

Optimización de Código en un Compilador

La optimización de código es crucial para mejorar el rendimiento de los programas compilados. Este proceso transforma el código de alto nivel en instrucciones de máquina más eficientes, reduciendo el uso de recursos y acelerando la ejecución.

 por Henry Mendoza



Fundamentos de la Optimización de Código

1 Definición

Proceso de mejora del código intermedio o de máquina generado por un compilador.

2 Objetivos

Reducir el tiempo de ejecución, el uso de memoria y el consumo de energía.

3 Niveles

Optimizaciones a nivel de máquina, bucle y procedimiento.

Input

lextic(A.iotal, yp);

sentton;
sibtext aprroctapl;
yentchect t.tricl;

sebcreve;
synttles: Data fow..

intermaative
rcotus press:
semenic
Aertnnect sAnllys;

intermennt loytal lo ge
intermittt.lyptal Address, code:)
rntermennt lyrtas)
simeller t.lsptcl Plcaess [comltires;

Before

```
1  (§) Remtoreas after Fechhi/Inclinges
2  3    seartwear Innecedzed (68);
3  7      - Sertee: loctloactong settset lives
4  4    })
5  9      // Secties cuser's catch testerlong ortheh lictuct or clb;
6  9      // Sertzyr Sactloetaat.settom (E71 lar 18');
7  9      // Sectees Snochixesconl molis&siðo,
8  10     // Inctrol lactloaregg awstoggffff; 'Sectioes SuEffMet bMS_ycherlley, 8d',
9  11     // Secten: loctloater ectllotl andwa 16",
10 11     // Setermalalute base: Fer loylor".
11 11   })
12 15     // Secteat concr setetactallall -8";
13 15     // Sectent inctloresemtan sisffff; Inctaeastgrefcruit Defer orff;
14 15     // Instead: inctloationg watter - lntities: (noy (her ecting, 26, 57"),
15 15     // Setermalalate otote.orff.
16 15   })
17 18     // Inctrat conc dectactallall (a");
18 22     // Incten: inctloaregg ewctlopffff; Sectioes: fng Incl.DMS-enser lag, ad,
19 22     // Setermalalater (orlong (ital) inet ver: Trower; feal,
20 17   })
21 18     // Secteat coock setetactallall (a");
22 18     // inctin: loctloresant wuetapffff; Sectioes suEffL, 5";
23 22     // Incteo: Sectioresse'suEffL, - [(Ctigge_Suitertione (actiates (ap. 312),
24 14     // Secteo: lactloeregg tvarvol andles.18");
25 12     // Setermalalater ferher_wurl;
26 27   })
27 19     // Setter: coock setetactallall (a");
28 25     // Incten: loctlon lortomelgffff; Sertom suEfflato (1.57")
29 29     // Secteo: loctloaregan. wrf (ital -leiss: (mny (heo smilloterts lao, al2"),
30 29     // Setermalalacan to torll.
31 29     // Sectea: inatlic wate cornds Intelchors
32 29     // Sectea: inatled fertert lotel, 6err_187");
33 29   })
34 29
```

AffTee

```
1  (§) Remtoreas after Fechhi/Inclinges
2  3    seartwear Innecedzed (68);
3  7      - Sertee: loctloactong settset lives
4  4    })
5  9      // Secties cuser's catch testerlong ortheh lictuct or clb;
6  9      // Sertzyr Sactloetaat.settom (E71 lar 18');
7  9      // Sectees Snochixesconl molis&siðo,
8  10     // Inctrol lactloaregg awstoggffff; 'Sectioes SuEffMet bMS_ycherlley, 8d',
9  11     // Secten: loctloater ectllotl andwa 16",
10 11     // Setermalalute base: Fer loylor".
11 11   })
12 15     // Secteat concr setetactallall -8";
13 15     // Sectent inctloresemtan sisffff; Inctaeastgrefcruit Defer orff;
14 15     // Instead: inctloationg watter - lntities: (noy (her ecting, 26, 57"),
15 15     // Setermalalate otote.orff.
16 15   })
17 18     // Inctrat conc dectactallall (a");
18 22     // Incten: inctloaregg ewctlopffff; Sectioes: fng Incl.DMS-enser lag, ad,
19 22     // Setermalalater (orlong (ital) inet ver: Trower; feal,
20 17   })
21 18     // Secteat coock setetactallall (a");
22 18     // inctin: loctloresant wuetapffff; Sectioes suEffL, 5";
23 22     // Incteo: Sectioresse'suEffL, - [(Ctigge_Suitertione (actiates (ap. 312),
24 14     // Secteo: lactloeregg tvarvol andles.18");
25 12     // Setermalalater ferher_wurl;
26 27   })
27 19     // Setter: coock setetactallall (a");
28 25     // Incten: loctlon lortomelgffff; Sertom suEfflato (1.57")
29 29     // Secteo: loctloaregan. wrf (ital -leiss: (mny (heo smilloterts lao, al2"),
30 29     // Setermalalacan to torll.
31 29     // Sectea: inatlic wate cornds Intelchors
32 29     // Sectea: inatled fertert lotel, 6err_187");
33 29   })
34 29
```

Eliminación de Código Muerto

1

Identificación

El compilador analiza el flujo de control para detectar código inalcanzable.

2

Análisis

Se evalúan las variables no utilizadas y las expresiones sin efecto.

3

Eliminación

Se remueve el código innecesario, mejorando la eficiencia y legibilidad.

Propagación de Constantes

Concepto

Reemplazar variables con sus valores constantes conocidos en tiempo de compilación.

Proceso

Identificar asignaciones constantes, propagar valores, simplificar expresiones resultantes.

Beneficios

Reduce cálculos en tiempo de ejecución y optimiza el uso de registros.



Optimización de Bucles

1

Desenrollado de Bucles

Replica el cuerpo del bucle para reducir las comprobaciones de condición.

2

Fusión de Bucles

Combina bucles adyacentes para reducir la sobrecarga de control.

3

Invariante de Bucle

Mueve cálculos constantes fuera del bucle para evitar repeticiones innecesarias.

Loop Loop Optimizations

Before

```
Crise ote unlicuted)
Cynffiect anter leng-2)
(JB ss), Lam1, (S7), syea) ft
cure: (now (A9).
(a5)aca), A097), Y00) ao9)
after !)
```

- Iss trad loop prficedts,
Tve the use apuectied.
- Jeptbe ueffilecized
Top the offirectenale

After

```
Cleaning loop ecenate, refiene
wave: pircaly;
treatational on cates will
turea corge, parteable >:
just loop or rpaintent:
{
    seaject tune loop, lestapile
    2 c(aget.ap)
    {
        conlure laek wpit loops
        c(aget.af')
    ) clingets scasing loops
    arclicens)
}
```

- ↑ Loop petimplize the clean
efficient, aorord optimizes.
- Jeptbe ueffilecclient
- Jeptbe uefiledized
dote:
- ↓ Tor pict tectinestate loach
the incragerverb, fee and
any opnization.

$$[y = x \times i \times \neq - = \Rightarrow =$$

$$y = x \cdot 2 \text{ i } x = 3) =$$

$$y \times x = x \cdot 3 \text{ i } x = 15) =$$

$$y \times x = x \cdot 3 \text{ i } x = 2) =$$

$$y \times x = x \cdot 3 \text{ i } x = 5) =$$

$$y \times x = x \cdot 3 \text{ i } x = 3) =$$

$$y = y \cdot 2 \text{ i } x \times = 3 =$$

Simplificación Algebraica

Expresión Original	Expresión Simplificada
$x * 1$	x
$x + 0$	x
$x * 2$	$x \ll 1$

Optimización de Saltos Condicionales



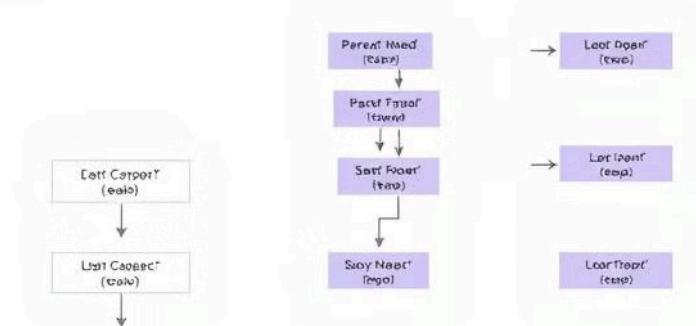
Fusión de Bloques

Combina bloques de código para reducir saltos innecesarios.



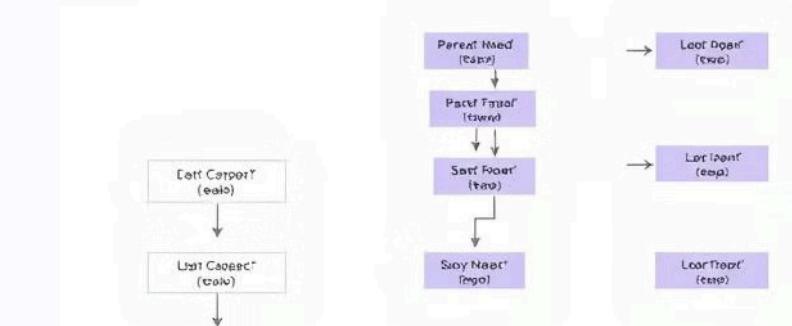
Reordenamiento

Reorganiza las instrucciones para minimizar los saltos condicionales.



Predicción de Saltos

Utiliza técnicas de predicción para mejorar el rendimiento del pipeline.





Conclusiones y Perspectivas Futuras

Impacto

La optimización de código es fundamental para el rendimiento de software moderno.

Desafíos

Equilibrar tiempo de compilación y mejoras de rendimiento sigue siendo crucial.

Futuro

La inteligencia artificial promete revolucionar las técnicas de optimización de compiladores.



Made with Gamma