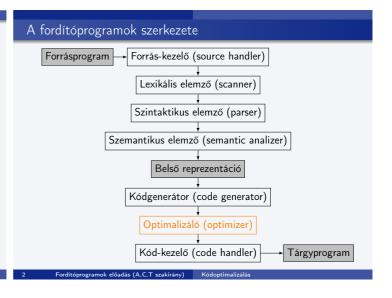
Kódoptimalizálás Fordítóprogramok előadás (A,C,T szakirány)



A szintézis menete "valójában" Optimalizálási lépések végrehajtása az eredeti programon (vagy annak egyszerűsített változatán). Kódgenerálás. Gépfüggő optimalizálás végrehajtása a generált kódon. A kódoptimalizálás célja Hatékonyabb program létrehozása: nagyobb sebesség kisebb méret Ezek a célok időnként ellentmondanak egymásnak!

Követelmény a kódoptimalizálással szemben

Fordítóprogramok előadás (A.C.T szakirány) Kódoptimalizála

- "Az optimalizált programnak ugyanúgy kell működnie, mint az eredetinek." Ez sok mindent jelenthet!
 - ugyanarra a bemenetre ugyanazt a kimenetet adja?
 - eseményvezérelt környezetben ugyanúgy viselkedik?
 - párhuzamos környezetben ugyanúgy viselkedik?
 - stb.
- Az adott nyelv szemantikája (jelentése) dönti el, hogy egy optimalizálási lépés megengedhető-e.

Kódoptimalizálási lépések osztályozása

- Mit optimalizálunk?
 - az eredeti programot (vagy annak egyszerűsített változatát): itt még vannak ciklusok, elágazások, kifejezéskiértékelés stb.
 - a tárgyprogramot: jellemzően assembly kódot
- Mi az átalakítás hatóköre?
 - lokális: kis programrészletek átalakítása
 - globális: a teljes program szerkezetét kell vizsgálni
- Mit használ ki az átalakítás?
 - általános optimalizálási stratégiák: algoritmusok javítása
 - gépfüggő optimalizálás: az adott architektúra sajátosságait használja ki

ok előadás (A,C,T szakirány)

Kódoptimalizálá

Fordítóprogramok előadás (A,C,T szakirány)

Kódoptimalizálás

Lokális optimalizálás

Lokális optimalizálás: alapblokk fogalma

Definíció: alapblokk

Egy programban egymást követő utasítások sorozatát alapblokknak nevezziik ha

- az első utasítás kivételével egyik utasítására sem lehet távolról átadni a vezérlést
 - (assembly programokban: ahová a jmp, call, ret utasítások "ugranak"; magas szintű nyelvekben: eljárások, ciklusok eleje, elágazások ágainak első utasítása, goto utasítások célpontjai)
- az utolsó utasítás kivételével nincs benne vezérlés-átadó utasítás
 - (assembly programban: jmp, call, ret magas szintű nyelvekben: elágazás vége, ciklus vége, eljárás vége, goto)
- az utasítás-sorozat nem bővíthető a fenti két szabály megsértése nélkül

Alapblokk szerepe az optimalizálásban

Lokális ontimalizálás

- A definíció következményei:
 - egy alapblokknak pontosan egy belépési pontja van (az első utasítás)
 - az utasításai szekvenciálisan hajtódnak végre
 - ha rá kerül a vezérlés, akkor az összes utasítása pontosan egyszer hajtódik végre
- Lokális optimalizálás: egy alapblokkon belüli átalakítások

Fordítóprogramok előadás (A,C,T szakirány) Kódoptimalizálás

Alapblokkok meghatározása

- Jelöljük meg:
 - a program első utasítását
 - azokat az utasításokat, amelyekre távolról át lehet adni a
 - a vezérlés-átadó utasításokat követő utasításokat
- Minden megjelölt utasításhoz tartozik egy alapblokk, ami a következő megjelölt utasításig (vagy az utolsó utasításig) tart.

Fordítóprogramok előadás (A,C,T szakirány) Kódoptimalizálá:

Alapblokkok: példa

```
main: mov eax,[Cimke1]
     cmp eax, [Cimke2]
      jz igaz
      dec dword [Cimke1]
     inc dword [Cimke2]
     jmp vege
igaz: inc dword [Cimke1]
      dec dword [Cimke2]
vege: ret
```

Tömörítés

- Cél: minél kevesebb konstans és konstans értékű változó legven!
- konstansok összevonása: a fordítási időben kiértékelhető kifejezések kiszámítása

Eredeti kód

Optimalizált kód

• konstans továbbterjesztése: a fordítási időben kiszámítható értékű változók helyettesítése az értékükkel

Eredeti kód

Optimalizált kód

c := 8;

Azonos kifejezések többszöri kiszámításának elkerülése

Eredeti kód

$$x := 20 - (a * b);$$

 $y := (a * b) ^ 2;$

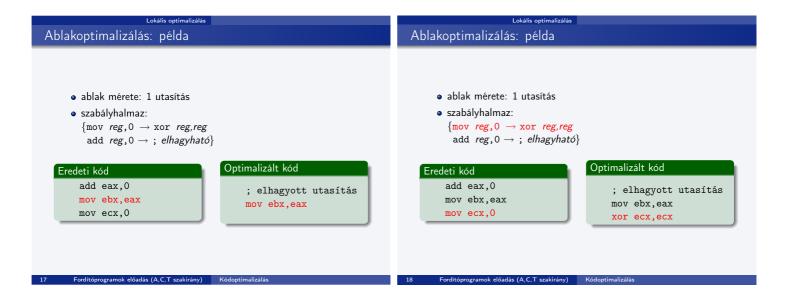
Optimalizált kód

t := a * b;x := 20 - t;

- Ez csak látszólag növeli a program méretét!
 - az a*b kifejezés kiértékelése is több assembly utasítás
 - a t változó lehet egy regiszter is
- Megvalósítás a gyakorlatban:
 - az utasításokból egy címkézett, irányított körmentes gráfot építünk,
 - ebből generálható az optimalizált kód

Lokális optimalizálás Változó továbbterjesztése Ablakoptimalizálás • Ez egy módszer a lokális optimalizálás egyes fajtáihoz. Eredeti kód Optimalizált kód Ablak: x := a + b;x := a + b;• egyszerre csak egy néhány utasításnyi részt vizsgálunk a kódból y := x; y := x; • a vizsgált részt előre megadott mintákkal hasonlítjuk össze z := y; z := x; • ha illeszkedik, akkor a mintához megadott szabály szerint átalakítjuk • Ha az y változóra a továbbiakban már nincs szükség, ezt az "ablakot" végigcsúsztatjuk a programon akkor y := x törölhető! (De ez a törlés már globális optimalizálás...) Az átalakítások megadása: • Ez is megoldható az előzőleg említett gráfos módszerrel. • { minta → helyettesítés} szabályhalmazzal • a mintában lehet paramétereket is használni Fordítóprogramok előadás (A,C,T szakirány) Kódoptimalizálás Fordítóprogramok előadás (A,C,T szakirány) Kódoptimalizálás

Ablakoptimalizálás: példa Ablakoptimalizálás: példa • ablak mérete: 1 utasítás • ablak mérete: 1 utasítás szabályhalmaz: szabályhalmaz: $\{\text{mov } reg, 0 \rightarrow \text{xor } reg, reg\}$ $\{\texttt{mov} \ \textit{reg}, \texttt{0} \ \rightarrow \texttt{xor} \ \textit{reg,reg}$ add $reg, 0 \rightarrow$; elhagyható} add $reg, 0 \rightarrow ; elhagyható$ Optimalizált kód Eredeti kód Eredeti kód Optimalizált kód add eax.0 add eax.0 ; elhagyott utasítás mov ebx, eax mov ebx, eax mov ecx,0 mov ecx,0 Fordítóprogramok előadás (A,C,T szakirány) Kódoptimalizála



Lokális optimalizálás

Tipikus egyszerűsítések ablakoptimalizáláshoz

- Globális optimalizálás
- felesleges műveletek törlése: nulla hozzáadása vagy kivonása
- egyszerűsítések: nullával szorzás helyett a regiszter törlése
- nulla mozgatása helyett a regiszter törlése
- regiszterbe töltés és ugyanoda visszaírás esetén a visszaírás elhagyható
- utasításismétlések törlése: ha lehetséges, az ismétlések törlése
- a teljes program szerkezetét meg kell vizsgálni
- ennek módszere az adatáram-analízis:

 - Mely változók értékeit számolja ki egy adott alapblokk?
 Mely változók értékeit melyik alapblokk használja fel?
- lehetővé teszi:
 - az azonos kifejezések többszöri kiszámításának kiküszöbölését akkor is, ha különböző alapblokkokban szerepelnek
 - konstansok és változók továbbterjesztését alapblokkok között is
 - elágazások, ciklusok optimalizálását

Fordítóprogramok előadás (A,C,T szakirány) Kódoptimalizálás

Fordítóprogramok előadás (A,C,T szakirány) Kódoptimalizálás

Kódkiemelés

Eredeti kód

```
if( x < 10 )
  a = 0;
else
  b--;
  a = 0;
```

Optimalizált kód

```
if(x < 10)
{
  b++;
}
else
  b--;
}
```

Kódsüllyesztés

Eredeti kód

```
if(x < 10)
{
  x = 0;
  b++:
}
else
{
  b--;
  x = 0;
}
```

Optimalizált kód

```
if(x < 10)
  b++;
else
{
  b--;
x = 0;
```

Mi lenne, ha itt kódkiemelést alkalmaznánk?

Fordítóprogramok előadás (A,C,T szakirány) Kódoptimalizá

Ciklusok kifejtése

Eredeti kód

```
for( int i=0; i<4; ++i )
 a += t[i];
```

Optimalizált kód

```
a += t[0];
a += t[1];
a += t[2];
a += t[3];
```

Mérlegelni kell, hogy a méret és a sebesség mennyire fontos...

Parciális kifejtés

Eredeti kód

```
for( int i=0; i<4; ++i )
{
  a += t[i];
```

Optimalizált kód

```
for( int i=0; i<4; i+=2
{
 a += t[i];
 a += t[i+1];
```

Ciklusok összevonása

Eredeti kód

```
for( int i=0; i<4; ++i )
  t1[i] = 0;
for( int i=0; i<4; ++i )
  t2[i] = 1;
```

Optimalizált kód

```
for( int i=0; i<4; i++ )
 t1[i] = 0;
 t2[i] = 1;
```

Fordítóprogramok előadás (A,C,T szakirány) Kódoptimalizálás

Frekvenciaredukálás

- költséges utasítások "átköltöztetése" ritkábban végrehajtódó alapblokkba
- példa:
 - ciklusinvariánsnak nevezzük azokat a kifejezéseket, amelyeknek a ciklus minden lefutásakor azonos az értékük
 - a ciklusinvariánsok (esetenként) kiemelhetők a ciklusból

Eredeti kód

```
cin >> a;
cin >> h;
while(h > 0)
  cout << h*h*sin(a);</pre>
  cin >> h;
```

Optimalizált kód cin >> a; cin >> h; double s = sin(a); while(h > 0) cout << h*h*s;</pre> cin >> h;

Erős redukció

• a ciklusban lévő költséges művelet (legtöbbször szorzás) kiváltása kevésbé költségessel

Eredeti kód

```
for( int i=a; i<b; i+=c</pre>
{
  cout << 3*i;
```

Optimalizált kód

```
int t1 = 3*a;
int t2 = 3*c;
for( int i=a; i<b; i+=c )
  cout << t1;
  t1 += t2;
```

Fordítóprogramok előadás (A,C,T szakirány) Kódoptimalizálás