GCC - překladač, optimalizace, měření, CPU

Petr Hodač

SUSE Labs

2. 11. 2024

Outline

GCC

- GNU Compiler Collection
- GNU Tools Cauldron - Conference

Podporovane jazyky

C, C++, Objective-C, Objective-C++, Fortran, Ada, D, a Go

ToolChain

gcc, gdb, glibc, debuginfod, valgrind, binutils

Vývojový cyklus

Stage 1 - cca 4 měsíce

Vývoj a přidávání velkých změn. Vývoj :)

Stage 2

Od gcc 4 4 stejné jako Stage 1 := Stage 2 tj. není

Stage 3 - cca 2 měsice

Bug fixy a review submitnutých patchu/iterace už není možné přidávat nové veci

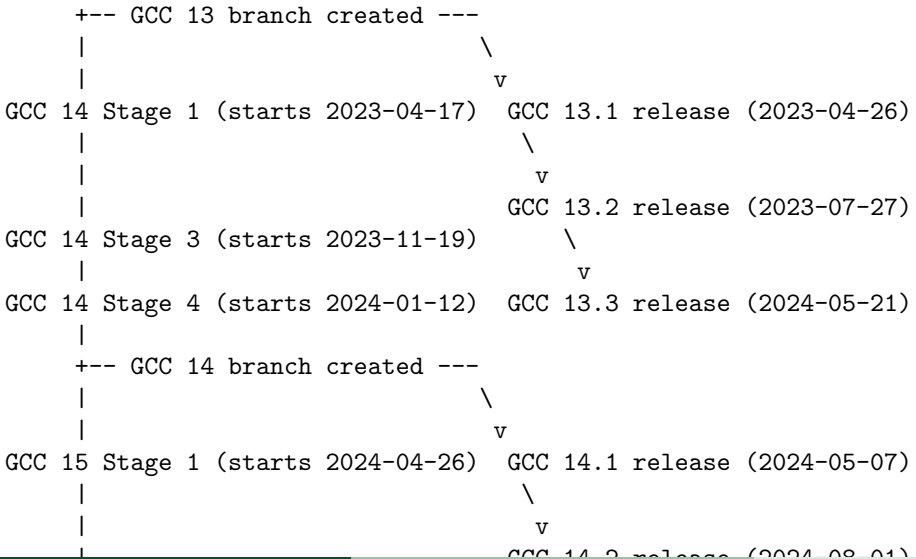
Stage 4 - do dalsi Stage 1

Je možné opravovat jen regrese a bugy trvá dokud není hotova :)

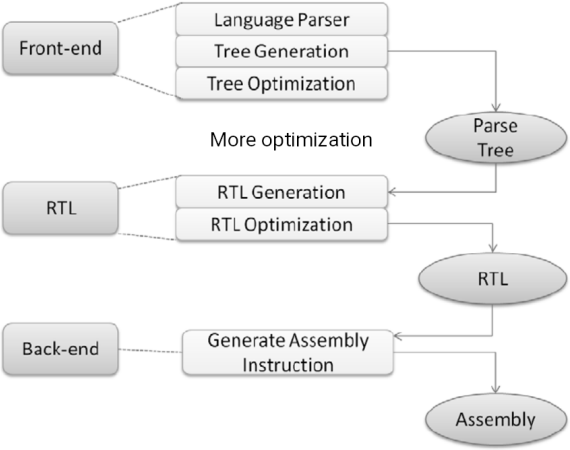
Verzování GCC

- VERSION=14
- VERSION.0.0 - experimental stage 1-3
- VERSION.0.1 - experimental stage 4
- VERSION+1.0.0 - switch to now trunk stage 1-3
- VERSION.1.0
- VERSION.2.0
- VERSION+1.0.1 - new gcc experimental stage 4

Priklad GCC 13, 14 a 15



compilator



[3]Architecture-of-the-Gnu-Compiler, Maria Varanda Pereira

Frontend

symbol := vyraz se symboly

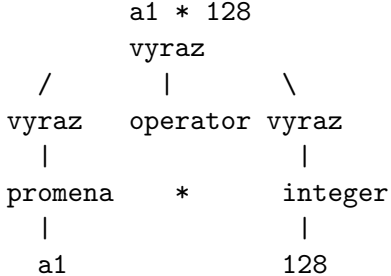
$$S -> A_1 A_2 A_N$$

$$A -> B_1 B_2 B_N$$

$$A -> C_1 C_2 C_N$$

$$A -> B_1 B_2 B_N | C_1 C_2 C_N$$

$$A -> B | \epsilon$$

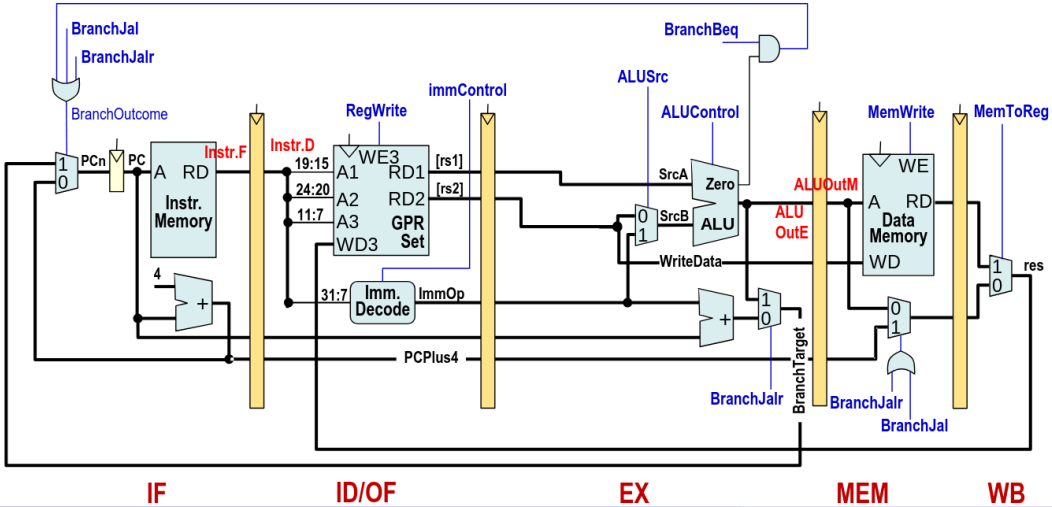


GCC - intermadia

```
GENERIC - AST -  TREE - frontend
GIMPLE/SSA  - > podmozina GENERIC
RTL low lewel -> pro code generator

-f dump-tree-all
-f dump-rtl-all
-f dum-$NAZEV-all
-f verbose asm
```

5 stupňový RISC



[1] BI-APS, FIT CVUT, Stepanovsky, Tvrdik, 2024

Intrukce a zrychlení

Instrukce v case

```
IF ID EX MEM WB  IF  ID  EX  MEM WB  IF  ID  EX  MEM WB  IF  ID  EX
   IF ID EX  MEM WB  IF  ID  EX  MEM WB  IF  ID  EX  MEM WB  IF  ID
      IF ID  EX  MEM WB  IF  ID  EX  MEM WB  IF  ID  EX  MEM WB  IF
    (4t) IF  ID  EX  MEM WB  IF  ID  EX  MEM WB  IF  ID  EX  MEM WB
            IF  ID  EX  MEM WB  IF  ID  EX  MEM WB  IF  ID  EX  MEM
            t   t   t   t   t   t   t   t   t   t   t
```

$$T_{k,n} = nkt$$

$$S_{k,n} = \frac{T_1 n}{T_{k,n}} = \frac{nkt}{kt + (n-1)t}$$

$$\lim_{n \to \infty} \frac{nkt}{kt + (n-1)t} = \infty$$

Jak měříme?

- Různé architektury
- Stejné systémy ,různe flagy
- Compilace -j max
- Run 1 static CPU pinning
- SPEC2017, 2006, own cpp benchmark

Sum

```
/proc/sys/kernel/randomize_va_space
0

ulimit -s  #Stacklimit
unlimited
```

LNT

Infrastructure for performance testing - LLVM

Dlouhodobe mereni regresi na trunku a vydanych GCC, kazdy den mnoho ops aktulanim trunkem

lntzen4.spec2006.gcc-trunk.O2_generic_lto_fortify

Test	1 week	1 week	1 week	1 week	1 week	1 week	1 week	5 days	3 days	3 days	1 day	23 hours	39 minutes	Sparkline
SPEC/SPEC2006/INT/456.hmmer	135.849	~	~	~	~	~	~	2.17%	2.14%	~	~	~	-6.58%	
SPEC/SPEC2006/FP/434.zeusmp	81.522	-2.15%	-2.12%	~	-2.10%	-2.20%	~	-2.16%	~	~	~	~	-2.01%	

- Ověřování zlepšení/zhoršení na různých platformách
- Hlídání Backporování
- Hlídání nechtěných regresi
- Misscompare, ICE, misscompilace
- zen2, zen3, zen4, arm, intel

Obecne

```
--help=optimizer
-fdump-tree-optimized
gcc -OPTIMALIZACE -Q --help=optimizers
```

- -O0 - nic
- -Os - code size
- -O1 - pro generované a obrovské kódy
- -O2 - standart :)
- -Ofast - pozor na matiku
- -O3

`Cilova architektura`

- -march znver3, znver4, x86-64-v{1..4}, dalsi
- -mtune

Unrolling

```
-funroll-loops
for (i = 1; i < 13 ; i++) {
    pole[i-1] += 10;
  }
#basic unrolling
for i in 0,1,2
do
 pole[0+4*i] += 10 ;
 pole[1+4*i] += 10 ;
 pole[2+4*i] += 10 ;
 pole[3+4*i] += 10 ;
done
for j in 13-(i+1*4);
pole [i+1*4] += 10;
```

Vektorizace

```
double A1 A2 A3 A4 (64b = 8B)
double B1 B2 B3 B4

ADD A1+B1
ADD A2+B2
ADD A3+B3
ADD A4+B4

Vektorizace s  AVX2 (256b = 32B)
ADD A1+B1 ; A2+B2 ; A3 +B3 ; A4+B4


-f-dump-optimized
```

-ofast pozor na asoci. $10^6 - 10^6 + 10^-6$

Inlining

Nahrazování volání tělem malých fci - máme limit jak moc malích

```
int pred(int x) {
    if (x == 0)
        return 0;
    else
        return x - 1;
}
int func(int y) {
    return pred(y) + pred(0) + pred(y+1);
}
int func(int y) {
    int tmp;
    if (y   == 0) tmp  = 0; else tmp  = y - 1;       /* (1) */
    if (0   == 0) tmp += 0; else tmp += 0 - 1;       /* (2) */
    if (y+1 == 0) tmp += 0; else tmp += (y + 1) - 1; /* (3) */
    return tmp;
}
```

PGO

Profile-guided optimization

```
#ideal C++17
-fprofile-generate
Spustit train runu pro vygenerovani .gnc[a-z]
souboru kde se pozdeji urci co stoji zato zoptimalizovat
--profile-use
```

Výhody

Nárust výkonu 5-35 procent ve specu zmenšení velikosti kodu

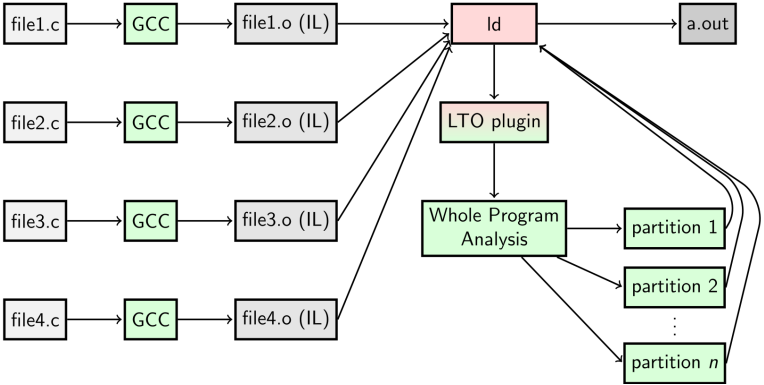
Neco

Nárust času kompilace + čas test runu

LTO

Link Time Optimization

bez LTO se zpracová každé .o v zvlášt, kdežto s LTO



[2]Advanced Optimization and New Capabilities of GCC Jambor, Hubicka, Biener, Matz,

LTO

Whole Porgram analysis

WPA probiha seriove (pomalejsi) komilace ale jedna se o relativne malou cast jednotlive partitions uz jedou zase paralelne dle -lto=128 (az 256)

- nemá cenu nad optimalním kódem -¿ enkody/kernel
- kompile time se zvyšuje
- velikost klesá až 30
- zrychleni se projevuje na složitych a velkých aplikacích

GCC v case gcc10 vs trunk

https://lnt.opensuse.org/db_default/v4/SPEC/56353?compare_to=37116

Compile time se zvysuje (na stejnem CPU)
Velikost klesa cca 10%
Runtime se zkracuje -> vyssi vykon az cca 5-15% dle testu (i 35%)

Bezpečnost?

```
openSUSE build

-U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=3 -fstack-protector-strong
-funwind-tables -fasynchronous-unwind-tables
-fstack-clash-protection

-fhardened #gcc 14.1+

-D_FORTIFY_SOURCE=3 # or 2 old glibc
-D_GLIBCXX_ASSERTIONS
-ftrivial-auto-var-init=pattern
-fPIE -pie -Wl,-z,relro,-z,now
-fstack-protector-strong
-fstack-clash-protection
-fcf-protection=full (x86 GNU/Linux only)
```

References I

Architektura počítačových systémů (BI-APS) FIT CVUT Návrh zřetězené RISC mikroarchitektury Michal Štepanovský, Pavel Tvrdík 2024

Advanced Optimization and New Capabilities of GCC Jambor, Hubicka, Biener, Matz, Hollingsworth https://documentation.suse.com/sbp/devel-tools/html/SBP-GCC-12/index.html

Maria Varanda Pereira https://www.researchgate.net/figure/Architecture-of-the-Gnu-Compiler-Collection-GCC_fig1_220117845