ale podporují všechny parametry – kompletní přehledy pro všechny stroje jsou uvedeny v dodatku Příloha A. Jako první se při testu přeloží programy s parametry optimalizace Op–void, která je totožná s Op–0. Je zde za účelem inicializace testu (provede se kompletní překlad a test – nakešují se data atd). Výsledky z této optimalizace se nezahrnují do uvedených charakteristik.

Tabulka 3.1: Parametry překladu

argumenty překladače
–g
–O0
–O1
–O2
–O3
–Os
–mcpu={i386,i686,athlon–xp,pentium3}
–march={i386,i686,athlon–xp,pentium3}
–fomit–frame–pointer
–fexpensive–optimizations
–ffast–math
–funroll–loops
–funroll–all–loops

Pro samotný překlad byly použity překladače **gcc** a **g++**. Testovací stroje používaly každý jiné verze těchto překladačů (celkově byly použity verze 2.95, 2.96, 3.0.4 a 3.2). Konkrétní verze dostupné na daných strojích jsou uvedeny v sekcích Oddíl 3.1 a Příloha A. Program OPatGen bylo nutné překládat verzí 2.9x (verze 3.x zpřísnily podmínky pro zápis šablon, které jsou v OPatGenu hojně využity, a program tudíž na této verzi nelze přeložit).

3.5 Struktura adresáře s testy

Hlavní adresář se soubory a obslužnými skripty vypadá následovně:

`optimization-tests/` adresář obsahující informace o výsledcích kompilací a testů

`optimization-progs/` adresář obsahující přeložené aplikace a testovací data

`programs/` adresář obsahující zdrojové kódy aplikací

`test_all` hlavní skript řídící všechny testy

`do_test` skript, který provede 1 test

`archok_cflags` pomocný skript

`archok_cppflags` pomocný skript

`test_parameters.txt` textový soubor obsahující parametry kompilací

Pro účely testu byly vytvořeny pomocné skripty a upraveny, případně vytvořeny soubory `Makefile` pro překlad. Zdrojové kódy testovaných aplikací nebyly měněny.

3.6 Postup testování

Pro každé nastavení optimalizace byl proveden test skládající se z následujících kroků:

1. Vytvoření adresářů pro výsledky testů a přeložené programy.
2. Vyčištění adresáře se zdrojovými kódy testovaných aplikací (`programs/`) – odstranění všech přeložených objektů.
3. Přeložení požadovaných knihoven v pořadí `kpathsea`, `zlib`, `libpng`, `xpdf`, `lib`.
4. Přeložení aplikací v pořadí PatGen, OPatGen, TEX, eTEX, pdfTEX, pdfeTEX, Omega a dvipdfm.
5. Zkopírování přeložených aplikací do výstupního adresáře a odstranění ladících symbolů (pomocí programu **strip**).
6. Testování aplikací na jejich testovacích datech (viz Oddíl 3.3) v pořadí pdfTEX, pdfeTEX, Omega, TEX, eTEX, dvipdfm, PatGen a OPatGen. Před každým testem byly odstraněny soubory vytvořené v předchozích testech.

Zjišťovanými charakteristikami byly doby běhu překladu programů a následných testů.

Doba běhu jednotlivých testů byla zjišťována pomocí nástroje **time** (externí aplikace, ne součást shellu). Pro výstup bylo použito následující formátování (exportované jako proměnná prostředí):

```
TIME="Command: %C\nExit status: %x\n\nCPU:\nReal\t\t%E\n
User\t\t%U\nSystem\t\t%S\nCPU Percentage\t%P\n\nMemory
(KB):\nMaximum resident set size\t\t%M\nAverage resident
setsize\t\t%t\nAverage total memory use\t\t%K\nAverage size
of unshared data area\t%D\nAverage size of unshared stack
space\t%p\nAverage size of shared text space\t%X\nSystem
page size\t\t%Z\nMajor page faults\t\t%F\nMinor page
faults\t\t%R\nNumber of swapps out of memory\t%W\nNumber of
context-switching\t%c\nNumber of waits\t\t\t%w\n\nI/O:\nNumber
of file system inputs\t\t%I\nNumber of file system
outputs\t\t%O\nNumber of socket messages received\t%r\nNumber
of socket messages sent\t\t%s\nNumber of signals delivered to
process\t%k"
```

Použity byly hodnoty `CPU User`. Kompletní výstupy jsou k dispozici na přiloženém CD (Příloha D).

3.7 Výsledky

Následují zjištěné hodnoty pro všechny testovací stroje. Uvedeny jsou pouze některé statistické hodnoty, kompletní výsledky jsou uvedeny v části Příloha B. Použity jsou následující parametry (t(jméno_optimalizace) představuje dobu běhu daného testu programem přeloženým s parametry optimalizace jméno_optimalizace):

min nejkratší doba běhu

max nejdelší doba běhu

průměr aritmetický průměr všech dob běhu

m. ur. ukazuje, kolik času (v %) je možno ušetřit použitím nejrychlejší optimalizace oproti nejpomalejší (m. ur. = (max – min) / max).

ur. ukazuje, kolik času (v %) je možno ušetřit použitím nejrychlejší optimalizace oproti žádné optimalizaci (ur. = (t(Op–0) – min) / t(Op–0)).

nejv. opt. nejvhodnější optimalizace (t(nejv. opt.) = min) V případě více shod je použita ta s nejkratší dobou překladu.

Z výsledků je patrné, že použitím vhodných parametrů překladu je možné dosáhnout podstatných urychlení. Velice ale záleží na typu programu – nejlépe se dařilo urychlit OPatGen (až o 85 %), naopak aplikaci Omega se podařilo urychlit nejméně (nejvíce o 11 %). Použitím nevhodných parametrů je také možné program dokonce

zpomalit. Velkou komplikací je také fakt, že většina optimalizací několikanásobně prodlouží dobu překladu (v nejhorších případech až 30 ×). Nicméně i přes tuto nevýhodu je evidentní, že má cenu překlady optimalizovat.

Následují shrnutí výsledků a doporučení pro testované stroje. Je třeba poznamenat, že ve většině případů byly u nejvýhodnějších optimalizací minimální rozdíly. V tabulkách jsou uvedeny optimalizace nejvhodnější pro danou aplikaci.

3.7.1 Stroj č. 1 (Athlon XP 1700+)

Výsledky jsou zobrazeny v tabulce 3.2.

Výhodná je optimalizace Op–14. Celkově nejrychlejší byly optimalizace Op–35, Op–14 a Op–32.

Tabulka 3.2: Výsledky ze stroje č. 1

program	doba překladu (s)			doba testu (s)			m. ur.	ur.	nejv.
	min	max	průměr	min	max	průměr	(%)	(%)	opt.
patgen	0.31	1.39	0.86	1.62	5.85	2.36	72	72	Op–19
opatgen	4.57	103.30	60.35	13.14	48.11	21.09	72	72	Op–32
tex	3,65	77,82	24,45	1,19	1,79	1,31	33	28	Op–9
etex	4,22	98,59	29,98	1,24	1,87	1,37	33	29	Op–14
pdftex	8,33	124,48	43,06	2,86	4,99	3,20	42	39	Op–14
pdfetex	5.91	133.18	41.05	2.89	4.95	3.24	41	39	Op–8
omega	7.37	41.86	28.07	28.99	31.97	29.43	9	8	Op–15
dvipdfm	5.22	16.34	12.23	1.52	2.70	1.77	43	41	Op–17

3.7.2 Stroj č. 2 (Intel Mobile Pentium III 800)

Výsledky jsou zobrazeny v tabulce 3.3.

Výhodné jsou optimalizace Op–35, Op–14 a Op–8. Celkově nejrychlejší byly Op–35, Op–14 a Op–32.

Tabulka 3.3: Výsledky ze stroje č. 2

program	doba překladu (s)			doba testu (s)			m. ur.	ur.	nejv.
	min	max	průměr	min	max	průměr	(%)	(%)	opt.
patgen	0.66	2.99	1.78	8.15	27.11	10.75	69	69	Op–18
opatgen	8.63	173.00	110.46	97.35	339.86	153.86	71	71	Op–35
tex	7.38	169.10	48.17	2.38	3.38	2.61	29	28	Op–35
etex	8.60	220.66	58.53	2.49	3.53	2.72	29	26	Op–14
pdftex	17.08	271.34	85.31	5.22	8.52	5.87	38	37	Op–8
pdfetex	12.02	301.59	80.27	5.38	8.63	5.97	37	35	Op–14
omega	14.98	83.86	56.09	38.23	41.33	38.78	7	7	Op–8
dvipdfm	10.48	34.18	25.30	3.04	4.70	3.36	35	33	Op–14

3.7.3 Stroj č. 3 (Athlon XP 1600+)

Výsledky jsou zobrazeny v tabulce 3.4.

Výhodné jsou optimalizace Op–35 a Op–14. Celkově nejrychlejší byly Op–14, Op–35, Op–9 a Op–10.

Tabulka 3.4: Výsledky ze stroje č. 3

program	doba překladu (s)			doba testu (s)			m. ur.	ur.	nejv.
	min	max	průměr	min	max	průměr	(%)	(%)	opt.
patgen	0.34	1.20	0.82	5.91	14.69	7.08	59	59	Op–22
opatgen	4.71	106.03	64.07	44.29	191.99	72.39	76	75	Op–14
tex	4.00	36.93	17.15	1.38	1.99	1.51	30	30	Op–35
etex	4.64	49.73	21.57	1.47	2.02	1.56	27	27	Op–35
pdftex	9.15	63.64	31.79	3.51	5.64	3.82	37	37	Op–14
pdfetex	6.43	68.04	29.81	3.55	5.68	3.88	37	37	Op–14
omega	8.04	32.27	24.07	26.32	29.70	27.20	11	11	Op–12
dvipdfm	5.59	14.32	11.80	1.55	2.64	1.71	41	41	Op–35

3.7.4 Stroj č. 4 (2 × Intel Xeon 2,2)

Výsledky jsou zobrazeny v tabulce 3.5.

Žádná optimalizace není jednoznačně výhodná. Celkově nejrychlejší byly Op–32, Op–9, Op–14 a Op–1.

Tabulka 3.5: Výsledky ze stroje č. 4

program	doba překladu (s)			doba testu (s)			m. ur.	ur.	nejv.
	min	max	průměr	min	max	průměr	(%)	(%)	opt.
patgen	0.25	1.03	0.64	3.77	13.06	4.43	71	71	Op–24
opatgen	2.44	61.61	43.69	27.20	194.54	57.27	86	85	Op–1
tex	2.52	47.10	16.87	1.29	1.77	1.40	27	26	Op–34
etex	3.00	61.65	21.34	1.31	1.90	1.47	31	26	Op–6
pdftex	5.65	73.40	29.10	2.88	4.10	3.09	29	29	Op–8
pdfetex	4.04	80.52	28.83	3.02	4.26	3.21	29	28	Op–32
omega	6.01	23.53	17.29	26.12	28.30	26.49	7	6	Op–9
dvipdfm	3.51	11.26	8.59	1.24	1.79	1.34	30	30	Op–28

3.8 Doporučení

Jak již bylo uvedeno, mezi mnoha optimalizacemi byly minimální rozdíly (je to dáno i tím, že měly hodně společných parametrů). Přesto je možné doporučit vhodné parametry:

- V každém případě by měly být použity parametry `-O2` nebo `-O3`.
- Pokud není potřeba program ladit, je výhodné použít `-fomit-frame-pointer`.

- V některých případech můžeme program urychlit i volbou dané platformy pomocí `-march=platforma` nebo `-mcpu=platforma`.
- Ve specifických případech (některé typy výpočtů) by urychlení mohlo přinést použití parametru `-ffast-math`.
- Naopak nebyl prokázán výraznější vliv samotné verze použitého překladače, kromě toho, že nové verze přinášejí nové volby. Bohužel ale nebyly k dispozici jiné překladače než různé verze GCC.

Kapitola 4

Profilování

Jak již bylo nastíněno v úvodu, vytvoření a analýza profilu aplikace je základním a nezbytným krokem před každou optimalizací. Pomocí profilování je možné určit, kde program strávil nejvíce času a jak je vzájemně provázáno volání funkcí. Pro tuto kapitolu byl zvolen program gprof.

4.1 Vytvoření a analýza profilu

Profil dané aplikace se vytváří v následujících 2 krocích (uvedené parametry platí pro překladače z GCC:

1. Přeložení a „slinkování“ programu s podporou pro profilování (`-pg` pro gprof) a případně s podporou pro ladění (`-g`).

2. Spuštění programu nad testovanými daty. Data a argumenty by měly být vhodně zvoleny, protože mají přímý vliv na vygenerovaný profil. Program po ukončení běhu vytvoří soubor s profilem (obvykle `gmon.out`).

Vytvořený profil se poté analyzuje pomocí profilovacího programu (**gprof**). Je možné vygenerovat následující výstupy:

- *„Přímý“ profil (flat profile)* – zobrazí celkový čas, který program strávil v každé funkci a kolikrát byla tato funkce volána.

  Syntaxe pro jeho vytvoření je:

  gprof -p <**program**>

- *Graf funkcí (call graph)* – pro každou funkci zobrazí, kolik času strávil program v této funkci a ve funkcích, které jsou v ní volány. Toto je užitečné, když potřebujeme najít funkce, které samy neběží dlouho, ale volají časově náročné funkce.

  Syntaxe pro jeho vytvoření je:

  gprof -q <**program**>

- *Komentovaný kód (annotated source)* – zobrazí zdrojový kód programu a ke každé funkci zobrazí, kolikrát byla volána. Je také možné zobrazit, kolikrát byl proveden každý řádek kódu.

  Syntaxe pro jeho vytvoření je:

  gprof -A <**program**>

4.2 Ukázky profilování

Pro názornou ilustraci výsledků profilování byly vybrány programy PatGen a OPatGen. Jako data a parametry byly použity stejné hodnoty jako pro testy v předchozí kapitole (viz Oddíl 3.3).

4.2.1 Profily aplikace PatGen

Ukázán je pouze „Přímý“ profil, zbylé dva jsou příliš dlouhé na to, aby byly obsaženy přímo v této práci. Jsou ale k dispozici na přiloženém CD (Příloha D).

- „Přímý“ profil:

  Kompletní výpis je uveden v části Příloha C.

  Při použití optimalizace Op–0 se nejvíce času (90,7 %) stráví ve funkci `readword`. Druhá nejpoužívanější (5,3 %) funkce je `hyphenate`. Zbytek má méně než 2 %.

  Funkce `readword` se stará o načtení a zpracování řádky vstupního textu. Je tedy evidentní, že vstup-výstupní operace zabírají více než 90 % celkového času.

  Při použití optimalizace Op–3 se podíl funkce `readword` zmenší na 84,6 %, podíly ostatních vzrostou (funkce `hyphenate` nyní zabere 8,2 %).

4.2.2 Profily aplikace OPatGen

Bohužel všechny profily jsou příliš velké na přímé vložení do textu této práce. Jsou ale obsaženy na přiloženém CD (Příloha D).

- „Přímý“ profil:

  V případě OPatGenu není rozdělení celkové doby běhu tak jednoznačné, žádná nepřekračuje 6 %. Při použití optimalizace Op–0 se nejvíce času (5,2 %) stráví ve funkci `Competitive_multi_out_pat_manip::competitive_word_output`, druhá nejpoužívanější funkce (4,65 %) je `vector::_M_insert_aux`.

  Při použití optimalizace Op–3 se tyto podíly výrazným způsobem změní. Funkce `Competitive_multi_out_pat_manip::competitive_word_output` nyní zabere 24,6 %, funkce `vector::_M_insert_aux` 13,1 %. Ze zbylých funkcí nezabírá žádná více než 5 %.

- Z grafu funkcí je patrné, že se příliš často zbytečně prochází několika funkcemi (zavolá se funkce, která pouze zavolá jinou funkci).

Kapitola 5

Generování vzorů dělení slov

Technika vzorů je jednou z nejefektivnějších metod pro uchovávání důležitých informací o datech a jejich následné rozpoznávání. Ukázala se být obzvláště výhodnou pro ukládání a rozpoznávání vhodných míst pro dělení slov. V tomto smyslu je implementována nejen v TEXu, ale i v mnoha jiných (i komerčních) produktech. Přestože má tato metoda podstatně širší uplatnění, zmíněno bude pouze toto její uplatnění.

5.1 Vzory

Vzor je řetězec znaků nad stejnou abecedou, která je použita ve zpracovávaných textech (vzory jsou podřetězce slov z těchto textů). Vzory se používají pro popis pozic v textu s danou vlastností. To může být například pozice vhodná pro rozdělení slova. Informace spojená s touto pozicí určuje, jestli je vhodná (*pokrývací* vzor) nebo není (*zabraňující* vzor). Většinou se pro reprezentaci této informace používají přirozená čísla – sudá pro pokrývací, lichá pro zabraňující. Kromě toho se ještě používají i další symboly (např. symbol „.“ pro označení začátku nebo konce slova). Vzory jsou např. „1na“, „la2p“, „i2hl“.

Aplikace vzorů na dané slovo probíhá podle následujícího postupu:

1. Pro všechna podslova zadaného slova se vyhledají všechny odpovídající vzory.
2. Vzory spolu následně „soutěží“ – pokud se vzory na některé pozici v hodnotách překrývají, vyhrává ten s vyšší hodnotou (pokud danou pozici nepokrývá žádný vzor, je její hodnota 0).
3. Jako vhodná pro rozdělení slova jsou pak ta místa, která mají nejvyšší liché hodnoty.

Výše uvedený postup je názorně ukázán na následujícím příkladu.

```
.h y p h e n a t i o n.   slovo
          1n a
              1t i
o n2a t
                2i o      vzory
       h e2n
.h y3p h
       h e n a4
       h e n5a t
.h0y3p0h0e2n5a4t2i0o0n.   vítězný vzor
 h y-p h e n-a t i o n    vhodná místa pro dělení slova
```

5.2 Generování vzorů

Pro generování vzorů potřebujeme dostatečně velkou množinu vstupních dat, ve kterých jsou již označena místa, která nás zajímají.

Tato data se poté několikrát procházejí. Při lichých průchodech se hledají pokrývací vzory, při sudých zabraňující. Při každém průchodu se nejprve vyberou vhodní kandidáti na vzory podle zadaného pravidla (např. všechny podřetězce slov obsahující požadovanou pozici délky nejvýše x symbolů). Z těchto kandidátů se podle vhodného pravidla vyberou vzory. Takovým pravidlem je například jejich otestování na datech, ze kterých byli vybráni (určí se počty, kolikrát určili pozici dobře, kolikrát špatně a zvolí se přijatelná hodnota).

Na tomto principu pracují oba programy uvedené v následující části.

5.3 Přehled existujících programů pro generování vzorů

Prvním programem řešícím generování vzorů dělení slov pro TEX byl PatGen. Pokusem o jeho nahrazení je pak program OPatGen [Antos]. Následují jejich stručné charakteristiky:

5.3.1 PatGen

Autorem PatGenu je Franklin Mark Liang [Liang]. Vzhledem k době vzniku tohoto programu (počátek 80. let) a požadavkům pro jeho vznik je zřejmé, že tento program již nevyhovuje dnešním potřebám. Je napsán v systému WEB. Mezi jeho hlavní nedostatky patří:

- Podporuje pouze osmibitové kódování (není navíc možné použít všechny symboly, některé mají speciální význam).
- Je to monolitický program (není jednoznačně rozdělen na funkční bloky).
- Je vysoce optimalizován na úkor přehlednosti a funkčnosti. Tyto úpravy ale byly nutné, aby mohl běžet na stroji PDP–10, což bylo požadavkem.
- Používá statické datové struktury. Pokud se tyto struktury zaplní, program se ukončí a je nutno pozměnit zdrojový kód a znova program přeložit.
- Nejvyšší počet úrovní (průchodů) je 9.
- Jakékoli úpravy zdrojového kódu (např. přidání podpory pro UNICODE) by vyžadovalo obrovské úsilí a změnu velké části zdrojového kódu.
- Je použitelný pouze pro generování vzorů dělení slov.

Naopak jeho hlavní výhodou je rychlost.

5.3.2 OPatGen

OPatGen vznikl v rámci diplomové práce na Fakultě informatiky Masarykovy univerzity [Antos]. Jeho autorem je David Antoš. Základním požadavkem bylo vytvořit víceúčelovou knihovnu pro manipulaci se vzory. Výsledkem je knihovna PatLib. OPatGen je pouze aplikací této knihovny na generování vzorů dělení slov. PatLib i OPatGen

jsou napsány v systému CWEB, jako programovací jazyk je použit C++. Jako datové struktury jsou použity zhuštěné digitální vyhledávací stromy (packed trie).

Mezi hlavní výhody OPatGenu patří:

- Podporuje UNICODE (konkrétně kódování UTF–8).
- Knihovna PatLib nabízí široké možnosti použití.
- Používá dynamické struktury (omezení je tedy až velikost operační paměti).
- Má čistý objektový strukturovaný návrh.
- Knihovna PatLib je napsána s maximálním využitím šablon, což umožňuje flexibilnější definice typů.
- Možnost použít libovolnou metodu výběru kandidátů.

I OPatGen má ale několik nevýhod, mezi něž patří:

- Rychlost. Překlad trvá v průměru 100 × déle než překlad PatGenu. Doba generování vzorů ze stejných dat je v průměru 10 × delší než v případě PatGenu.
- Návrh je až příliš „přeobjektovaný“ a transparentní. Z výsledků profilování je patrné, že granularita návrhu jednotlivých tříd a částí je příliš jemná.
- Alokace paměti je řešena sice nejjednodušším, ale také ne zrovna nejvhodnějším způsobem. Paměť pro datové struktury se alokuje pomocí třídy (šablony) `Growing_array` (používá kontejner `vector`). Pokud se objeví požadavek na neexistující prvek, vyvolá se funkce `push_back` (součást kontejneru `vector`). Při každé alokaci paměti se tedy musí projít několika třídami a zavolat tuto funkci. To je sice transparentní řešení, ale z hlediska rychlosti to není optimální.
- Použití šablon přináší mnohé výhody, ale i problémy. Program je díky nim méně čitelný (i přes kvalitní dokumentaci) a jsou problémy s přenositelností (program byl testován a přeložen pomocí překladače g++ verze 2.96, na novějších verzích řady 3 jej není možné přeložit). Problém přináší standardní šablonová knihovna C++ STL, jejíž implementace v překladačích není jednotná a přesná.
- Implementace byla provedena spíše s ohledem na funkčnost a použitelnost. Optimalizace je obtížná (např. přílišná provázanost některých tříd brání jednoduché úpravě datových struktur).

5.4 UPatGen

Vzhledem k nevýhodám obou programů zmíněných v předchozí kapitole bylo rozhodnuto analyzovat dané problémy a navrhnout vhodné řešení. Výsledný návrh dostal pracovní jméno UPatGen (Unicode PATtern GENerator).

5.4.1 Hlavní cíle

- Dostatečná obecnost (využití i pro jiné účely než jen generování vzorů dělení slov).
- Modulární struktura.
- Podpora pro UNICODE.
- Přijatelná rychlost.
- Stejné možnosti jako OPatGen.
- Kvalitní dokumentace.
- Použití vhodné datové struktury.
- Přenositelnost a stabilita.
- Struktura umožňující případné úpravy.
- Kompatibilita s PatGenem a OPatGenem.

5.4.2 Návrh

Jako programovací jazyk byl zvolen jazyk C, hlavně z důvodů široké podpory, kompatibility a dlouhodobé stability.

Pro dokumentaci byl vybrán systém Doxygen[1]. Umožňuje psát dokumentaci jako speciálně formátované komentáře přímo ve zdrojovém kódu programu. Poté je možné z těchto kódů vygenerovat různé formáty dokumentů (HTML, XML, LaTeX, ...).

Jako jazyk pro dokumentaci byla zvolena angličtina.

Jako hlavní datová struktura byla vybrána knihovna Judy [Judy], konkrétně její varianta JudySL. Je to vlastně digitální strom na principu trie. Indexem je řetězec, výstupem je číslo. Knihovna a rozhraní je napsáno v jazyce C. Hlavní výhodou je vysoká optimalizace na rychlost, což kompenzuje i drobné komplikace při práci s ní (např. její dynamičnost).

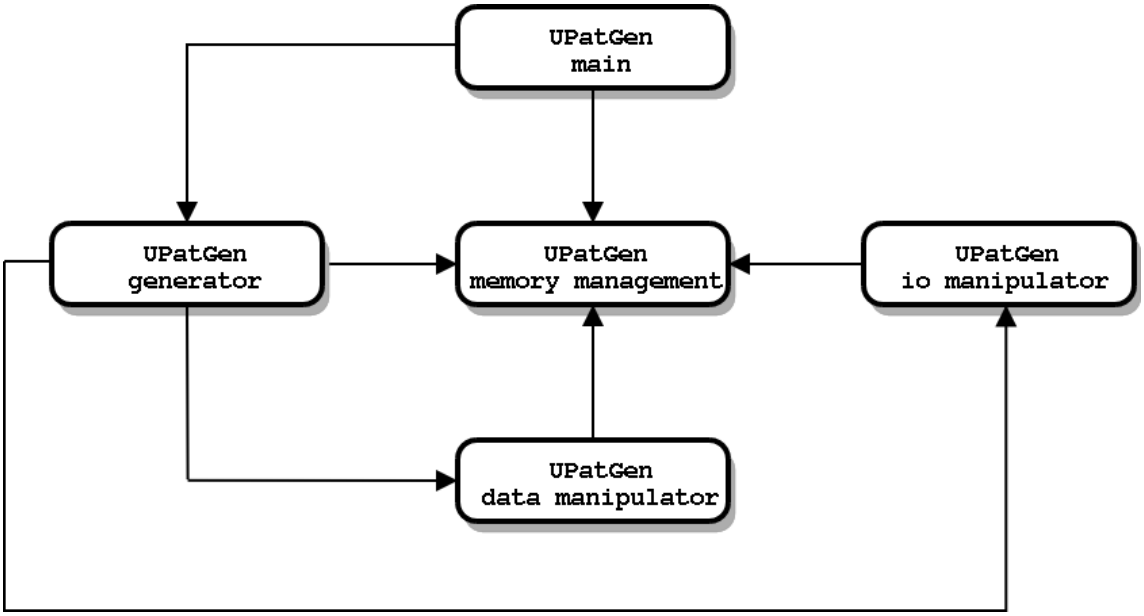
Jako vstupně-výstupní kódování bylo rozhodnuto použít kódování UTF–8 vzhledem k nejvyšší efektivitě ze všech schémat definovaných ve standardu UNICODE. Je také nejvhodnější vzhledem k požadavku kompatibility s programem PatGen. Bylo rozhodnuto neprovádět překlad znaků z UTF–8 do žádné vnitřní reprezentace, ale pracovat přímo v UTF–8. To přináší některé podstatné výhody:

- Nemusí se provádět žádný překlad, pouze kontrola správnosti vstupu (vícebajtových sekvencí).
- Reprezentace řetězců jako pole typu char by mělo být efektivnější než pole typu int. Také práce s touto reprezentací je jednodušší (hlavně přístup k indexu pole Judy).

Strukturní členění programu je zobrazeno na následujícím obrázku (5.1).

Každá část zde uvedená tvoří samostatný modul. Každý lze samostatně přeložit, výsledná aplikace vznikne slinkováním těchto modulů. Rozhraní je definováno pro

1. http://www.stack.nl/~dimitri/doxygen/



Obrázek 5.1: UPatGen – schéma

každý modul v samostatném hlavičkovém souboru. Moduly nesmí přistupovat k jiným funkcím než těm, které jsou exportovány.

Zobrazené části mají tyto funkce:

- *UPatGen main* – hlavní program. Slouží k načítání argumentů a spouštění generování.
- *UPatGen memory management* – základ používaný všemi ostatními částmi. Obsahuje funkce pro správu dynamické paměti.
- *UPatGen generator* – zde jsou definovány funkce pro generování vzorů.
- *UPatGen io manipulator* – Řeší vstupně-výstupní operace. Obsahuje funkce pro kontrolu správnosti vstupů (UTF–8).
- *UPatGen data manipulator* – obsahuje funkce pro přístup k datovým strukturám.

5.4.3 Implementace

K implementaci výše uvedeného programu bylo přistoupeno až dodatečně a nebylo to cílem této práce, proto tento program není úplně dokončen. Nicméně byly implementovány některé jednotky a byla otestována funkčnost předchozího návrhu. Drobné komplikace nastaly s knihovnou Judy.

Jedním z hlavních požadavků na tento program byla rychlost. K dosažení co nejvyšší rychlosti by měly vést následující rozhodnutí:

- Použití knihovny Judy (je rychlejší než packed trie použité v OPatGenu).
- Modulární návrh programu v jazyce C zajišťuje, že v programu nebudou funkce vzájemně příliš provázány (problém šablon v OPatGenu).

- Každý modul si alokuje paměť podle svých potřeb, odkazy na tuto paměť se poté ukládají do speciálních polí pomocí funkcí z části `UPatGen memory management`. Tím je zajištěno jednoduché uvolnění veškeré dynamicky alokované paměti (stačí projít všechna tato pole).

- Program neprovádí překlad z a do vnitřní reprezentace, ale pracuje přímo s řetězci v UTF–8. To by mělo přinést další zrychlení oproti PatGenu a OPatGenu.

Díky výše uvedeným vlastnostem je očekávána rychlost UPatGenu na úrovni programu PatGen.

Zdrojové kódy a dokumentace ve formátu HTML jsou k dispozici na přiloženém CD (Příloha D).

Na tomto programu budu dále pracovat a doufám, že ho dovedu k plně funkční verzi.

Kapitola 6

Závěr

Je zřejmé, že i v dnešní době překotného vývoje počítačové techniky má cenu programy optimalizovat. Přestože je toto nepostradatelné pro aplikace kriticky závislé na výkonu, je vhodné myslet na to i při psaní běžných programů.

Uvedeny byly základní metody pro optimalizaci programů. Podrobně popsána byla metoda optimalizace pomocí překladu. Tato metoda byla také použita pro následné testování.

teTEX je jednou z nejpopulárnějších distribucí TEXu, proto se zdál být vhodným adeptem pro praktické ověření vlivu vybraných optimalizací. Tyto testy prokázaly, že optimalizace překladu u většiny těchto aplikací má rozhodně smysl (dosažené zrychlení oproti neoptimalizované verzi se pohybovalo mezi 6 % až 85 % v závislosti na použitém stroji a programu). Nejlépe se dařilo urychlit programy OPatGen (zrychlení o 71–85 %) a PatGen (zrychlení o 59–72 %), nejhůře na tom byl program Omega (zrychlení o 6–11 %). U ostatních programů bylo dosaženo zrychlení v rozmezí 20 % až 40 %. Nejvýraznější vliv na výslednou rychlost mělo použití základních optimalizačních parametrů (`-O1`, `-O2` a `-O3`).

Zároveň bylo provedeno porovnání rychlostí daných aplikací na stejných datech. Velice zajímavý je rozdíl mezi programy TEX a Omega (rozdíl v rychlosti je zhruba dvacetinásobný ve prospěch programu TEX). Výrazný rozdíl v rychlosti je také mezi programy PatGen a OPatGen (OPatGen je zhruba desetkrát pomalejší než PatGen). Z porovnání rychlostí programů pro generování do formátu PDF (pdfTEX a kombinace TEX a dvipdfm) vyšlo najevo, že časově jsou oba tato řešení srovnatelná.

Pokud by se měly tyto programy dále optimalizovat, bylo by nutné přímo pozměnit zdrojové kódy těchto programů.

Podrobně analyzovány a testovány byly programy pro generování vzorů. Testy prokázaly výraznou degradaci rychlosti generování vzorů v případě použití programu OPatGen (oproti programu PatGen). Na základě provedené analýzy bylo rozhodnuto neprovádět optimalizaci tohoto programu (bylo by to příliš komplikované a výsledek by byl značně nejistý), ale navrhnout novou aplikaci. Hlavním požadavkem při tomto návrhu byla rychlost. Tento návrh dostal pracovní název UPatGen. Byly implementovány některé jeho části pro účely prvních předběžných testů. Z nich vyplývá, že rychlost tohoto programu by se měla pohybovat přibližně na úrovni programu PatGen (měl by tedy být několikanásobně rychlejší než OPatGen). Na vývoji tohoto programu budu dále pracovat.

Příloha A

Parametry překladu

Konkrétní hodnoty jsou uvedeny v následujících tabulkách:

- Parametry překladu pro stroj č.1 jsou uvedeny v tabulce A.1. Optimalizace používají překladače gcc a g++ následujících verzí:

  Op–void – Op–24, Op–35 verze 3.2

  Op–25 – Op–34 verze 2.96

- Parametry překladu pro stroj č.2 jsou uvedeny v tabulce A.2. Optimalizace používají překladače gcc a g++ následujících verzí:

  Op–void – Op–24, Op–35 verze 3.2

  Op–25 – Op–34 verze 2.96

- Parametry překladu pro stroj č.3 jsou uvedeny v tabulce A.3. Optimalizace používají překladače gcc a g++ verze 2.96.

- Parametry překladu pro stroj č.4 jsou uvedeny v tabulce A.4. Optimalizace používají překladače gcc a g++ následujících verzí:

  Op–void – Op–24, Op–35 verze 3.0.4

  Op–25 – Op–34 verze 2.95

Tabulka A.1: Parametry překladu pro stroj č. 1 (Athlon XP 1700+)

optimalizace	argumenty překladu
Op–void	–g –O0
Op–0	–g –O0
Op–1	–g –O1
Op–2	–g –O2
Op–3	–g –O3
Op–4	–g –Os
Op–5	–g –O2 –march=i386 –mcpu=i386
Op–6	–g –O2 –march=i686 –mcpu=i686
Op–7	–g –O2 –march=athlon–xp –mcpu=athlon–xp
Op–8	–g –O2 –march=athlon–xp –mcpu=athlon–xp –fomit–frame–pointer
Op–9	–g –O2 –march=athlon–xp –mcpu=athlon–xp –fexpensive–optimizations
Op–10	–g –O2 –march=athlon–xp –mcpu=athlon–xp –fexpensive–optimizations –ffast–math
Op–11	–g –O3 –march=i386 –mcpu=i386
Op–12	–g –O3 –march=i686 –mcpu=i686
Op–13	–g –O3 –march=athlon–xp –mcpu=athlon–xp
Op–14	–g –O3 –march=athlon–xp –mcpu=athlon–xp –fomit–frame–pointer
Op–15	–g –O3 –march=athlon–xp –mcpu=athlon–xp –fexpensive–optimizations
Op–16	–g –O3 –march=athlon–xp –mcpu=athlon–xp –fexpensive–optimizations –ffast–math
Op–17	–g –O3 –march=athlon–xp –mcpu=athlon–xp –fexpensive–optimizations –ffast–math –funroll–loops
Op–18	–g –O3 –march=athlon–xp –mcpu=athlon–xp –fexpensive–optimizations –ffast–math –funroll–all–loops
Op–19	–g –O3 –funroll–loops
Op–20	–g –O3 –funroll–all–loops
Op–21	–g –O3 –march=i386 –mcpu=i386 –funroll–loops
Op–22	–g –O3 –march=i386 –mcpu=i386 –funroll–all–loops
Op–23	–g –O3 –march=i386 –mcpu=i386 –ffast–math –funroll–loops
Op–24	–g –O3 –march=i386 –mcpu=i386 –ffast–math –funroll–all–loops
Op–25	–g –O0
Op–26	–g –O1
Op–27	–g –O2
Op–28	–g –O3
Op–29	–g –Os
Op–30	–g –O2 –march=i386 –mcpu=i386
Op–31	–g –O2 –march=i686 –mcpu=i686
Op–32	–g –O3 –march=i686 –mcpu=i686 –fomit–frame–pointer
Op–33	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations
Op–34	–g –O3 –march=i686 –mcpu=i686 –funroll–loops
Op–35	–O3 –march=athlon–xp –mcpu=athlon–xp –ffast–math –funroll–loops –fomit–frame–pointer –fexpensive–optimizations

Tabulka A.2: Parametry překladu pro stroj č. 2 (Intel Mobile Pentium III 800)

optimalizace	argumenty překladu
Op–void	–g –O0
Op–0	–g –O0
Op–1	–g –O1
Op–2	–g –O2
Op–3	–g –O3
Op–4	–g –Os
Op–5	–g –O2 –march=i386 –mcpu=i386
Op–6	–g –O2 –march=i686 –mcpu=i686
Op–7	–g –O2 –march=pentium3 –mcpu=pentium3
Op–8	–g –O2 –march=pentium3 –mcpu=pentium3 –fomit–frame–pointer
Op–9	–g –O2 –march=pentium3 –mcpu=pentium3 –fexpensive–optimizations
Op–10	–g –O2 –march=pentium3 –mcpu=pentium3 –fexpensive–optimizations –ffast–math
Op–11	–g –O3 –march=i386 –mcpu=i386
Op–12	–g –O3 –march=i686 –mcpu=i686
Op–13	–g –O3 –march=pentium3 –mcpu=pentium3
Op–14	–g –O3 –march=pentium3 –mcpu=pentium3 –fomit–frame–pointer
Op–15	–g –O3 –march=pentium3 –mcpu=pentium3 –fexpensive–optimizations
Op–16	–g –O3 –march=pentium3 –mcpu=pentium3 –fexpensive–optimizations –ffast–math
Op–17	–g –O3 –march=pentium3 –mcpu=pentium3 –fexpensive–optimizations –ffast–math –funroll–loops
Op–18	–g –O3 –march=pentium3 –mcpu=pentium3 –fexpensive–optimizations –ffast–math –funroll–all–loops
Op–19	–g –O3 –funroll–loops
Op–20	–g –O3 –funroll–all–loops
Op–21	–g –O3 –march=i386 –mcpu=i386 –funroll–loops
Op–22	–g –O3 –march=i386 –mcpu=i386 –funroll–all–loops
Op–23	–g –O3 –march=i386 –mcpu=i386 –ffast–math –funroll–loops
Op–24	–g –O3 –march=i386 –mcpu=i386 –ffast–math –funroll–all–loops
Op–25	–g –O0
Op–26	–g –O1
Op–27	–g –O2
Op–28	–g –O3
Op–29	–g –Os
Op–30	–g –O2 –march=i386 –mcpu=i386
Op–31	–g –O2 –march=i686 –mcpu=i686
Op–32	–g –O3 –march=i686 –mcpu=i686 –fomit–frame–pointer
Op–33	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations
Op–34	–g –O3 –march=i686 –mcpu=i686 –funroll–loops
Op–35	–O3 –march=pentium3 –mcpu=pentium3 –ffast–math –funroll–loops –fomit–frame–pointer –fexpensive–optimizations

Tabulka A.3: Parametry překladu pro stroj č. 3 (Athlon XP 1600+)

optimalizace	argumenty překladu
Op–void	–g –O0
Op–0	–g –O0
Op–1	–g –O1
Op–2	–g –O2
Op–3	–g –O3
Op–4	–g –Os
Op–5	–g –O2 –march=i386 –mcpu=i386
Op–6	–g –O2 –march=i686 –mcpu=i686
Op–7	–g –O2 –march=i686 –mcpu=i686
Op–8	–g –O2 –march=i686 –mcpu=i686 –fomit–frame–pointer
Op–9	–g –O2 –march=i686 –mcpu=i686 –fexpensive–optimizations
Op–10	–g –O2 –march=i686 –mcpu=i686 –fexpensive–optimizations –ffast–math
Op–11	–g –O3 –march=i386 –mcpu=i386
Op–12	–g –O3 –march=i686 –mcpu=i686
Op–14	–g –O3 –march=i686 –mcpu=i686 –fomit–frame–pointer
Op–15	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations
Op–16	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations –ffast–math
Op–17	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations –ffast–math –funroll–loops
Op–18	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations –ffast–math –funroll–all–loops
Op–19	–g –O3 –funroll–loops
Op–20	–g –O3 –funroll–all–loops
Op–21	–g –O3 –march=i386 –mcpu=i386 –funroll–loops
Op–22	–g –O3 –march=i386 –mcpu=i386 –funroll–all–loops
Op–23	–g –O3 –march=i386 –mcpu=i386 –ffast–math –funroll–loops
Op–24	–g –O3 –march=i386 –mcpu=i386 –ffast–math –funroll–all–loops
Op–35	–O3 –march=i686 –mcpu=i686 –ffast–math –funroll–loops –fomit–frame–pointer –fexpensive–optimizations

Tabulka A.4: Parametry překladu pro stroj č. 4 (2 × Intel Xeon)

optimalizace	argumenty překladu
Op–void	–g –O0
Op–0	–g –O0
Op–1	–g –O1
Op–2	–g –O2
Op–3	–g –O3
Op–4	–g –Os
Op–5	–g –O2 –march=i386 –mcpu=i386
Op–6	–g –O2 –march=i686 –mcpu=i686
Op–8	–g –O2 –march=i686 –mcpu=i686 –fomit–frame–pointer
Op–9	–g –O2 –march=i686 –mcpu=i686 –fexpensive–optimizations
Op–10	–g –O2 –march=i686 –mcpu=i686 –fexpensive–optimizations –ffast–math
Op–11	–g –O3 –march=i386 –mcpu=i386
Op–12	–g –O3 –march=i686 –mcpu=i686
Op–13	–g –O3 –march=i686 –mcpu=i686
Op–14	–g –O3 –march=i686 –mcpu=i686 –fomit–frame–pointer
Op–15	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations
Op–16	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations –ffast–math
Op–17	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations –ffast–math –funroll–loops
Op–18	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations –ffast–math –funroll–all–loops
Op–19	–g –O3 –funroll–loops
Op–20	–g –O3 –funroll–all–loops
Op–21	–g –O3 –march=i386 –mcpu=i386 –funroll–loops
Op–22	–g –O3 –march=i386 –mcpu=i386 –funroll–all–loops
Op–23	–g –O3 –march=i386 –mcpu=i386 –ffast–math –funroll–loops
Op–24	–g –O3 –march=i386 –mcpu=i386 –ffast–math –funroll–all–loops
Op–25	–g –O0
Op–26	–g –O1
Op–27	–g –O2
Op–28	–g –O3
Op–29	–g –Os
Op–30	–g –O2 –march=i386 –mcpu=i386
Op–31	–g –O2 –march=i686 –mcpu=i686
Op–32	–g –O3 –march=i686 –mcpu=i686 –fomit–frame–pointer
Op–33	–g –O3 –march=i686 –mcpu=i686 –fexpensive–optimizations
Op–34	–g –O3 –march=i686 –mcpu=i686 –funroll–loops
Op–35	–O3 –march=i686 –mcpu=i686 –ffast–math –funroll–loops –fomit–frame–pointer –fexpensive–optimizations

Příloha B

Kompletní výsledky testů

Konkrétní hodnoty jsou uvedeny v následujících tabulkách:

Stroj č. 1:

- Doby překladů jednotlivých programů jsou uvedeny v tabulce B.1.
- Doby testů jednotlivých programů jsou uvedeny v tabulce B.2.
- Velikosti přeložených programů jsou uvedeny v tabulce B.3.

Stroj č. 2:

- Doby překladů jednotlivých programů jsou uvedeny v tabulce B.4.
- Doby testů jednotlivých programů jsou uvedeny v tabulce B.5.
- Velikosti přeložených programů jsou uvedeny v tabulce B.6.

Stroj č. 3:

- Doby překladů jednotlivých programů jsou uvedeny v tabulce B.7.
- Doby testů jednotlivých programů jsou uvedeny v tabulce B.8.
- Velikosti přeložených programů jsou uvedeny v tabulce B.9.

Stroj č. 4:

- Doby překladů jednotlivých programů jsou uvedeny v tabulce B.10.
- Doby testů jednotlivých programů jsou uvedeny v tabulce B.11.
- Velikosti přeložených programů jsou uvedeny v tabulce B.12.

Tabulka B.1: Doby překladů (s) pro stroj č. 1 (Athlon XP 1700+)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	0.37	4.92	4.22	4.96	9.78	7.15	8.66	5.64
Op–1	0.53	44.43	6.88	8.16	15.02	11.61	14.19	8.51
Op–2	0.80	71.13	11.43	13.47	22.99	18.90	25.62	11.80
Op–3	0.91	72.49	34.50	42.25	59.14	57.88	36.06	14.06
Op–4	0.74	103.30	9.93	11.90	19.77	16.84	23.02	9.69
Op–5	0.79	4.57	11.38	13.45	22.38	18.88	25.62	11.77
Op–6	0.83	4.61	10.95	12.95	21.40	18.09	25.45	11.37
Op–7	0.86	73.06	11.04	13.04	21.96	18.27	25.28	11.66
Op–8	0.82	4.75	10.89	12.85	21.32	17.74	24.98	11.59
Op–9	0.85	73.12	10.99	13.02	22.14	18.16	25.24	11.61
Op–10	0.86	72.14	11.16	13.15	22.35	18.28	25.33	12.08
Op–11	0.90	72.68	34.61	42.34	59.12	57.81	36.15	13.97
Op–12	0.90	73.43	32.94	40.29	55.98	55.21	35.02	13.89
Op–13	0.91	73.32	32.88	40.00	56.08	54.87	34.55	14.04
Op–14	0.90	75.80	32.69	39.92	55.43	54.54	34.18	13.76
Op–15	0.92	73.50	32.88	40.14	56.07	54.80	34.63	14.03
Op–16	0.93	73.50	33.11	40.42	56.18	55.39	34.83	14.43
Op–17	1.13	73.15	34.41	42.10	61.17	57.69	35.66	15.19
Op–18	1.39	77.75	77.82	98.59	124.48	133.18	40.31	16.34
Op–19	1.04	72.65	35.90	43.70	63.73	60.15	36.94	14.93
Op–20	1.33	77.31	61.01	75.78	100.23	102.53	41.64	15.93
Op–21	1.05	72.58	35.85	43.90	63.72	60.41	37.06	14.82
Op–22	1.32	77.53	60.93	76.01	100.27	102.44	41.69	15.88
Op–23	1.10	72.96	35.93	44.10	63.91	60.65	37.10	15.23
Op–24	1.33	77.68	61.49	76.35	100.96	102.86	41.86	16.19
Op–25	0.31	4.85	3.65	4.22	8.33	5.91	7.37	5.22
Op–26	0.52	44.20	5.67	6.44	12.44	8.91	11.57	7.76
Op–27	0.69	70.99	8.65	10.09	17.84	14.01	19.16	10.10
Op–28	0.73	72.45	16.60	20.81	30.66	28.55	24.35	10.85
Op–29	0.65	103.21	8.42	9.82	16.97	13.79	19.53	8.88
Op–30	0.67	4.63	8.62	10.12	17.56	14.14	19.22	10.38
Op–31	0.74	4.62	8.75	10.15	17.63	14.25	19.52	10.53
Op–32	0.76	75.72	16.55	20.64	30.45	28.40	24.28	11.12
Op–33	0.78	73.33	16.75	20.86	30.72	28.88	24.59	10.93
Op–34	0.81	73.14	17.42	21.88	32.58	29.78	25.11	11.80
Op–35	1.08	73.14	33.65	41.53	59.45	56.89	34.91	14.43

Tabulka B.2: Doby testů (s) pro stroj č. 1 (Athlon XP 1700+)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	5.85	48.11	1.66	1.77	4.70	4.80	31.72	2.61
Op–1	2.18	15.19	1.39	1.42	3.22	3.29	29.90	1.67
Op–2	2.41	15.60	1.31	1.36	3.12	3.18	29.37	1.71
Op–3	2.18	15.46	1.26	1.36	3.09	3.17	29.41	1.76
Op–4	2.65	15.26	1.44	1.45	3.41	3.43	29.47	1.96
Op–5	2.43	48.04	1.32	1.38	3.10	3.18	29.41	1.79
Op–6	2.16	45.41	1.23	1.33	2.97	3.04	29.11	1.76
Op–7	2.59	14.71	1.22	1.28	2.92	3.02	29.02	1.70
Op–8	2.60	40.29	1.21	1.26	2.89	2.89	29.02	1.71
Op–9	2.60	14.70	1.19	1.27	2.90	3.07	29.04	1.67
Op–10	2.60	14.73	1.21	1.29	2.92	2.99	29.04	1.70
Op–11	2.17	15.47	1.27	1.38	3.15	3.18	29.40	1.69
Op–12	2.17	15.02	1.23	1.29	2.96	2.96	29.09	1.63
Op–13	2.59	15.07	1.21	1.30	2.97	2.95	29.06	1.69
Op–14	2.59	13.22	1.26	1.24	2.86	2.93	29.01	1.66
Op–15	2.60	15.00	1.21	1.29	2.90	2.94	28.99	1.71
Op–16	2.60	15.02	1.19	1.24	3.00	3.04	29.17	1.69
Op–17	1.63	15.17	1.24	1.28	2.99	3.00	29.07	1.52
Op–18	1.64	14.93	1.22	1.29	3.03	3.10	29.14	1.61
Op–19	1.62	15.66	1.32	1.40	3.19	3.18	29.38	1.74
Op–20	1.67	15.55	1.34	1.40	3.16	3.20	29.42	1.77
Op–21	1.63	15.62	1.31	1.38	3.19	3.19	29.37	1.70
Op–22	1.66	15.50	1.35	1.39	3.16	3.20	29.35	1.75
Op–23	1.63	15.83	1.34	1.38	3.08	3.15	29.38	1.74
Op–24	1.66	15.45	1.31	1.39	3.24	3.20	29.45	1.69
Op–25	5.46	47.91	1.79	1.87	4.99	4.95	31.97	2.70
Op–26	2.22	15.20	1.35	1.42	3.30	3.41	29.90	1.85
Op–27	2.20	15.50	1.36	1.34	3.30	3.35	29.48	1.86
Op–28	2.24	15.47	1.37	1.40	3.29	3.45	29.51	1.76
Op–29	2.20	15.22	1.35	1.39	3.37	3.35	29.60	1.81
Op–30	2.19	48.01	1.35	1.40	3.22	3.31	29.52	1.82
Op–31	2.19	45.51	1.31	1.36	3.19	3.27	29.06	1.77
Op–32	2.19	13.14	1.25	1.34	3.12	3.19	29.12	1.72
Op–33	2.20	15.08	1.33	1.33	3.15	3.13	29.03	1.80
Op–34	2.19	15.19	1.31	1.35	3.21	3.30	29.11	1.69
Op–35	1.63	13.29	1.19	1.31	2.95	2.95	29.52	1.60

Tabulka B.3: Velikosti přeložených programů (kB) pro stroj č. 1 (Athlon XP 1700+)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	36	176	320	364	1132	1176	560	436
Op–1	32	560	252	284	860	892	396	336
Op–2	32	592	256	288	880	916	408	340
Op–3	32	596	428	476	1192	1228	512	360
Op–4	28	556	236	268	820	852	372	316
Op–5	32	176	256	288	992	1024	408	340
Op–6	32	184	276	312	1020	1056	436	384
Op–7	32	600	272	308	920	956	432	376
Op–8	32	260	272	308	1052	1088	432	380
Op–9	32	600	272	308	920	956	432	376
Op–10	32	600	272	308	920	956	432	376
Op–11	32	596	428	476	1192	1228	512	360
Op–12	36	608	444	492	1244	1280	528	404
Op–13	36	608	436	484	1224	1260	524	396
Op–14	36	868	440	488	1228	1264	528	404
Op–15	36	608	436	484	1224	1260	524	396
Op–16	36	608	436	484	1224	1260	524	396
Op–17	40	612	448	500	1368	1404	532	444
Op–18	56	656	864	948	2008	2048	628	504
Op–19	36	600	440	488	1332	1364	524	408
Op–20	48	640	760	832	1824	1856	608	460
Op–21	36	600	440	488	1332	1364	524	408
Op–22	48	640	760	832	1824	1856	608	460
Op–23	36	600	440	488	1332	1364	524	408
Op–24	48	640	760	832	1824	1856	608	460
Op–25	40	176	332	376	1104	1152	588	456
Op–26	32	560	264	296	856	892	412	348
Op–27	32	592	256	292	844	880	404	344
Op–28	32	596	380	444	1060	1124	488	364
Op–29	32	556	252	288	836	868	400	336
Op–30	32	176	256	292	944	976	404	344
Op–31	36	184	268	304	980	1016	436	364
Op–32	36	868	396	460	1224	1292	516	384
Op–33	36	608	400	468	1116	1180	520	388
Op–34	36	612	408	476	1196	1264	528	428
Op–35	40	872	448	500	1372	1408	536	452

Tabulka B.4: Doby překladů (s) pro stroj č. 2 (Intel Mobile Pentium III 800)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	0.77	9.13	8.68	10.11	19.80	14.43	17.66	11.62
Op–1	1.14	81.06	14.35	16.80	30.59	23.44	29.28	17.42
Op–2	1.68	129.91	23.73	27.92	47.44	38.96	52.40	24.43
Op–3	1.83	133.17	64.60	77.45	111.39	105.10	72.08	29.36
Op–4	1.46	173.00	20.72	24.70	41.12	34.40	46.91	20.16
Op–5	1.68	8.65	23.70	27.96	46.39	38.90	52.45	24.53
Op–6	1.72	8.65	22.97	27.10	44.31	37.39	51.83	24.06
Op–7	1.70	135.35	23.02	27.08	45.58	37.57	52.00	24.15
Op–8	1.70	8.90	22.68	26.74	43.98	36.91	51.28	23.86
Op–9	1.72	135.09	23.03	27.06	45.58	37.57	51.75	24.13
Op–10	1.75	132.70	23.23	27.36	46.18	37.81	52.19	24.92
Op–11	1.85	132.95	64.64	77.67	111.59	105.49	72.18	29.37
Op–12	1.87	136.07	61.27	73.09	105.19	99.23	69.40	29.23
Op–13	1.87	136.37	61.50	72.82	104.90	99.61	69.62	29.18
Op–14	1.86	141.70	60.97	72.48	104.16	98.67	68.55	28.93
Op–15	1.88	136.18	61.27	73.11	104.65	100.11	69.62	29.23
Op–16	1.95	133.94	61.40	72.67	105.78	100.10	70.08	29.96
Op–17	2.34	135.31	64.31	77.37	116.27	105.33	71.41	31.84
Op–18	2.99	144.89	169.10	220.66	271.34	301.59	80.97	34.18
Op–19	2.20	131.34	67.51	81.13	121.02	111.46	73.73	31.14
Op–20	2.73	146.02	124.08	154.24	205.87	211.40	83.05	33.30
Op–21	2.21	131.95	67.51	80.94	121.07	112.17	73.74	31.06
Op–22	2.75	144.22	125.05	154.29	206.49	211.01	83.08	33.26
Op–23	2.26	132.36	67.97	81.36	122.04	111.47	74.06	31.75
Op–24	2.76	144.98	124.82	156.75	209.70	209.28	83.86	34.01
Op–25	0.66	9.14	7.38	8.60	17.08	12.02	14.98	10.48
Op–26	1.00	81.34	11.30	13.07	24.94	17.77	22.68	15.78
Op–27	1.38	130.09	17.25	20.24	36.04	28.00	37.43	20.68
Op–28	1.48	133.55	32.83	40.10	60.97	56.62	47.02	22.02
Op–29	1.29	172.69	16.69	19.60	33.85	27.26	37.92	17.85
Op–30	1.39	8.63	17.27	20.24	35.29	27.95	37.43	20.58
Op–31	1.42	8.65	17.40	20.33	35.24	28.17	37.83	20.99
Op–32	1.53	142.08	32.42	39.68	59.80	54.90	47.20	22.13
Op–33	1.52	136.40	33.43	40.40	61.02	56.94	47.46	22.41
Op–34	1.63	135.18	34.41	42.19	64.69	59.54	48.48	23.70
Op–35	2.15	135.02	61.90	74.05	109.99	101.27	67.98	29.40

Tabulka B.5: Doby testů (s) pro stroj č. 2 (Intel Mobile Pentium III 800)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	27.11	338.75	3.34	3.41	8.30	8.39	41.25	4.54
Op–1	10.45	118.32	2.74	2.83	6.04	6.15	38.88	3.48
Op–2	10.34	113.34	2.59	2.70	5.81	5.90	38.85	3.40
Op–3	12.09	115.89	2.60	2.71	5.87	5.95	38.96	3.30
Op–4	13.93	115.17	2.78	2.89	6.12	6.25	38.98	3.46
Op–5	10.32	338.58	2.58	2.69	5.77	5.82	38.84	3.38
Op–6	10.29	323.92	2.48	2.54	5.45	5.59	38.34	3.15
Op–7	10.28	110.02	2.54	2.54	5.43	5.54	38.31	3.21
Op–8	10.21	292.84	2.39	2.51	5.22	5.43	38.23	3.11
Op–9	10.28	109.95	2.46	2.54	5.45	5.57	38.36	3.15
Op–10	10.27	109.98	2.47	2.55	5.44	5.50	38.30	3.17
Op–11	12.09	115.92	2.60	2.70	5.88	5.95	39.00	3.28
Op–12	10.22	112.86	2.44	2.56	5.43	5.55	38.43	3.16
Op–13	10.21	112.85	2.44	2.57	5.47	5.55	38.37	3.13
Op–14	10.22	97.99	2.39	2.49	5.36	5.38	38.33	3.04
Op–15	10.22	112.85	2.48	2.56	5.47	5.61	38.32	3.14
Op–16	10.22	112.86	2.46	2.58	5.48	5.58	38.34	3.13
Op–17	8.53	109.61	2.49	2.62	5.51	5.56	38.39	3.16
Op–18	8.15	111.03	2.52	2.69	5.58	5.70	38.49	3.14
Op–19	8.17	113.44	2.61	2.82	5.83	5.83	38.95	3.30
Op–20	8.20	112.54	2.68	2.78	5.90	6.03	39.08	3.37
Op–21	8.17	113.78	2.62	2.81	5.78	5.91	38.94	3.29
Op–22	8.27	112.73	2.70	2.77	5.87	5.96	39.08	3.34
Op–23	8.21	113.61	2.61	2.76	5.78	5.90	38.99	3.34
Op–24	8.47	112.76	2.68	2.79	5.87	6.02	39.09	3.29
Op–25	26.33	339.61	3.38	3.53	8.52	8.63	41.33	4.70
Op–26	10.55	118.53	2.79	2.82	6.12	6.19	38.95	3.55
Op–27	10.46	113.80	2.64	2.77	6.05	6.15	38.78	3.51
Op–28	10.45	116.24	2.72	2.80	6.07	6.17	38.88	3.45
Op–29	10.46	115.76	2.70	2.80	6.02	6.14	38.79	3.53
Op–30	10.44	339.86	2.67	2.81	5.99	6.12	38.77	3.55
Op–31	8.68	324.68	2.51	2.62	5.77	5.95	38.30	3.40
Op–32	8.87	98.53	2.53	2.55	5.61	5.78	38.32	3.23
Op–33	8.67	113.24	2.57	2.65	5.80	5.97	38.39	3.38
Op–34	8.70	109.79	2.58	2.65	5.90	6.10	38.40	3.41
Op–35	8.53	97.35	2.38	2.51	5.38	5.40	38.32	3.08

Tabulka B.6: Velikosti přeložených programů (kB) pro stroj č. 2 (Intel Mobile Pentium III 800)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	36	176	320	364	1132	1176	560	436
Op–1	32	560	252	284	860	892	396	336
Op–2	32	592	256	288	880	916	408	340
Op–3	32	596	428	476	1192	1228	512	360
Op–4	28	556	236	268	820	852	372	316
Op–5	32	176	256	288	992	1024	408	340
Op–6	32	184	276	312	1020	1056	436	384
Op–7	32	600	276	312	940	976	436	384
Op–8	32	260	276	312	1060	1096	436	388
Op–9	32	600	276	312	940	976	436	384
Op–10	32	600	276	312	940	976	436	384
Op–11	32	596	428	476	1192	1228	512	360
Op–12	36	608	444	492	1244	1280	528	404
Op–13	36	608	444	492	1244	1280	528	404
Op–14	36	868	444	492	1248	1284	532	412
Op–15	36	608	444	492	1244	1280	528	404
Op–16	36	608	444	492	1244	1280	528	404
Op–17	40	612	452	504	1388	1424	536	452
Op–18	56	656	872	956	2028	2068	632	512
Op–19	36	600	440	488	1332	1364	524	408
Op–20	48	640	760	832	1824	1856	608	460
Op–21	36	600	440	488	1332	1364	524	408
Op–22	48	640	760	832	1824	1856	608	460
Op–23	36	600	440	488	1332	1364	524	408
Op–24	48	640	760	832	1824	1856	608	460
Op–25	40	176	332	376	1104	1152	588	456
Op–26	32	560	264	296	856	892	412	348
Op–27	32	592	256	292	844	880	404	344
Op–28	32	596	380	444	1060	1124	488	364
Op–29	32	556	252	288	836	868	400	336
Op–30	32	176	256	292	944	976	404	344
Op–31	36	184	268	304	980	1016	436	364
Op–32	36	868	396	460	1224	1292	516	384
Op–33	36	608	400	468	1116	1180	520	388
Op–34	36	612	408	476	1196	1264	528	428
Op–35	40	872	456	508	1388	1424	540	456

Tabulka B.7: Doby překladů (s) pro stroj č. 3 (Athlon XP 1600+)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	0.34	4.87	4.00	4.64	9.15	6.43	8.04	5.59
Op–1	0.54	44.12	6.06	7.04	13.39	9.58	12.28	8.40
Op–2	0.74	71.96	9.25	10.94	19.29	15.52	20.47	11.12
Op–3	0.77	73.79	18.26	22.57	33.18	30.57	26.00	11.94
Op–4	0.69	106.03	9.26	10.90	18.78	15.18	21.22	9.88
Op–5	0.76	4.71	9.52	11.21	19.55	15.54	21.02	11.40
Op–6	0.78	4.73	9.63	11.33	19.54	15.62	21.32	11.52
Op–7	0.77	73.97	9.64	11.16	19.89	15.50	21.22	11.46
Op–8	0.75	4.83	9.42	11.03	19.20	15.28	20.96	11.39
Op–9	0.77	73.18	9.51	11.15	19.66	15.56	21.07	11.45
Op–10	0.77	72.30	9.48	11.09	19.80	15.35	20.97	11.57
Op–11	0.79	72.77	17.78	22.08	32.94	30.40	25.88	11.85
Op–12	0.83	72.41	17.81	22.14	32.78	30.35	25.99	11.97
Op–14	0.76	76.44	17.57	21.87	32.53	29.75	25.86	11.92
Op–15	0.82	74.44	18.34	22.71	33.75	31.26	26.70	12.25
Op–16	0.85	74.23	18.41	22.84	34.20	31.47	26.90	12.66
Op–17	0.89	75.83	19.28	23.92	36.08	32.68	27.42	13.33
Op–18	1.19	80.21	34.64	46.62	60.46	64.84	30.53	13.76
Op–19	0.85	75.20	18.98	23.59	35.75	32.33	26.96	12.84
Op–20	1.16	84.26	36.93	49.73	63.64	68.04	32.27	14.32
Op–21	0.89	75.55	19.11	23.58	35.50	32.29	26.89	12.73
Op–22	1.13	78.16	33.52	45.30	58.29	62.36	29.39	12.91
Op–23	0.86	74.45	19.28	23.90	36.28	32.81	27.28	13.34
Op–24	1.20	80.58	35.94	46.65	58.89	66.83	30.18	13.35
Op–35	0.81	72.78	17.17	21.50	32.47	29.81	24.97	12.16

Tabulka B.8: Doby testů (s) pro stroj č. 3 (Athlon XP 1600+)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	14.69	180.71	1.99	2.02	5.64	5.68	29.70	2.64
Op–1	6.90	53.31	1.51	1.54	3.77	3.80	26.91	1.69
Op–2	6.81	52.45	1.49	1.54	3.96	4.00	27.20	1.71
Op–3	6.95	56.03	1.58	1.52	3.75	4.00	27.32	1.67
Op–4	7.20	62.32	1.55	1.58	3.92	3.95	27.69	1.76
Op–5	7.19	191.99	1.55	1.60	3.88	3.90	27.66	1.77
Op–6	7.18	176.53	1.49	1.58	3.78	3.81	27.17	1.70
Op–7	7.19	54.71	1.48	1.56	3.71	3.81	26.97	1.70
Op–8	7.18	148.23	1.42	1.50	3.57	3.62	26.86	1.62
Op–9	6.79	51.09	1.44	1.53	3.71	3.75	26.59	1.61
Op–10	6.79	51.27	1.45	1.50	3.65	3.65	26.47	1.63
Op–11	6.98	54.80	1.50	1.53	3.73	3.81	27.02	1.68
Op–12	6.82	53.82	1.43	1.50	3.59	3.65	26.32	1.61
Op–14	6.81	44.29	1.42	1.47	3.51	3.55	26.36	1.56
Op–15	6.81	53.57	1.50	1.56	3.71	3.72	26.87	1.64
Op–16	6.80	53.89	1.48	1.56	3.68	3.74	26.92	1.66
Op–17	6.82	51.72	1.49	1.54	3.73	3.81	27.00	1.66
Op–18	6.01	52.37	1.51	1.54	3.80	3.79	27.13	1.63
Op–19	6.87	54.50	1.58	1.58	3.82	3.89	27.49	1.71
Op–20	5.91	52.45	1.64	1.71	3.95	4.02	29.15	1.82
Op–21	6.86	54.48	1.51	1.52	3.83	3.86	27.01	1.66
Op–22	5.91	52.39	1.48	1.54	3.79	3.81	26.92	1.69
Op–23	6.89	54.40	1.58	1.58	3.83	3.94	27.73	1.73
Op–24	5.91	52.28	1.51	1.58	3.83	3.84	26.95	1.67
Op–35	6.88	46.24	1.38	1.47	3.53	3.72	26.62	1.55

Tabulka B.9: Velikosti přeložených programů (kB) pro stroj č. 3 (Athlon XP 1600+)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	40	176	332	376	1108	1152	588	456
Op–1	32	564	264	296	852	888	412	352
Op–2	32	588	256	292	840	876	404	344
Op–3	32	596	380	444	1052	1116	488	368
Op–4	32	552	252	288	828	864	400	336
Op–5	32	176	256	292	944	976	404	344
Op–6	36	184	268	304	980	1016	436	364
Op–7	36	600	268	304	884	920	436	364
Op–8	32	260	268	304	1152	1188	432	364
Op–9	36	600	268	304	884	920	436	364
Op–10	36	600	268	304	884	920	436	360
Op–11	32	596	380	444	1052	1116	488	368
Op–12	36	608	400	468	1108	1176	520	388
Op–14	36	864	396	460	1216	1284	516	384
Op–15	36	608	400	468	1108	1176	520	388
Op–16	36	608	400	468	1108	1172	520	384
Op–17	36	612	408	476	1188	1256	528	428
Op–18	48	656	632	780	1516	1664	596	456
Op–19	36	600	388	452	1136	1204	496	408
Op–20	48	640	608	748	1456	1596	568	436
Op–21	36	600	388	452	1136	1204	496	408
Op–22	48	640	608	748	1456	1596	568	436
Op–23	36	600	388	452	1136	1200	496	408
Op–24	48	640	608	748	1456	1596	568	436
Op–35	36	872	404	472	1324	1392	524	436

Tabulka B.10: Doby překladů (s) pro stroj č. 4 (2 × Intel Xeon 2,2)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	0.25	2.80	3.04	3.41	7.10	4.93	6.01	4.10
Op–1	0.44	42.65	4.72	5.41	10.51	7.55	9.25	6.26
Op–2	0.62	54.16	7.58	8.89	15.57	12.20	15.88	8.59
Op–3	0.64	53.58	21.50	27.36	36.13	36.94	19.98	9.42
Op–4	0.59	46.98	7.25	8.60	14.76	11.88	15.88	7.70
Op–5	0.59	2.48	7.53	8.76	15.37	12.09	15.80	8.70
Op–6	0.61	2.44	7.50	8.83	15.28	12.20	16.33	8.52
Op–8	0.63	2.83	7.53	8.76	15.14	12.05	16.13	8.72
Op–9	0.65	54.82	7.59	8.89	15.63	12.15	16.05	8.63
Op–10	0.64	54.62	7.57	8.96	15.79	12.37	16.48	8.99
Op–11	0.66	53.68	21.43	27.29	36.26	36.61	19.86	9.44
Op–12	0.69	54.54	21.70	27.20	36.55	36.86	20.30	9.45
Op–13	0.69	56.32	21.21	27.39	36.49	36.90	20.42	9.53
Op–14	0.64	57.55	21.35	27.04	35.97	36.39	20.05	9.48
Op–15	0.68	54.83	21.41	27.21	36.40	36.76	20.26	9.65
Op–16	0.71	55.98	21.89	27.58	36.88	36.94	20.45	9.86
Op–17	0.80	55.31	22.86	28.43	38.79	38.80	20.93	10.55
Op–18	1.02	61.61	47.10	61.65	71.94	80.44	23.53	11.26
Op–19	0.77	54.67	22.19	28.45	38.75	38.35	20.38	10.16
Op–20	1.00	59.44	46.69	59.98	73.36	79.91	22.99	10.75
Op–21	0.77	54.74	22.30	28.57	38.28	38.38	20.27	10.12
Op–22	0.97	60.52	46.94	59.98	73.40	80.52	22.82	10.75
Op–23	0.79	54.46	22.47	28.94	38.70	38.55	20.50	10.52
Op–24	1.03	59.89	46.48	60.05	72.94	79.90	23.13	10.97
Op–25	0.25	2.86	2.52	3.00	5.65	4.04	6.45	3.51
Op–26	0.39	42.18	4.33	4.96	9.44	6.77	9.80	5.56
Op–27	0.51	54.21	6.26	7.25	12.85	9.97	14.64	7.27
Op–28	0.53	55.18	12.11	15.97	22.19	22.47	17.79	7.78
Op–29	0.51	47.96	6.32	7.15	12.38	9.93	15.60	6.27
Op–30	0.49	2.53	6.27	7.13	12.57	10.08	14.72	7.19
Op–31	0.63	2.77	6.81	7.42	12.90	10.73	14.22	7.65
Op–32	0.54	55.89	12.03	16.18	22.05	22.82	15.76	7.69
Op–33	0.54	55.91	11.95	16.51	22.04	22.96	16.19	7.60
Op–34	0.58	55.57	12.69	16.99	23.88	23.69	16.54	8.27
Op–35	0.73	43.35	21.34	26.71	36.60	36.01	19.87	9.81

Tabulka B.11: Doby testů (s) pro stroj č. 4 (2 × Intel Xeon 2,2)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	13.06	191.32	1.76	1.78	4.10	4.22	27.88	1.79
Op–1	3.97	27.20	1.58	1.44	3.17	3.33	26.17	1.40
Op–2	3.93	28.69	1.39	1.40	3.08	3.20	26.48	1.37
Op–3	3.88	28.23	1.39	1.43	2.97	3.16	26.56	1.38
Op–4	3.92	27.59	1.43	1.43	3.03	3.24	26.46	1.31
Op–5	4.36	191.00	1.39	1.40	3.08	3.19	26.59	1.34
Op–6	3.86	149.84	1.36	1.31	2.90	3.08	26.19	1.35
Op–8	3.87	135.75	1.32	1.87	2.88	3.03	26.26	1.32
Op–9	3.86	28.05	1.34	1.36	3.05	3.06	26.12	1.33
Op–10	3.87	28.34	1.31	1.37	2.92	3.03	26.16	1.39
Op–11	3.87	28.34	1.36	1.43	3.02	3.10	26.34	1.33
Op–12	3.87	29.86	1.36	1.39	3.00	3.10	26.23	1.35
Op–13	3.86	29.35	1.33	1.43	3.01	3.19	26.22	1.40
Op–14	3.84	27.56	1.35	1.87	2.95	3.13	26.25	1.24
Op–15	3.87	29.05	1.33	1.40	3.04	3.09	26.26	1.28
Op–16	3.90	28.96	1.39	1.42	3.04	3.11	26.31	1.35
Op–17	3.92	28.35	1.35	1.40	3.10	3.21	26.28	1.36
Op–18	3.82	28.54	1.40	1.47	3.12	3.12	26.30	1.39
Op–19	3.88	28.51	1.40	1.41	3.09	3.27	26.30	1.33
Op–20	3.84	29.38	1.39	1.49	3.02	3.18	26.22	1.35
Op–21	3.88	28.42	1.41	1.42	3.03	3.30	26.31	1.29
Op–22	3.87	28.65	1.44	1.43	3.16	3.18	26.32	1.33
Op–23	3.88	28.12	1.44	1.42	3.18	3.15	26.20	1.34
Op–24	3.77	28.85	1.40	1.50	3.15	3.25	26.21	1.34
Op–25	12.81	191.70	1.77	1.77	4.09	4.26	28.30	1.69
Op–26	4.00	28.04	1.38	1.38	3.14	3.21	26.79	1.37
Op–27	4.02	28.55	1.37	1.40	2.93	3.20	26.45	1.25
Op–28	4.05	28.42	1.37	1.41	3.08	3.15	26.77	1.24
Op–29	4.02	28.62	1.44	1.57	3.08	3.13	26.78	1.32
Op–30	4.01	194.54	1.47	1.44	3.00	3.05	26.59	1.27
Op–31	4.02	149.59	1.38	1.37	3.20	3.32	26.69	1.30
Op–32	3.94	27.76	1.37	1.36	2.88	3.02	26.32	1.25
Op–33	4.00	27.93	1.37	1.41	3.02	3.07	26.76	1.28
Op–34	3.95	33.77	1.29	1.38	2.98	3.13	26.72	1.25
Op–35	3.85	27.91	1.30	1.90	2.93	3.13	26.61	1.27

Tabulka B.12: Velikosti přeložených programů (kB) pro stroj č. 4 (2 × Intel Xeon 2,2)

optim.	patgen	opatgen	tex	etex	pdftex	pdfetex	omega	dvipdfm
Op–0	36	208	324	368	1144	1188	568	440
Op–1	32	608	260	296	904	940	412	352
Op–2	32	644	264	300	908	944	420	348
Op–3	32	644	476	556	1264	1348	508	376
Op–4	32	560	260	296	892	928	412	340
Op–5	32	208	264	300	1008	1044	420	348
Op–6	32	208	276	312	1044	1080	452	368
Op–8	32	332	276	312	1240	1276	448	368
Op–9	32	668	276	312	952	988	452	368
Op–10	32	668	276	312	952	988	452	368
Op–11	32	644	476	556	1264	1348	508	376
Op–12	36	664	512	596	1336	1416	532	396
Op–13	36	664	512	596	1336	1416	532	396
Op–14	36	684	508	592	1448	1536	532	396
Op–15	36	664	512	596	1336	1416	532	396
Op–16	36	664	512	596	1336	1416	532	396
Op–17	40	668	520	608	1440	1520	540	444
Op–18	48	712	1004	1208	2116	2316	628	492
Op–19	36	648	484	568	1368	1452	520	428
Op–20	48	692	972	1164	2048	2236	600	472
Op–21	36	648	484	568	1368	1452	520	428
Op–22	48	692	972	1164	2048	2236	600	472
Op–23	36	648	484	568	1368	1452	520	428
Op–24	48	692	972	1164	2048	2236	600	472
Op–25	40	208	332	376	1156	1200	652	464
Op–26	32	608	264	300	904	936	436	356
Op–27	32	644	264	296	900	932	432	352
Op–28	32	644	384	444	1108	1168	508	384
Op–29	32	560	260	292	888	920	428	348
Op–30	32	208	264	296	992	1024	432	352
Op–31	32	208	268	300	1000	1032	428	368
Op–32	36	684	384	444	1244	1308	504	404
Op–33	36	664	388	448	1120	1184	504	400
Op–34	36	668	396	456	1240	1304	516	460
Op–35	36	688	516	604	1580	1668	540	460

Příloha C

Kompletní ukázky profilů

PatGen

„Přímý“ profil:

```
Flat profile:

Each sample counts as 0.01 seconds.
  %  cumulative self              self  total
 time  seconds seconds   calls  s/call s/call name
 90.73  11.16  11.16   463824  0.00   0.00  readword
  5.28  11.81   0.65   463824  0.00   0.00  hyphenate
  1.63  12.01   0.20  9222606  0.00   0.00  eoln
  1.06  12.14   0.13   463824  0.00   0.00  changedots
  1.06  12.27   0.13   463824  0.00   0.00  doword
  0.16  12.29   0.02       18  0.00   0.68  dodictionary
  0.08  12.30   0.01        2  0.01   0.01  deletebadpatterns
  0.00  12.30   0.00   463844  0.00   0.00  eof
  0.00  12.30   0.00   463825  0.00   0.00  readln
  0.00  12.30   0.00     2439  0.00   0.00  zinsertcpat
  0.00  12.30   0.00     1522  0.00   0.00  zinsertpattern
  0.00  12.30   0.00     1522  0.00   0.00  znewtrieop
  0.00  12.30   0.00     1282  0.00   0.00  firstcfit
  0.00  12.30   0.00      758  0.00   0.00  zunpackc
  0.00  12.30   0.00      602  0.00   0.00  firstfit
  0.00  12.30   0.00      389  0.00   0.00  zunpack
  0.00  12.30   0.00       71  0.00   0.00  fprintreal
  0.00  12.30   0.00       22  0.00   0.00  kpse_fopen_trace
  0.00  12.30   0.00       21  0.00   0.00  cmdline
  0.00  12.30   0.00       21  0.00   0.00  xfopen
  0.00  12.30   0.00       19  0.00   0.00  initcounttrie
  0.00  12.30   0.00       18  0.00   0.00  collectcounttrie
  0.00  12.30   0.00       18  0.00   0.00  ztraversecounttrie
  0.00  12.30   0.00       12  0.00   0.00  xmalloc
  0.00  12.30   0.00        4  0.00   0.00  concat3
  0.00  12.30   0.00        4  0.00   0.00  xrealloc
  0.00  12.30   0.00        3  0.00   0.00  element
  0.00  12.30   0.00        3  0.00   0.00
                                            kpse_filename_component
  0.00  12.30   0.00        3  0.00   0.00  xdirname
  0.00  12.30   0.00        3  0.00   0.00  xputenv
```

```
 0.00   12.30    0.00         3  0.00    0.00  zinput2ints
 0.00   12.30    0.00         2  0.00    0.00  CopyFirst
 0.00   12.30    0.00         2  0.00    0.00  StripFirst
 0.00   12.30    0.00         2  0.00    0.00  xstrdup
 0.00   12.30    0.00         2  0.00    0.00  zdeletepatterns
 0.00   12.30    0.00         2  0.00    0.00  zinput3ints
 0.00   12.30    0.00         1  0.00    0.00  expand_symlinks
 0.00   12.30    0.00         1  0.00    0.00  find_suffix
 0.00   12.30    0.00         1  0.00    0.00  initialize
 0.00   12.30    0.00         1  0.00    0.00  initpatterntrie
 0.00   12.30    0.00         1  0.00    0.00  kpse_absolute_p
 0.00   12.30    0.00         1  0.00    0.00  kpse_fclose_trace
 0.00   12.30    0.00         1  0.00    0.00  kpse_set_progname
 0.00   12.30    0.00         1  0.00    0.00
                                               kpse_set_program_name
 0.00   12.30    0.00         1  0.00   12.30  mainbody
 0.00   12.30    0.00         1  0.00    0.00  parsearguments
 0.00   12.30    0.00         1  0.00    0.00  readpatterns
 0.00   12.30    0.00         1  0.00    0.00  readtranslate
 0.00   12.30    0.00         1  0.00    0.00  remove_dots
 0.00   12.30    0.00         1  0.00    0.00  selfdir
 0.00   12.30    0.00         1  0.00    0.00  xgetcwd
 0.00   12.30    0.00         1  0.00    0.00  zfindletters
 0.00   12.30    0.00         1  0.00    0.00  zoutputpatterns
```

Příloha D

Obsah CD

Součástí této práce je také CD. Disk je ve formátu Rock Ridge s rozšířením Joliet.

Obsah:

- Zdrojový kód této práce ve formátu XML (podle DTD DocBook), samotná práce v několika různých formátech.
- Zdrojové kódy aplikací použitých pro testy a všechna testovací data.
- Kompletní výsledky testů a všechny přeložené programy.
- Kompletní výstupy profilů uvedených v části Oddíl 4.2.
- Zdrojové kódy a dokumentace k programu UPatGen.

Literatura

[Antos] *PatLib, Pattern Manipulating Library*, David Antoš, Diplomová práce, Fakulta informatiky Masarykovy univerzity v Brně, 2001, http://www.fi.muni.cz/~xantos/patlib/. (document), 5.3, 5.3.2

[COptim] *Optimization of Computer Programs in C*, Michael E. Lee, 1997, http://www.leto.net/docs/C-optimization.html.

[CWeb] *The CWEB System of Structured Documentation*, Donald E. Knuth a Silvio Levy, 2002, http://www-cs-faculty.stanford.edu/~knuth/cweb.html. 2.3

[GVzUni] *Generování vzorů dělení slov v UNICODE*, David Antoš a Petr Sojka, SLT 2001, s. 23-32, 2001.

[Judy] *Judy library documentation*, Alan Silverstein a Doug Baskins, 2002, http://judy.sourceforge.net/, http://sourceforge.net/projects/judy/. 5.4.2

[Liang] *Word Hy-phen-a-tion by Com-put-er*, Franklin Mark Liang, Department of Computer Science, Stanford University, 1983. 5.3.1

[Mach] *Přebíjející vzory ve zpracování přirozeného jazyka*, David Macháček, Diplomová práce, Fakulta informatiky Masarykovy univerzity v Brně, 2003, http://www.fi.muni.cz/~xmachac1/I999/. 3.3

[Myth] *GCC Myths and Facts*, Joao Seabra, Freshmeat, 2003, http://freshmeat.net/articles/view/730/. 1.3.2

[Optim] *Optimization*, Jeff Covey, Freshmeat, 2000, http://freshmeat.net/articles/view/186/.

[PGRev] *Pattern Generation Revisited*, David Antoš a Petr Sojka, EuroTEX 2001 Proceedings, s. 7-17, 2001.

[Tetex] *teTEX Manual*, Thomas Esser, součást distribuce teTEX, 2003. 2

[Texbook] *The Texbook*, Donald E. Knuth, Addison-Wesley, 0-201-13447-0, 1996. 3.3

[Thai] *Thai Segmentation Pattern Generation by OPatGen*, David Antoš a Petr Sojka, TUGboat, Volume 23 (2002), s. 111-118, 2002.

[Web] *Literate Programming*, Donald E. Knuth, The Computer Journal, 1983. 2.3