Формални Езици и Езикови Процесори
ТУ, кат. КС, летен семестър 2012

Лекция 21

Тема:

Оптимизация на кода

Съдържание:

- Критерии за Code-Improving Transformations
- Getting better performance
- Peephole optimization
- Principal sources of optimizations
  - Function-preserving transformations
- Демо програми: 1adrstbo.bas,3adrstbo.bas

Въведение

- Ideally, компилаторите се очаква да генерират код така добър като ръчно написан.
- Reality е, че тази цел не се постига.
- Все пак, кодът генериран от комилатор, може да се подобри по бързодействие или по памет, или и по двете х-ки.
- Такива подобрения са резултат на програмни трансформации, наречени най-общо оптимизация на кода.

Видове оптимизация

- Машинно независима оптимизация
  - Program transformations that improve target code without taking into consideration any properties of the target machine (processor).
- Машинно зависима оптимизация
  - Program transformations that improve target code with taking into consideration specific properties of the target machine (processor).

Критерии за Code-Improving Transformations

Първо, трансформацията следва да съхрани значението/смисъла на оригиналната програма.

Критерии за Code-Improving Transformations

Второ, трансформацията следва да подобри значимо бързодействието на програмата.

Също интерес представлява и редуцирането размера на обектния код, но понастоящем размерът на кода не е така значим фактор както преди време.

Критерии за Code-Improving Transformations

Трето, трансформацията следва да си струва усилията.

Към по-добри характеристики

По принцип, подобрения се осъществяват на разни нива – от първичен текст до обектен код.

Source code	front end	intermediate code	code gen	target code

User can:	Compiler can:	Compiler can:
profile program	improve loops	use registers
change algorithm	procedure calls	select instructions
transform loops	address calculations	do peephole opt

Peephole optimization

- Стратегия stmt by stmt за генериране на код често ражда код с излишни инструкции и неоптимални конструкти.
- Техника за locally improving the target code се нарича peephole optimization. Това е метод за подобрение на кода чрез разглеждане на къси последователности от команди, наречени peephole, и замяната им с по-кратки или по-бързи такива.
- Peephole е като малък движещ се прозорец за разглеждане на обектната програма.

Peephole optimization

- Redundant instruction elimination (Loads and Stores)
- Ако има следната последователност

```
mov  Reg0,a
mov  a,Reg0
```

  То втората инструкция следва да отпадне
- Ако има следната последователност

```
sta  memory
lda  memory
```

  То и двете инструкции следва да отпаднат

Peephole optimization

- Unreachable Code
  - Неетикетирана инструкция следваща безусловен преход следва да се отстрани.

```
        goto lab
        codeop1   operand
        . . .
lab:  codeop2   operand
```

Peephole optimization

- Flow of Control Optimizations
  - Някои алгоритми за генерация на междинен код продуцират jumps to jumps, jumps to conditional jumps, or conditional jumps to jumps
  - Възможно е да се замени

```
    goto L1                    goto L2
    . . .                      . . .
L1: goto L2                L1: goto L2
```

Peephole optimization

- Algebraic Simplification
  - Нямат край възможните алгебрични опростявания. Например, оператори като

```
x := x + 0
x := x * 1
```

    се създават от code generation algorithms и могат леко да се заменят с по-прости

Peephole optimization

- Reduction in Strength. Замяна на expensive operations с еквивалентни cheaper ones за целевия компютър.
- Например, x^2 е cheaper да се имплементира като x*x вместо извикване на ПП за степенуване.

Peephole optimization

- Use of Machine Idioms. The target machine may have HW instructions to implement specific operations efficiently.
- For example, some machines have auto-increment and auto-decrement addressing modes. These add or subtract one from an operand before or after using the value.
- Application: pushing or popping a stack, parameter passing, assignment I := I + 1

Principal sources of optimization

- Една трансформация е локална, ако тя се изпълнява с разглеждане на операторите само в основен блок;
- В противен случай, се говори за глобална трансформация.
- Много трансформации се провеждат и на локално, и на глобално ниво.
- Обикновено първо са локалните трансформации.

Function-preserving transformations

- Общи под изрази
- Common sub expressions

Function-preserving transformations

- Copy propagation

Function-preserving transformations

- Елиминиране на мъртъв код
- Dead-Code Elimination

Function-preserving transformations

- Оптимизация на цикли Loop optimizations
  - Изваждане пред цикъл (loop-invariant computation).

```
while ( I <= limit – 2 )        |       t = limit – 2;
                                |       while(I<=t)
```

Function-preserving transformations

- Оптимизация на цикли Loop optimizations
  - Induction Variables

The Use of algebraic identities

- Arithmetic identities
  - $x + 0 = 0 + x = x$
  - $x - 0 = x$
  - $x * 1 = 1 * x = x$
  - $x / 1 = x$

The Use of algebraic identities

- Reduction in Strength: замяна на expensive операция с cheaper one както в
  - x ** 2 = x * x
  - 2.0 * x = x + x
  - x / 2 = x * 0.5

The Use of algebraic identities

- Constant Folding: да се оцени един константен израз по време на компилация и да се замени с неговата стойност
  - 2 * 3.14 се заменя с 6.28
  - #define PI 3.14
  - PI * 10. * 10. се заменя с 314.

Демо програми

- Демо програми: 1adrstbo.bas,3adrstbo.bas

Благодаря
За
Вниманието