Infrastruktura pro profilem řízené optimalizace v GCC

Zdeněk Dvořák, Jan Hubička, Pavel Nejedlý, Josef Zlomek

Historie GCC

- Začátek 80.let — Vznik projektu
- 1987 — Verze 1.0
- 1992 — Verze 2.0
- 1997 — Zahájeny práce na v. 3.0

Historie GCC

- Začátek 80.let Vznik projektu
- 1987 Verze 1.0
- 1992 Verze 2.0
- 1997 Zahájeny práce na v. 3.0
- 1997 Global CSE
- 1999 Verze 2.95
- 2000 SSA form
- 2001 Verze 3.0
- ? květen 2002 Verze 3.1

Struktura GCC

- FrontEnd
- BackEnd

Struktura GCC

- FrontEnd
  - Specifické pro jednotlivé jazyky
  - Podporované jazyky: c, c++, Java, Fortran, Ada, ...
  - Kód reprezentován syntaktickými stromy
  - Minimum optimalizací (např. inlinování)
- BackEnd

Struktura GCC

- FrontEnd
  - Specifické pro jednotlivé jazyky
  - Podporované jazyky: c, c++, Java, Fortran, Ada, ...
  - Kód reprezentován syntaktickými stromy
  - Minimum optimalizací (např. inlinování)
- BackEnd
  - Nezávislý na jazyce
  - Kód v RTL
  - Podporovaná podmnožina RTL je závislá na cílové architektuře
  - Většina optimalizací

Výhody GCC

- Snadné přidávání podpory pro nové architektury
- Snadná rozšiřitelnost pro nové jazyky
- Open-source
- Výkonné optimalizace

Problémy GCC

Zastaralý design:

- Heuristická řešení problémů
- Příliš nízká úroveň RTL
- Pouze lokální optimalizace

Problémy GCC

Zastaralý design:

- Heuristická řešení problémů
- Příliš nízká úroveň RTL
- Pouze lokální optimalizace

Další problémy:

- Duplikace kódu
- Nejasná interakce mezi jednotlivými optimalizačními průchody

Cíle projektu

- Zavést do GCC optimalizace řízené profilem
  - Naměřené chování
  - Statický odhad

Cíle projektu

- Zavést do GCC optimalizace řízené profilem
  - Naměřené chování
  - Statický odhad
- Sjednotit implementaci Control Flow Graphu
- Zajistit udržování této struktury

Cíle projektu

- Zavést do GCC optimalizace řízené profilem
  - Naměřené chování
  - Statický odhad
- Sjednotit implementaci Control Flow Graphu
- Zajistit udržování této struktury
- Použít ji pro sběr a ukládání profilovacích informací
- Využít profil v některých optimalizacích
- Přidat nové optimalizace na nich založené

Dosažené výsledky

- Všechny cíle dosaženy

Dosažené výsledky

- Všechny cíle dosaženy
- Optimalizace založené na profilovacích informacích, např.
  - Přerovnávání basic blocků – Software Trace Cache
  - Formace superbloků – Tracer
  - Code alignment
  - Přerovnávání funkcí
  - Eliminace předčasných konců superbloků

Dosažené výsledky

- Všechny cíle dosaženy
- Optimalizace založené na profilovacích informacích
- Další optimalizace, např.
  - Nový výkonnější optimalizátor skoků
  - Přepsána část loop optimalizátoru
  - Webizer, register coalescing

Dosažené výsledky

- Všechny cíle dosaženy
- Optimalizace založené na profilovacích informacích
- Další optimalizace
- Zahájeny práce na zvýšení úrovně RTL

Profilování

- Hledání částí kódu vhodných k optimalizaci

Profilování

- Hledání částí kódu vhodných k optimalizaci
- Různá kritéria a metody

Profilování

- Hledání částí kódu vhodných k optimalizaci
- Různá kritéria a metody
- Sampling

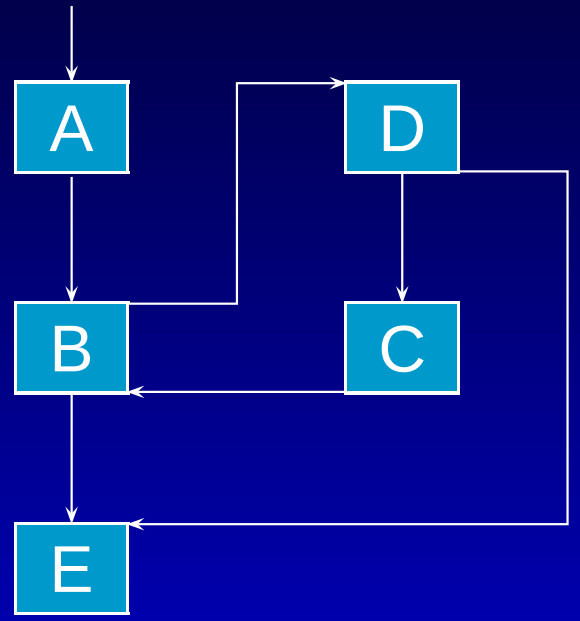
Profilování

- Hledání částí kódu vhodných k optimalizaci
- Různá kritéria a metody
- Sampling
- Coverage

Coverage (Pokrytí)

Control Flow Graph

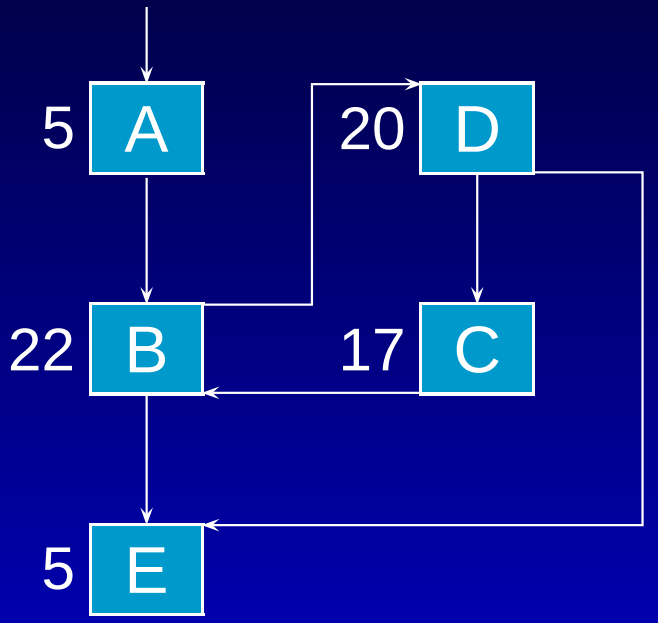
```
int f (int a, int b) {
    int k;
    for (k = 0; k < b; k++) {
        a += k*k;
        if (a > b)
            break;
        a *= 2;
    }
    return a + k;
}
```



Coverage (Pokrytí)

Block Coverage

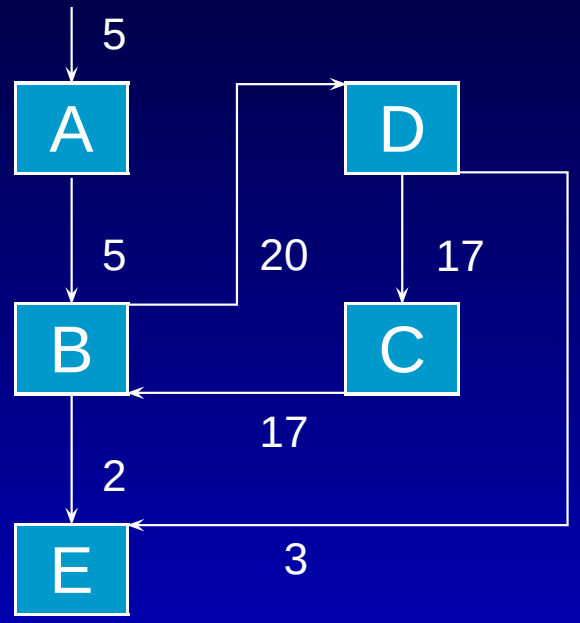
```
int f (int a, int b) {
    int k;
    for (k = 0; k < b; k++) {
        a += k*k;
        if (a > b)
            break;
        a *= 2;
    }
    return a + k;
}
```



Coverage (Pokrytí)

Branch Coverage

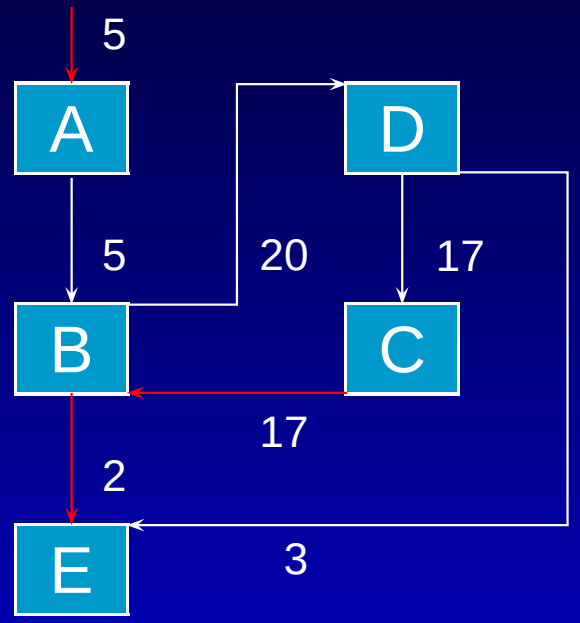
```
int f (int a, int b) {
    int k;
    for (k = 0; k < b; k++) {
        a += k*k;
        if (a > b)
            break;
        a *= 2;
    }
    return a + k;
}
```



Coverage (Pokrytí)

Branch Coverage – Instrumentace

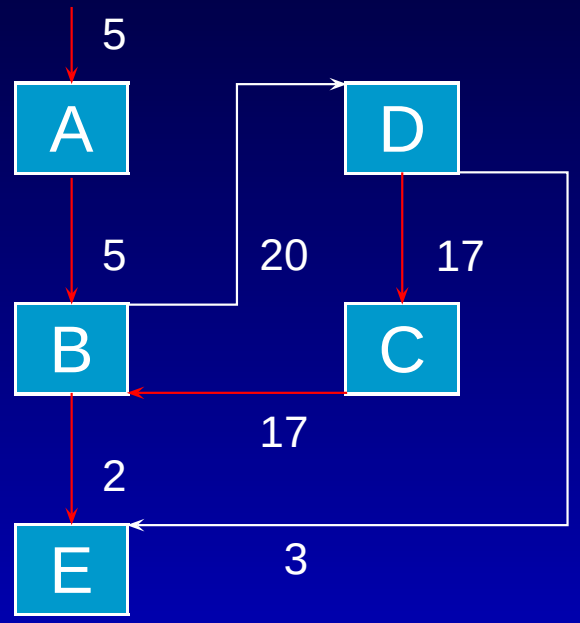
```
int f (int a, int b) {
    int k;
    for (k = 0; k < b; k++) {
        a += k*k;
        if (a > b)
            break;
        a *= 2;
    }
    return a + k;
}
```



Coverage (Pokrytí)

Branch Coverage – Instrumentace

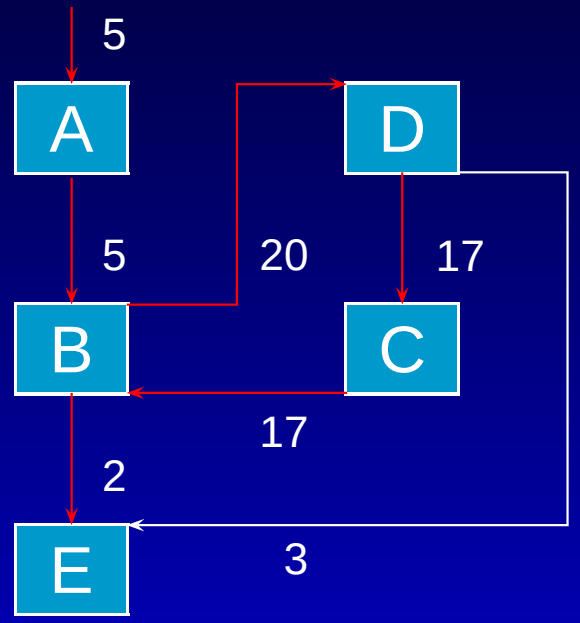
```
int f (int a, int b) {
    int k;
    for (k = 0; k < b; k++) {
        a += k*k;
        if (a > b)
            break;
        a *= 2;
    }
    return a + k;
}
```



Coverage (Pokrytí)

Branch Coverage – Instrumentace

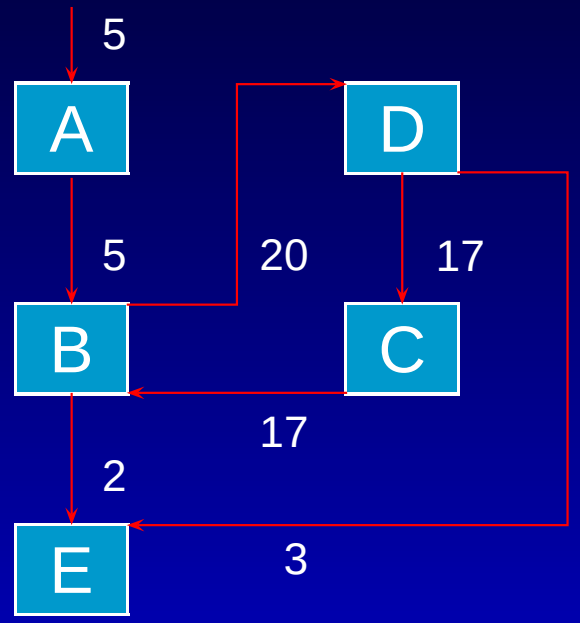
```
int f (int a, int b) {
    int k;
    for (k = 0; k < b; k++) {
        a += k*k;
        if (a > b)
            break;
        a *= 2;
    }
    return a + k;
}
```



Coverage (Pokrytí)

Branch Coverage – Instrumentace

```
int f (int a, int b) {
    int k;
    for (k = 0; k < b; k++) {
        a += k*k;
        if (a > b)
            break;
        a *= 2;
    }
    return a + k;
}
```



Příklady využití

- Predikce skoků
  - Přerovnáním kódu
  - Operandy instrukcí

Příklady využití

- Predikce skoků
  - Přerovnáním kódu
  - Operandy instrukcí
- Threshold pro provádění optimalizací

Nejpodstatnější optimalizace

- Optimalizátor skoků
  - Jump Threading
  - Cross Jumping
  - Odstranění nedosažitelného kódu a návěští
  - Spojování basic bloků

Nejpodstatnější optimalizace

- Optimalizátor skoků
- Register Allocator
  - Použití četností pro priority
  - Webizer
  - Register Coalescing

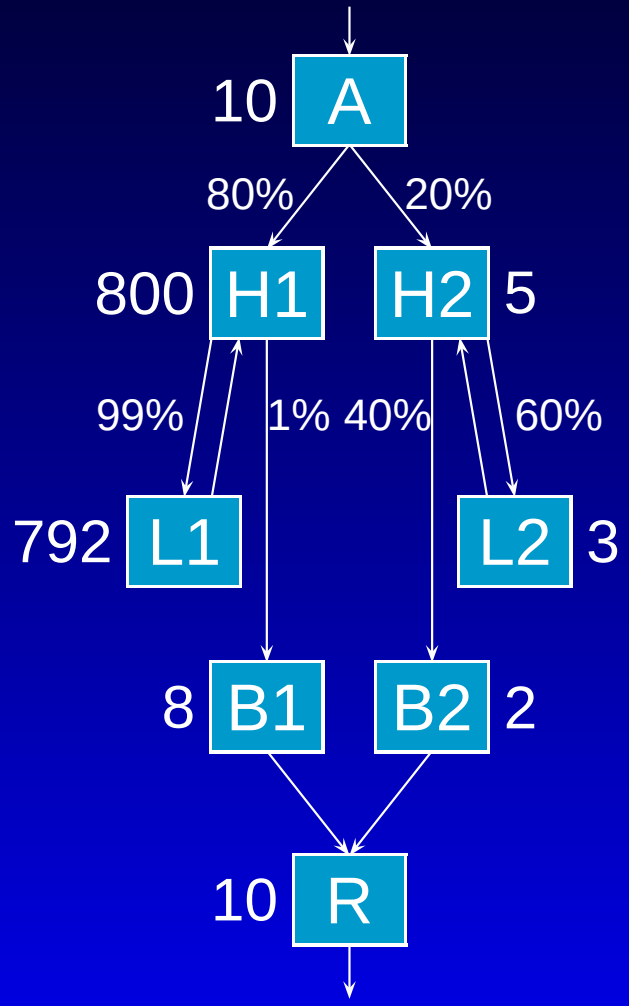
Nejpodstatnější optimalizace

- Optimalizátor skoků
- Register Allocator
  - Použití četností pro priority
  - Webizer
  - Register Coalescing
- Loop Optimizer
  - Loop Unrolling
  - Loop Peeling
  - Loop Unswitching

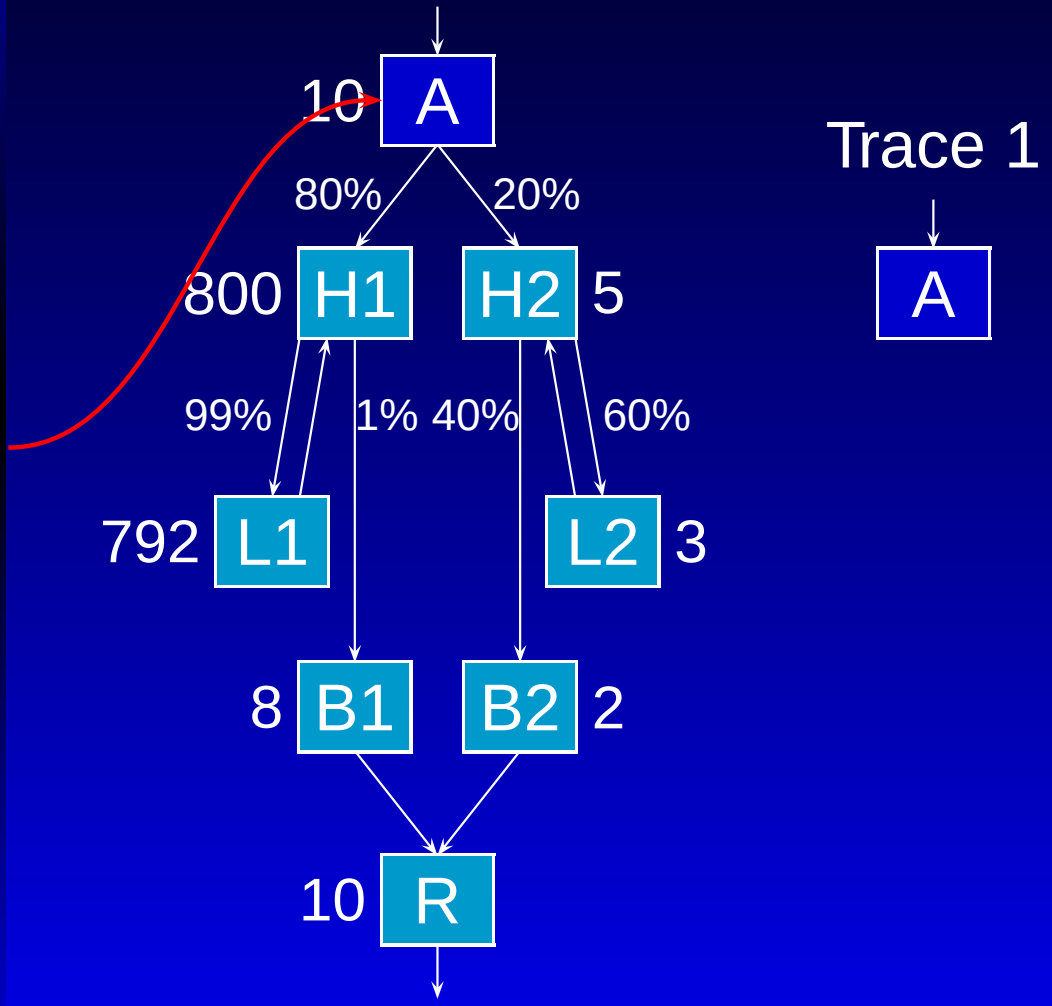
Nejpodstatnější optimalizace

- Optimalizátor skoků
- Register Allocator
  - Použití četností pro priority
  - Webizer
  - Register Coalescing
- Loop Optimizer
  - Loop Unrolling
  - Loop Peeling
  - Loop Unswitching
- Software Trace Cache

Software Trace Cache

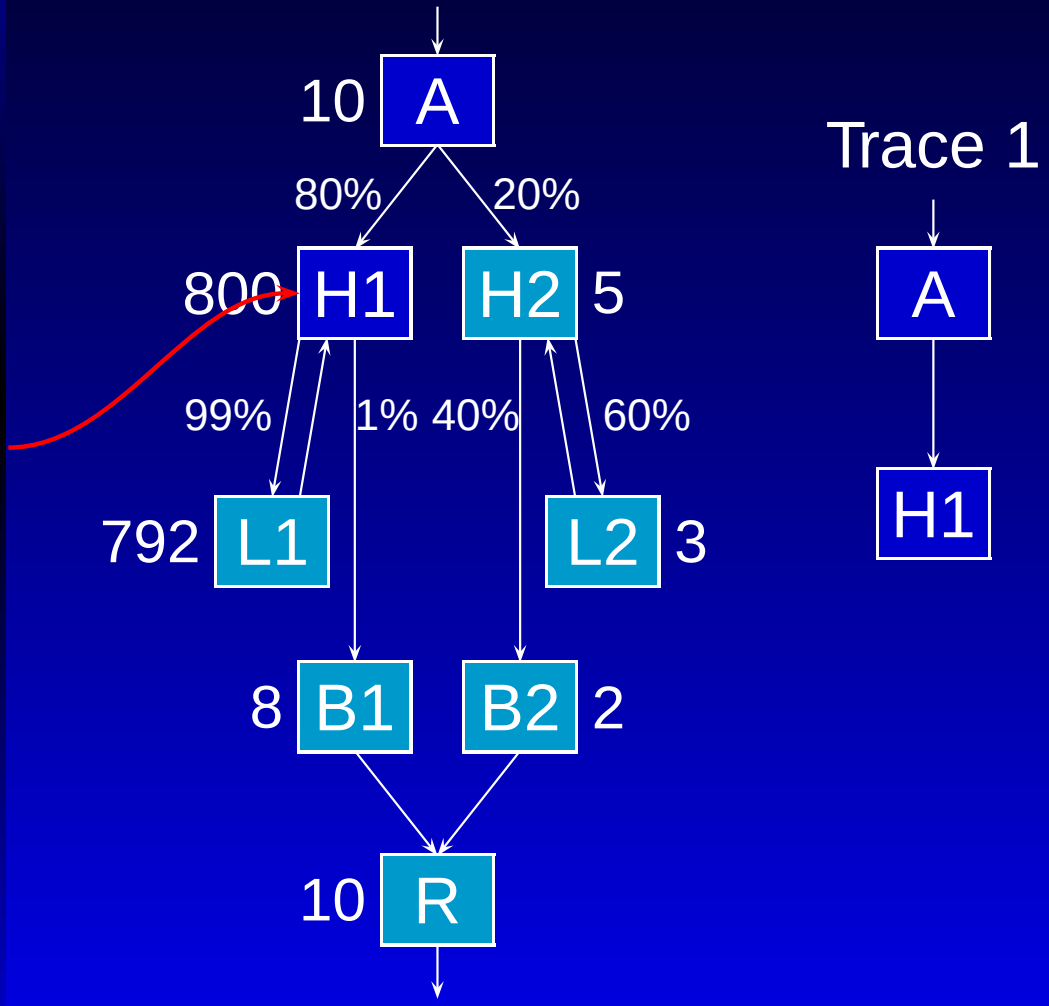


Software Trace Cache



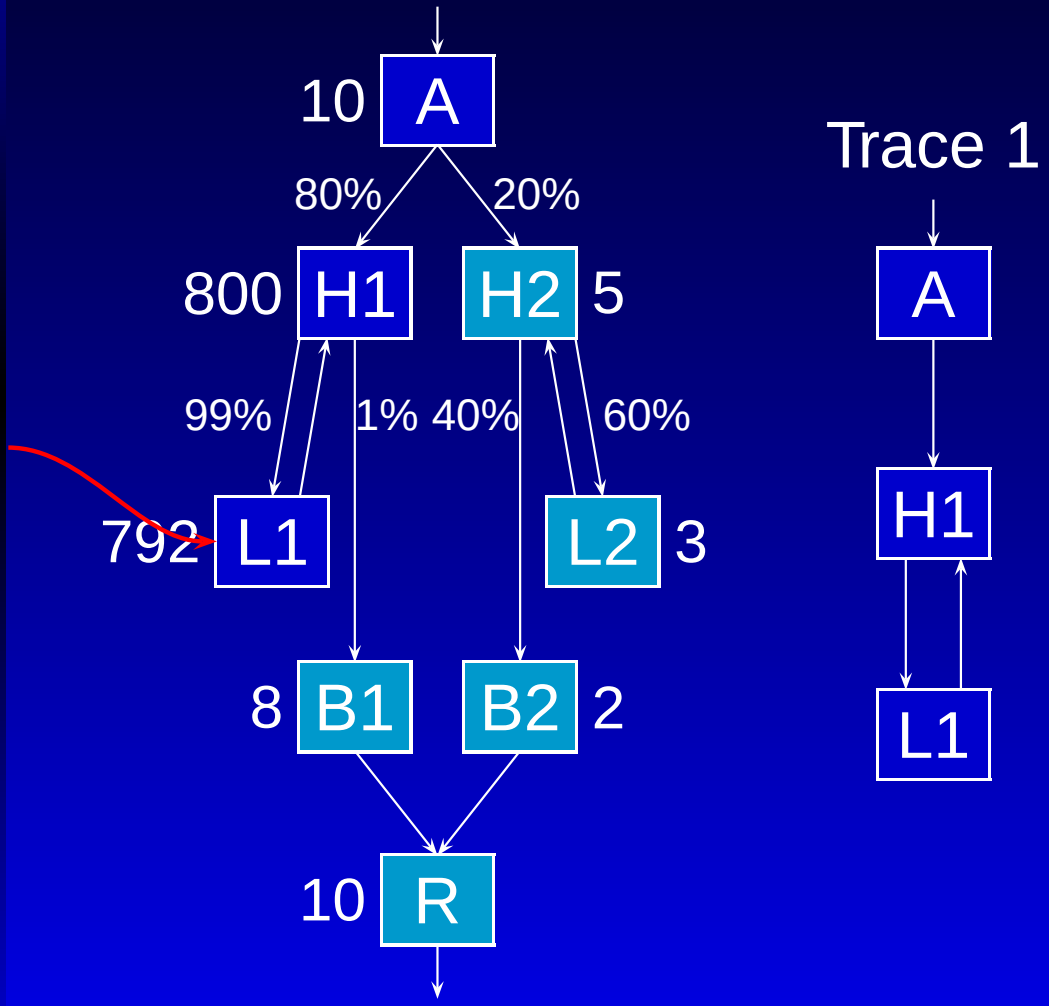
Trace 1

Software Trace Cache

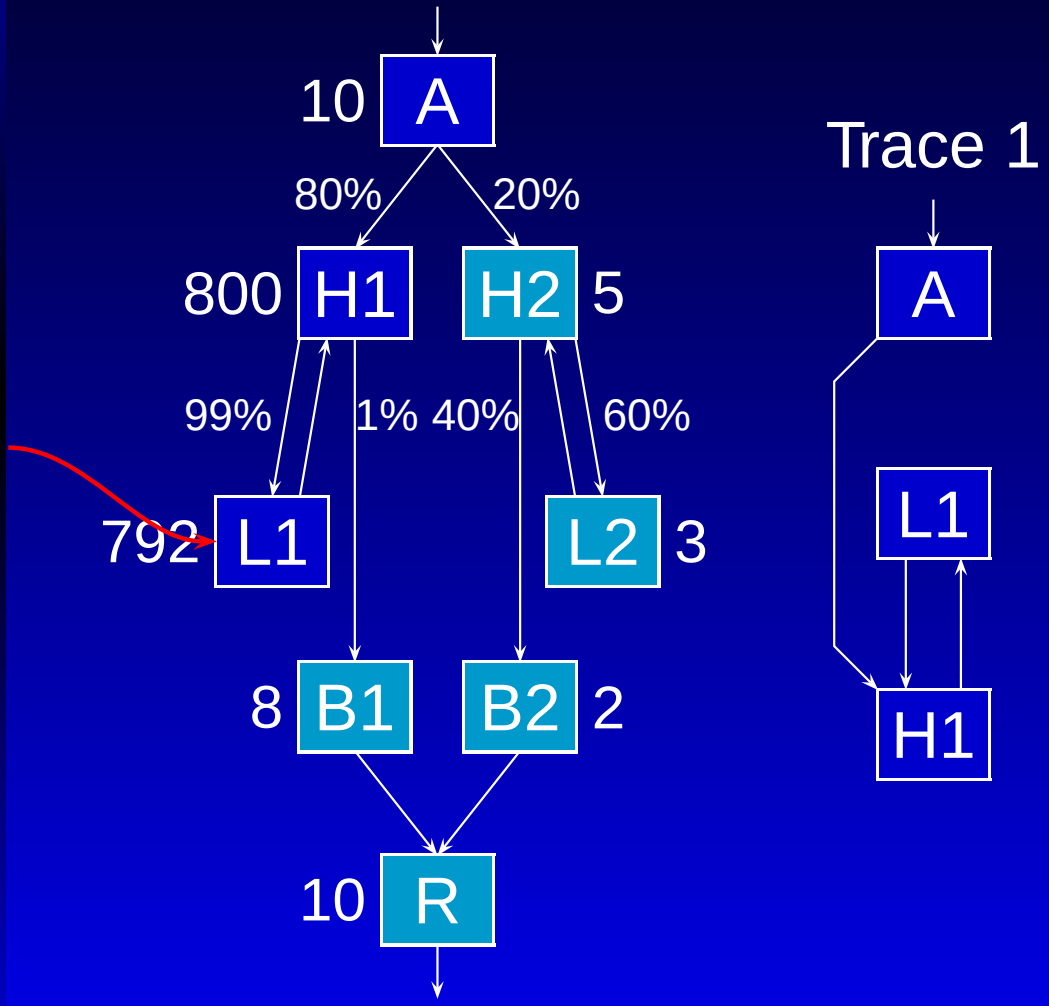


Trace 1

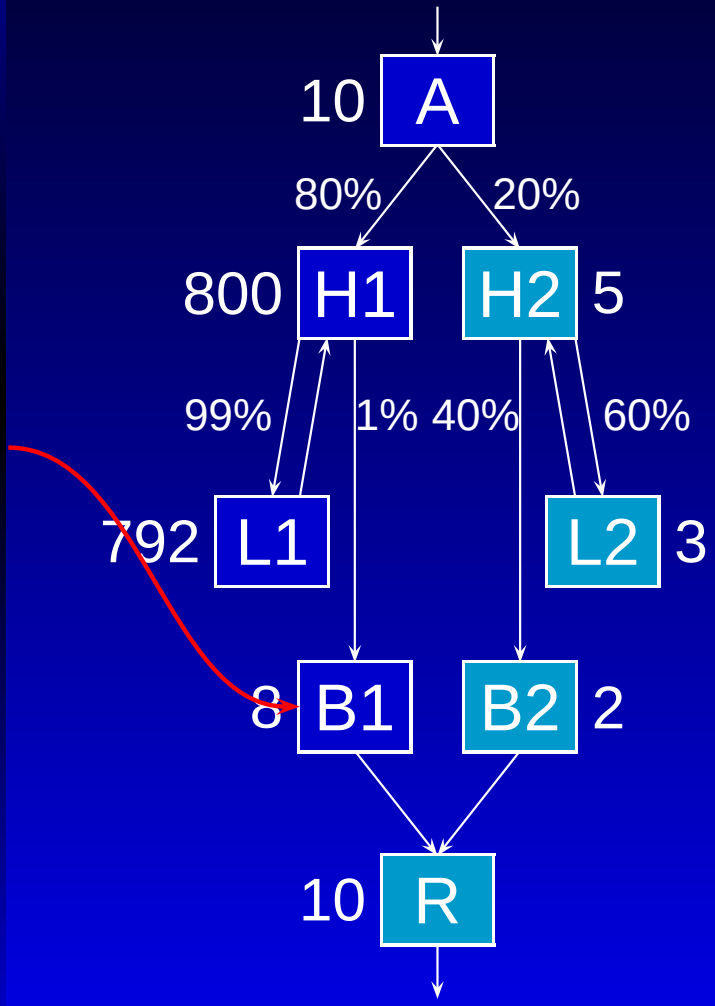
Software Trace Cache

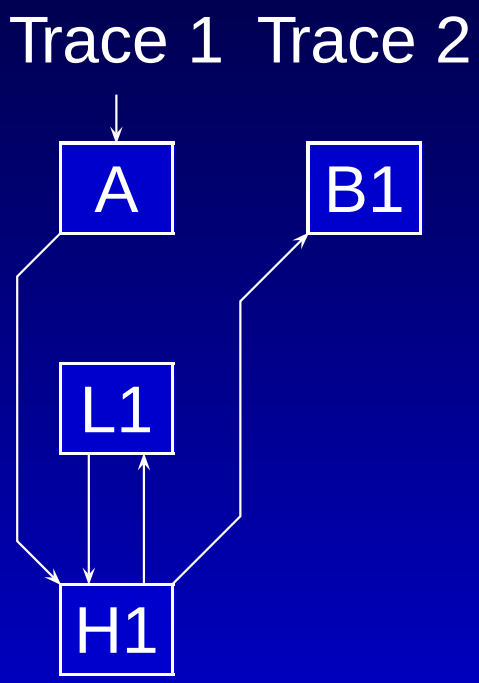


Software Trace Cache

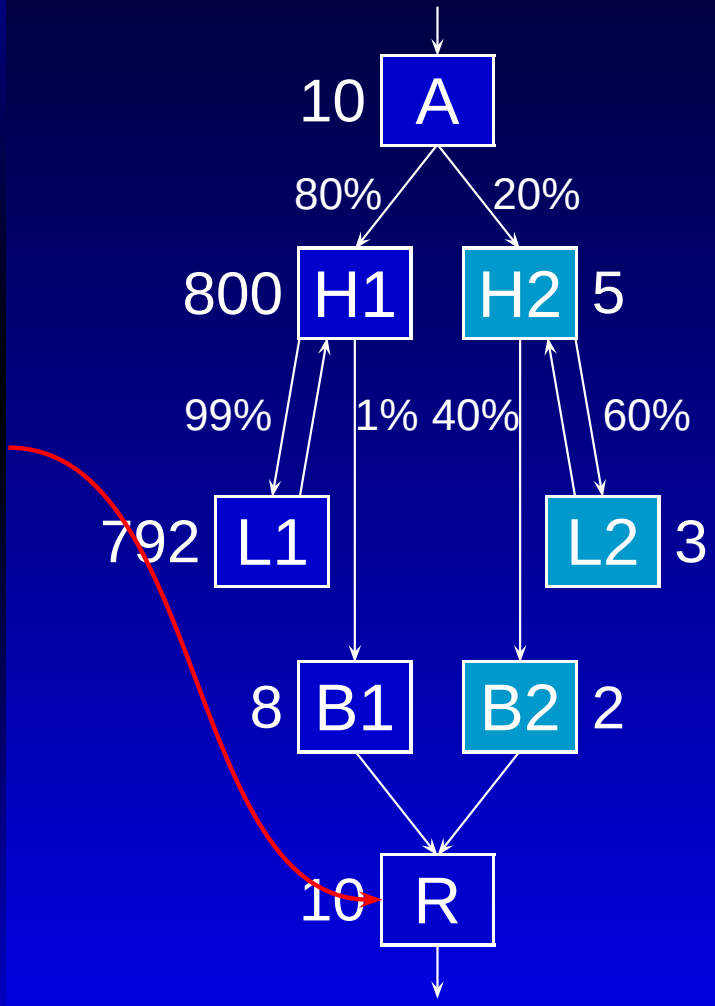


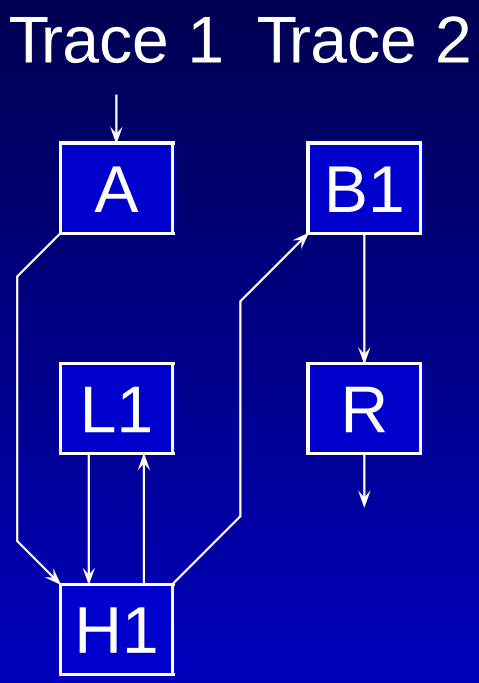
Software Trace Cache



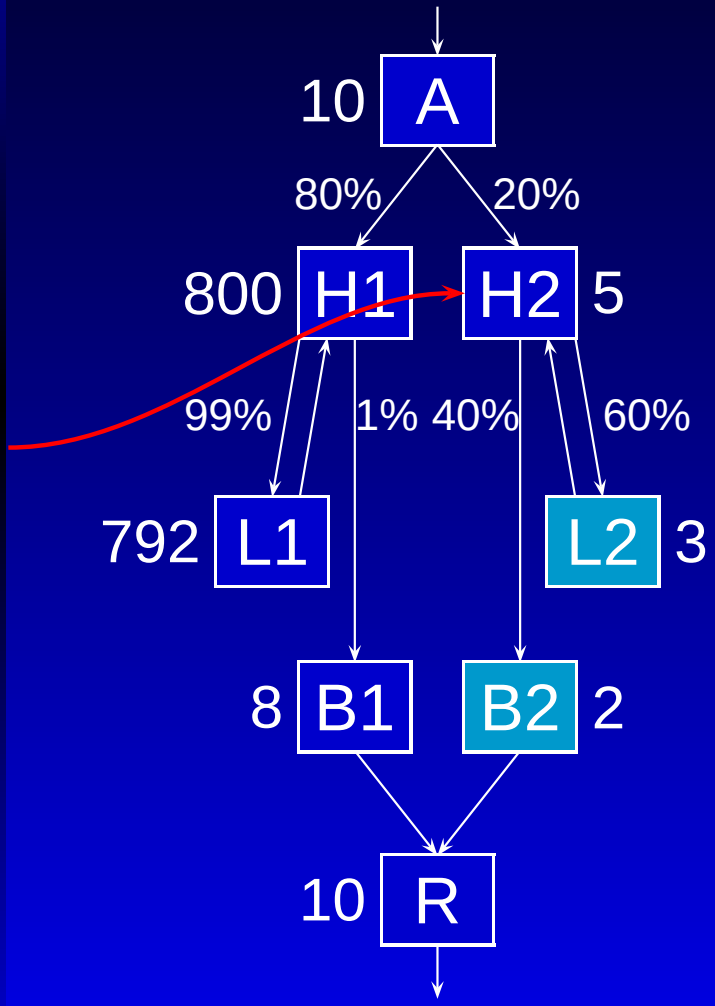


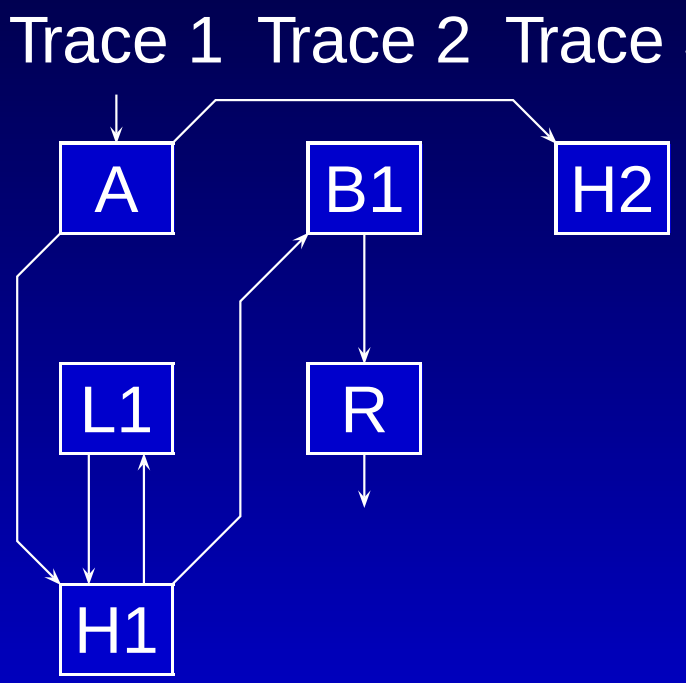
Software Trace Cache



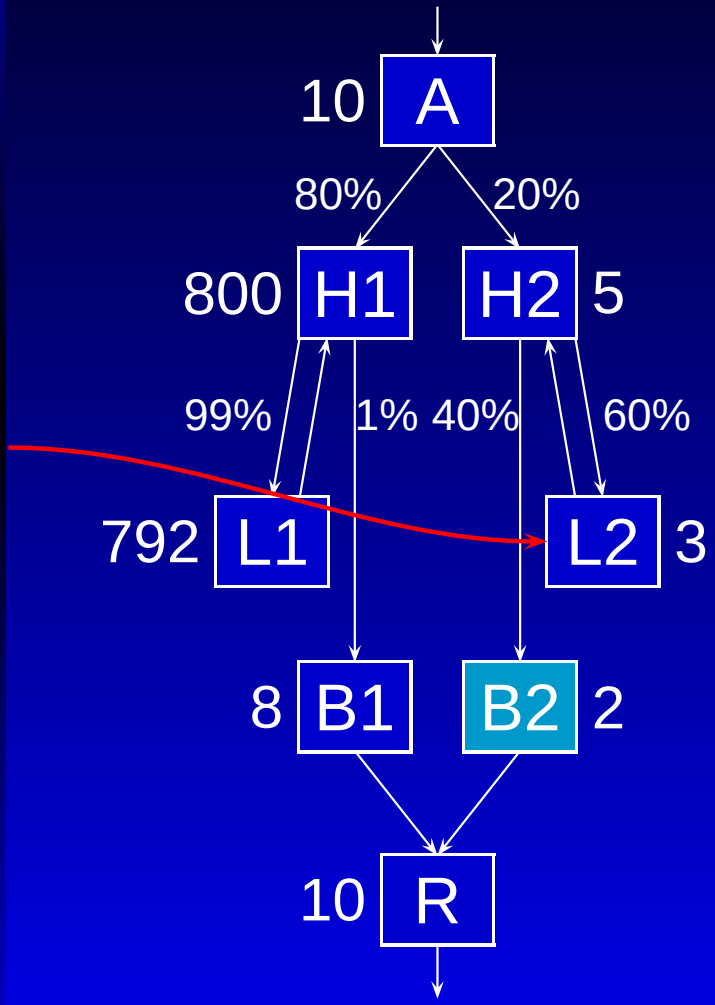


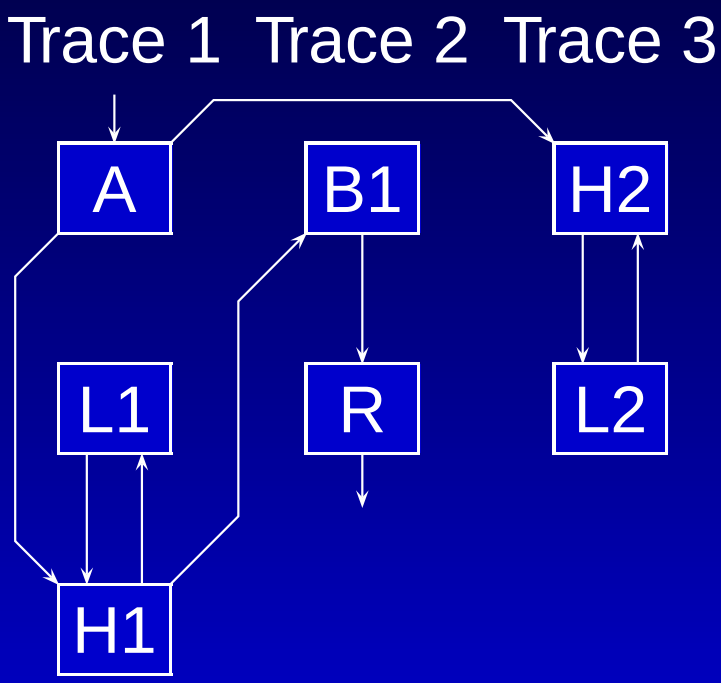
Software Trace Cache



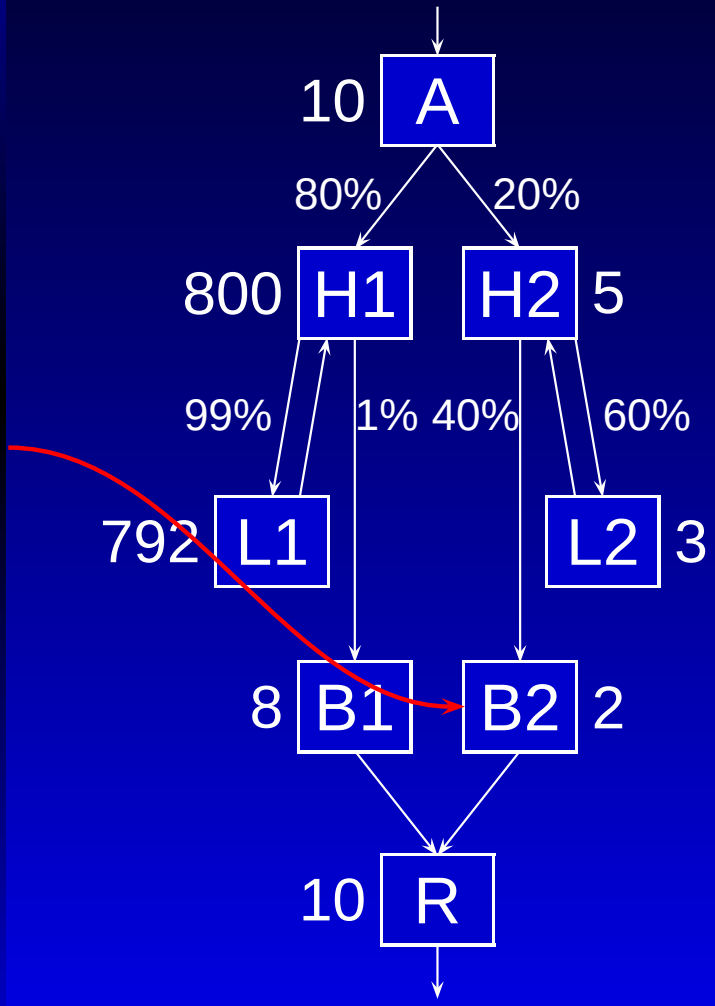


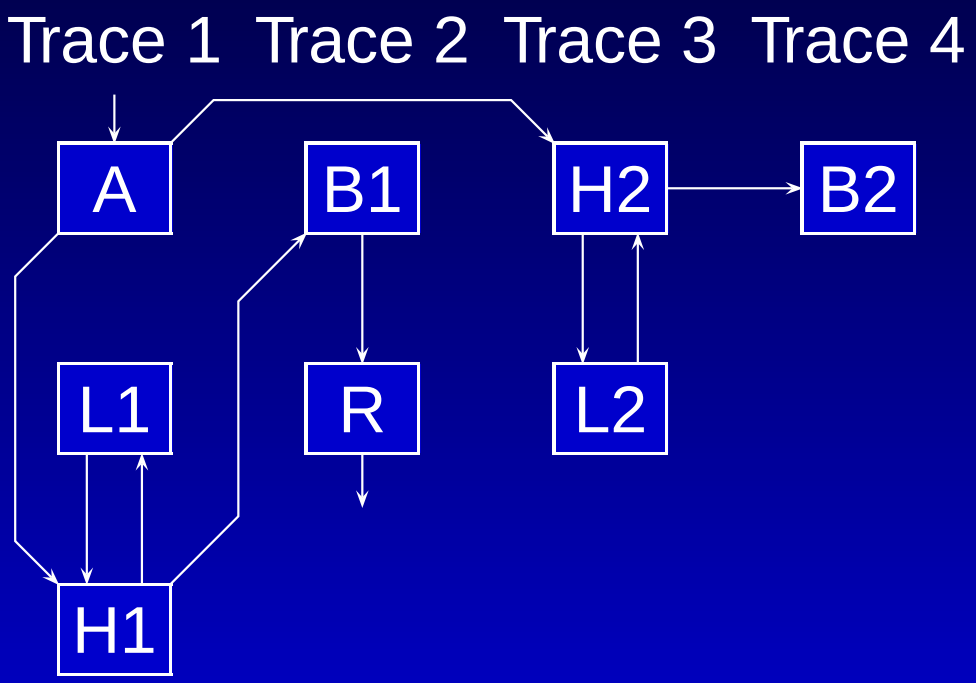
Software Trace Cache



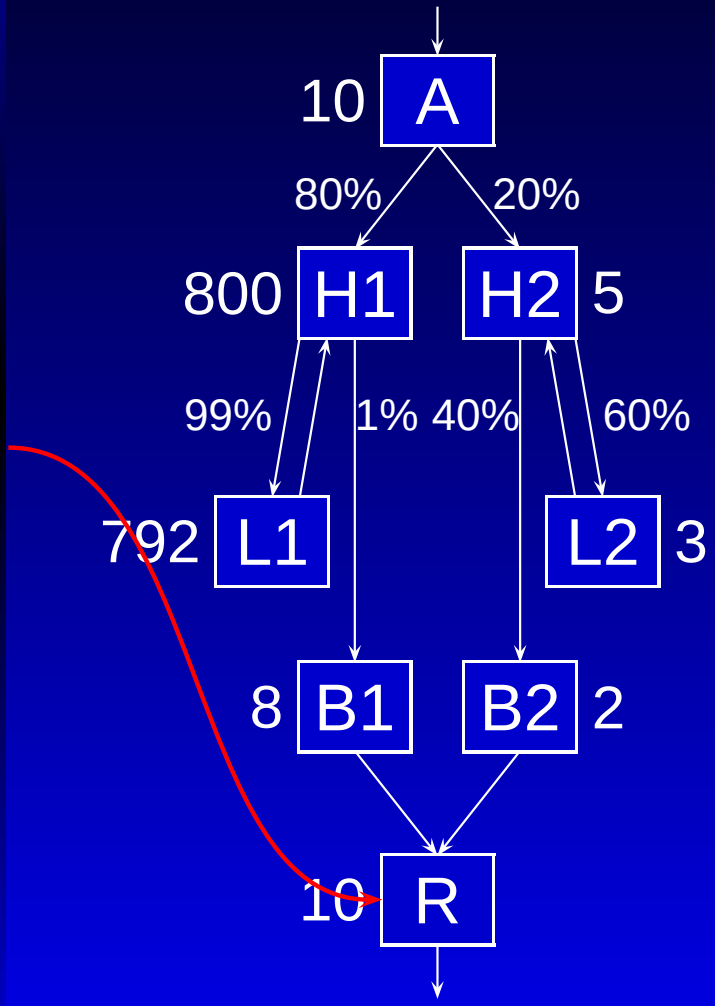


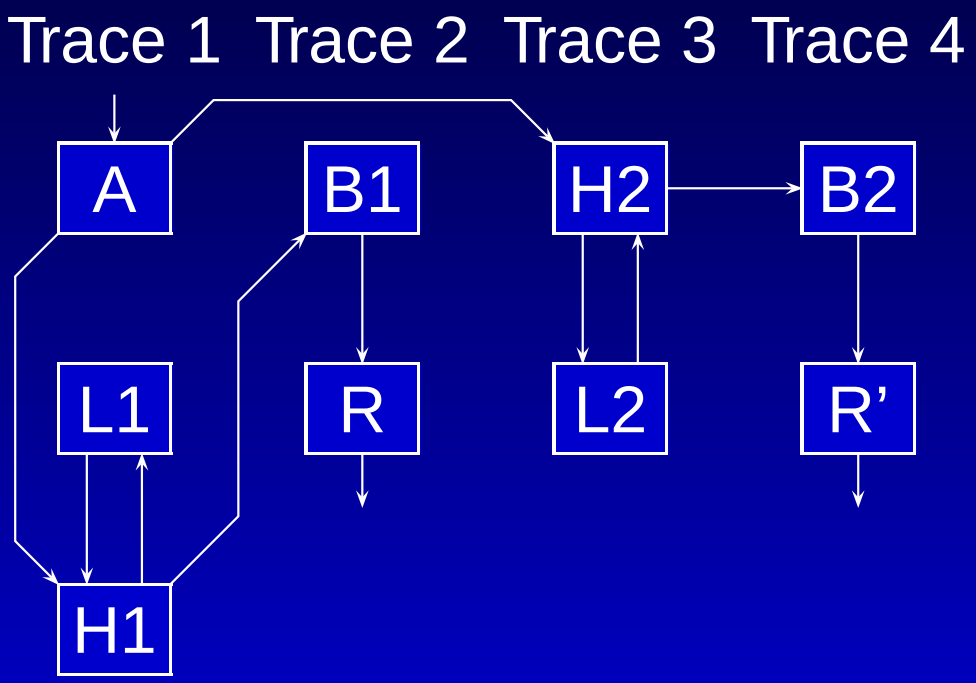
Software Trace Cache





Software Trace Cache





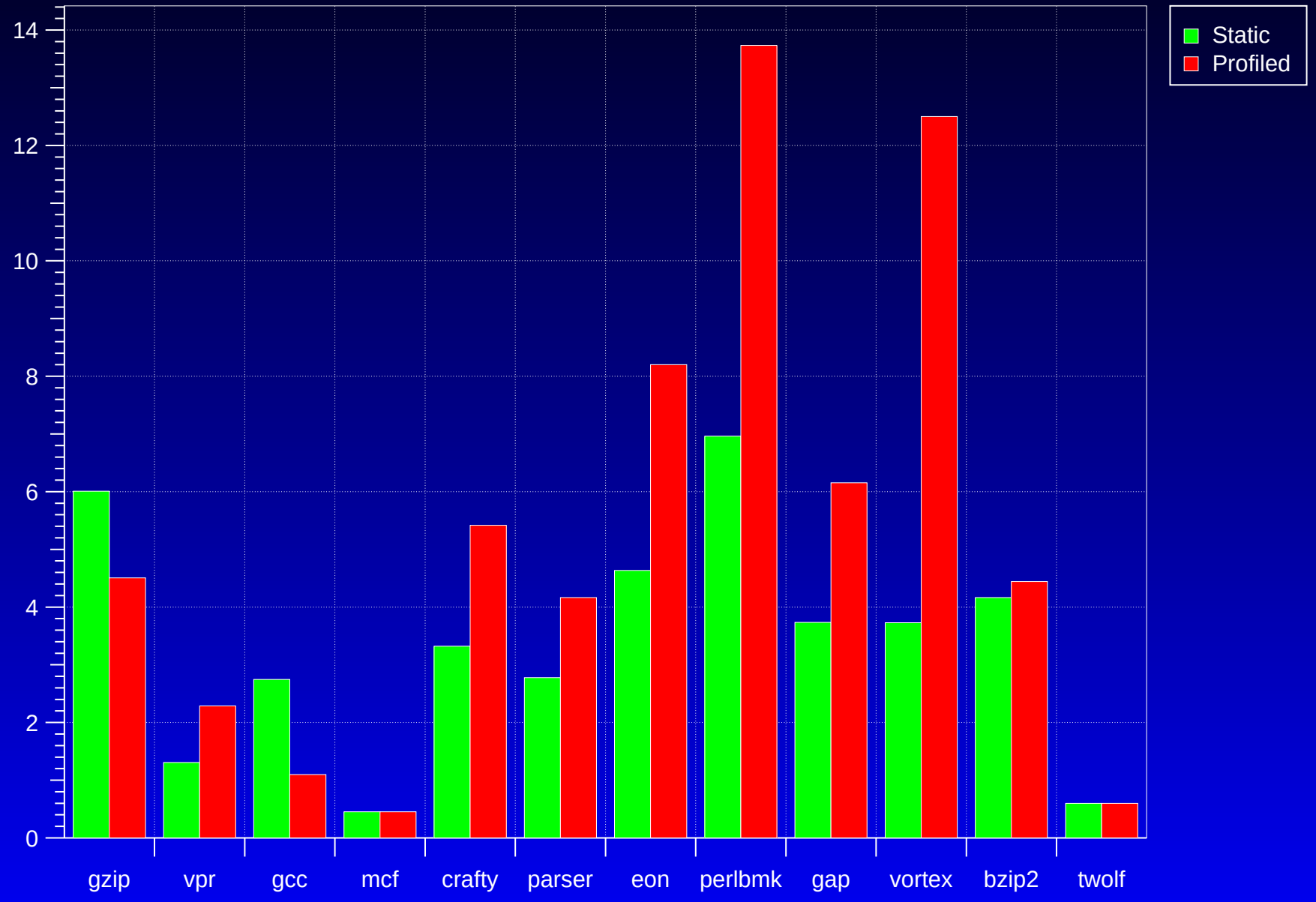
SPECint2000

- Testováno Andreasem Jaegerem na počítači s procesorem AMD Athlon 1.133Ghz

SPECint2000

- Testováno Andreasem Jaegerem na počítači s procesorem AMD Athlon 1.133Ghz
- 77% správně predikovaných skoků s odhadem
- 91% správně predikovaných skoků s profilem

SPECint2000



SPECint2000

- 1.3% zrychlení bez profilu
- 4.1% s profilem

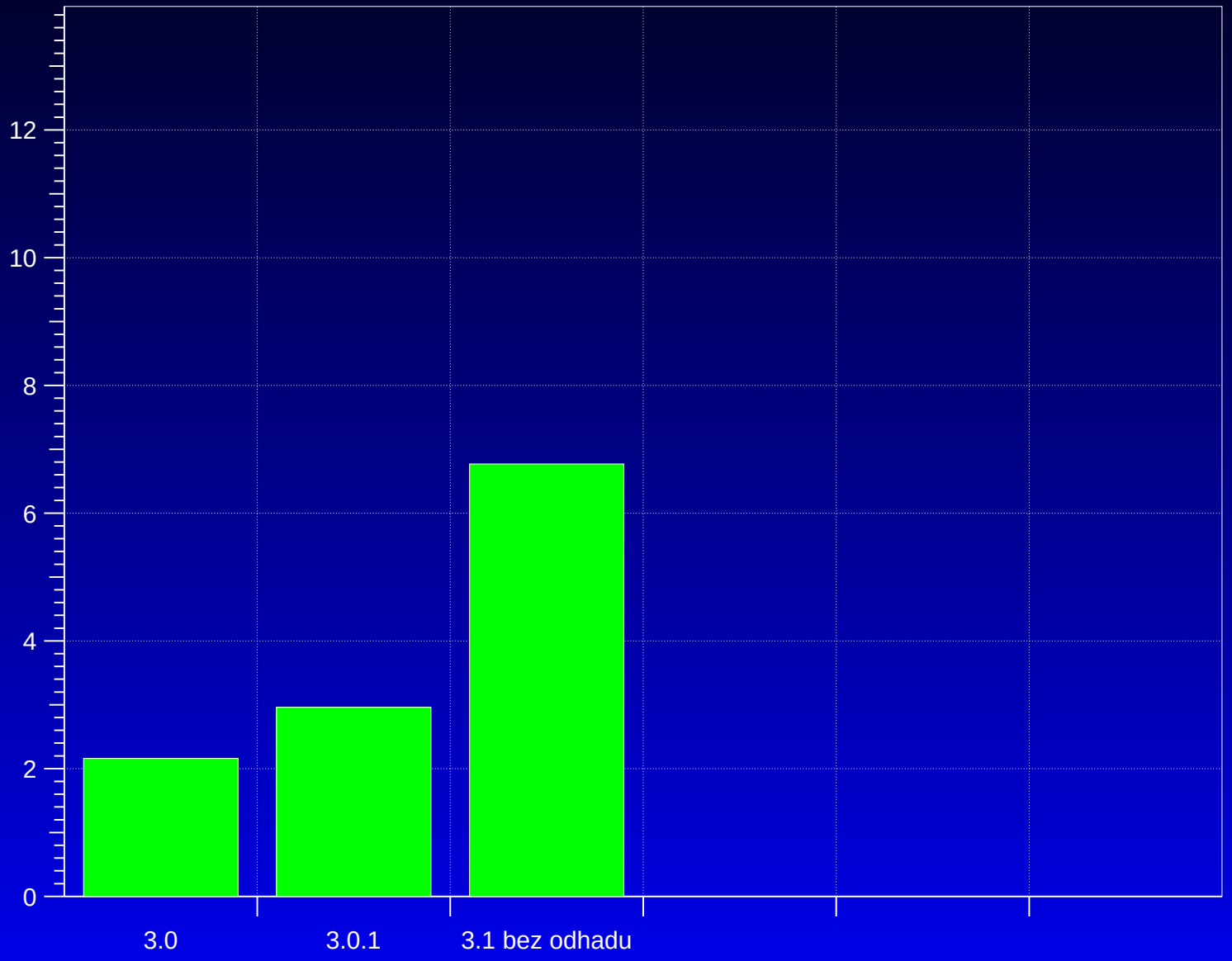
SPECint2000

- 1.3% zrychlení bez profilu
- 4.1% s profilem
- Athlon je navržen pro běh programů určených pro jinou architekturu
  - Překlad i386 instrukcí do mikroinstrukcí
  - Spekulativní on-chip scheduler
  - Velká nezávislost na kvalitě překladače

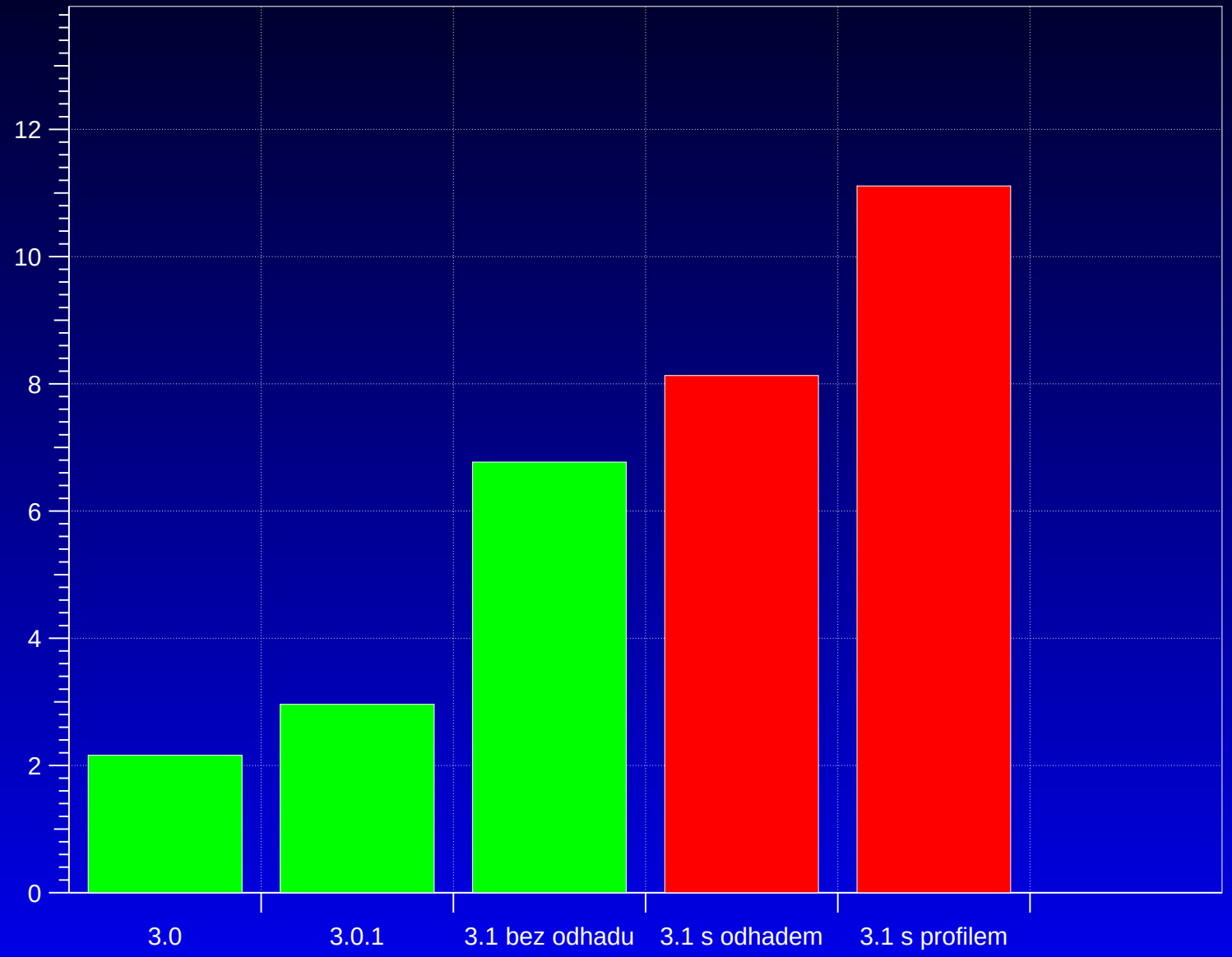
SPECint2000

- 1.3% zrychlení bez profilu
- 4.1% s profilem
- Athlon je navržen pro běh programů určených pro jinou architekturu
- SPEC2000 jsou navrženy pro porovnávání výkonu hardware
  - Snadno optimalizovatelné testy byly vyloučeny
  - Většina testů je limitována propustností paměti
  - Komplikované nebo dobře zoptimalizované interní smyčky

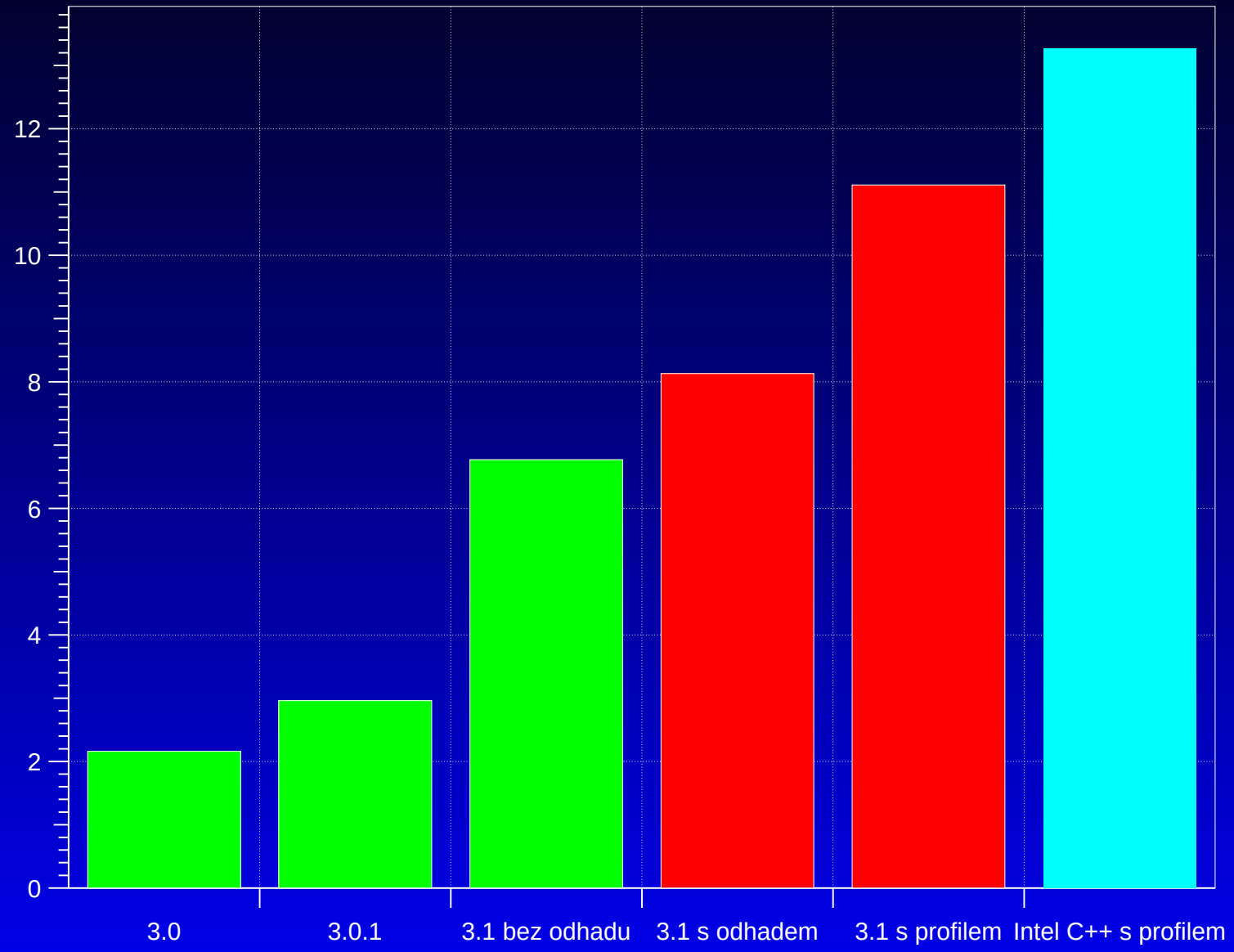
Srovnání s GCC 2.95.3



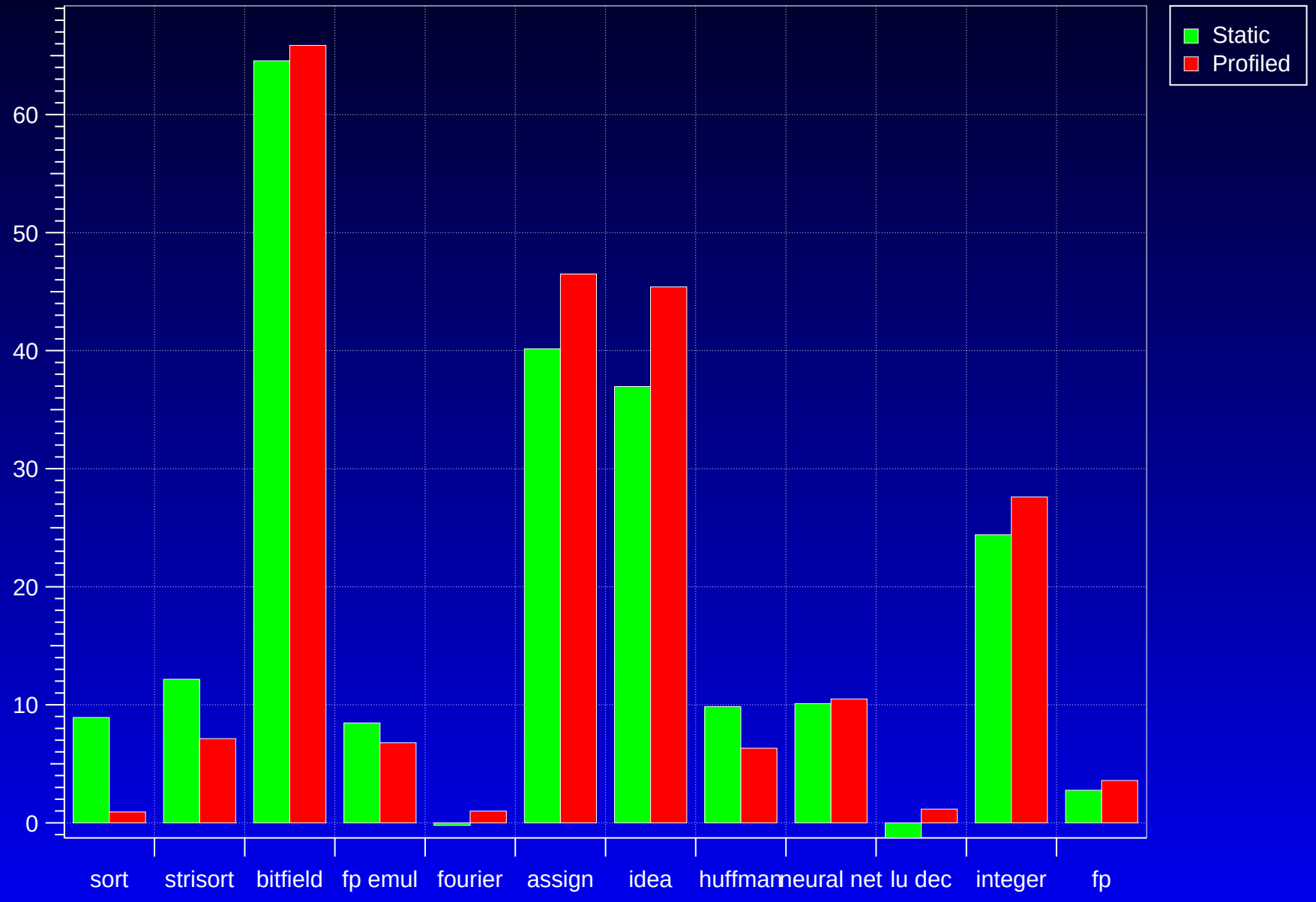
Srovnání s GCC 2.95.3



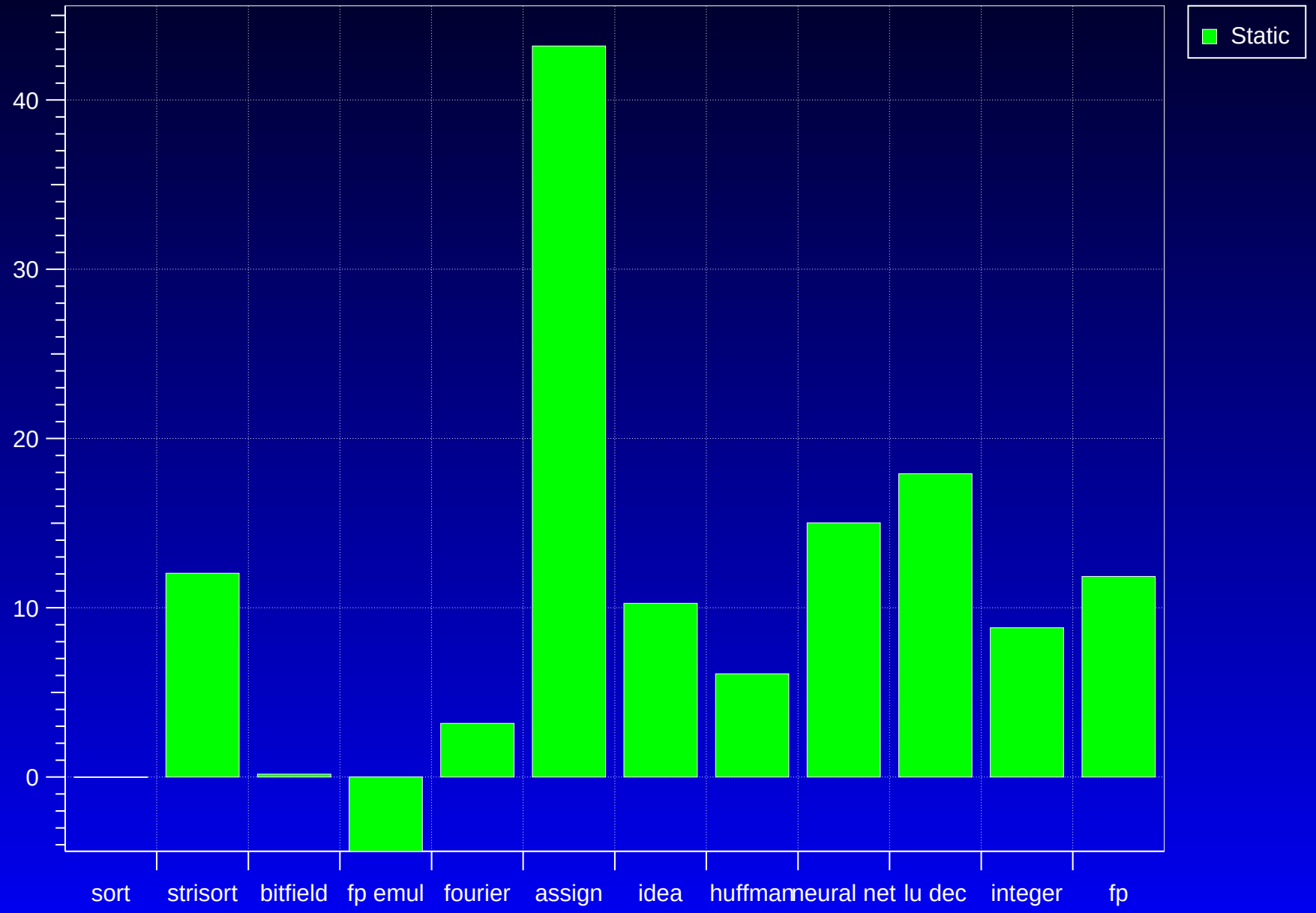
Srovnání s GCC 2.95.3



Byte benchmark, Athlon



Byte benchmark, Sparc



Nejpodstatnější optimalizace

- Athlon
- Sparc

Nejpodstatnější optimalizace

- Athlon
  - Alokace registrů
  - Software trace cache
  - Loop unrolling jednoduchých smyček
- Sparc

Nejpodstatnější optimalizace

- Athlon
  - Alokace registrů
  - Software trace cache
  - Loop unrolling jednoduchých smyček
- Sparc
  - Formace superbloků
  - Trace scheduling
  - Přerovnávání funkcí

Nejpodstatnější optimalizace

- Athlon
  - Alokace registrů
  - Software trace cache
  - Loop unrolling jednoduchých smyček
- Sparc
  - Formace superbloků
  - Trace scheduling
  - Přerovnávání funkcí
- Itanium?