Transformações para Otimização de Código

Introdução

- Usar as informações coletadas pelas análises de fluxo de dados
- Tornar o código mais eficiente
- Inicialmente serão vistas:
  - Constant Propagation
  - Dead Code Elimination
  - Copy Propagation
  - CSE

Constant Propagation

Dadas as instruções:

i_1: `t = c` , onde `c` é constante
i_2: `y = t + x`

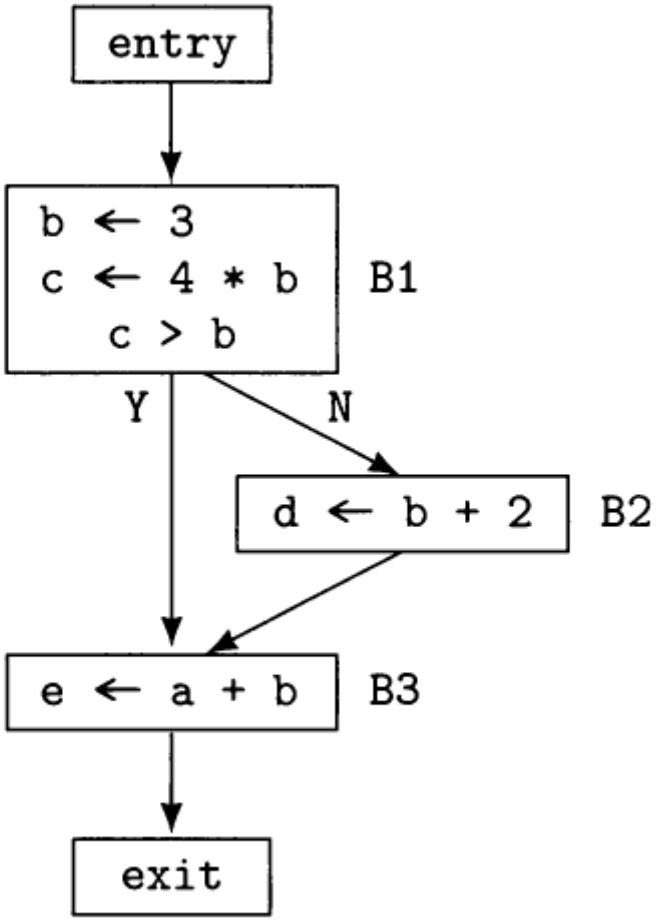
Pode-se afirmar que `t` é constante em i_2 se:

- i_1 alcança i_2 tal que **todo caminho** do início até i_2 contém i_1
- nenhuma outra definição de `t` alcança i_2

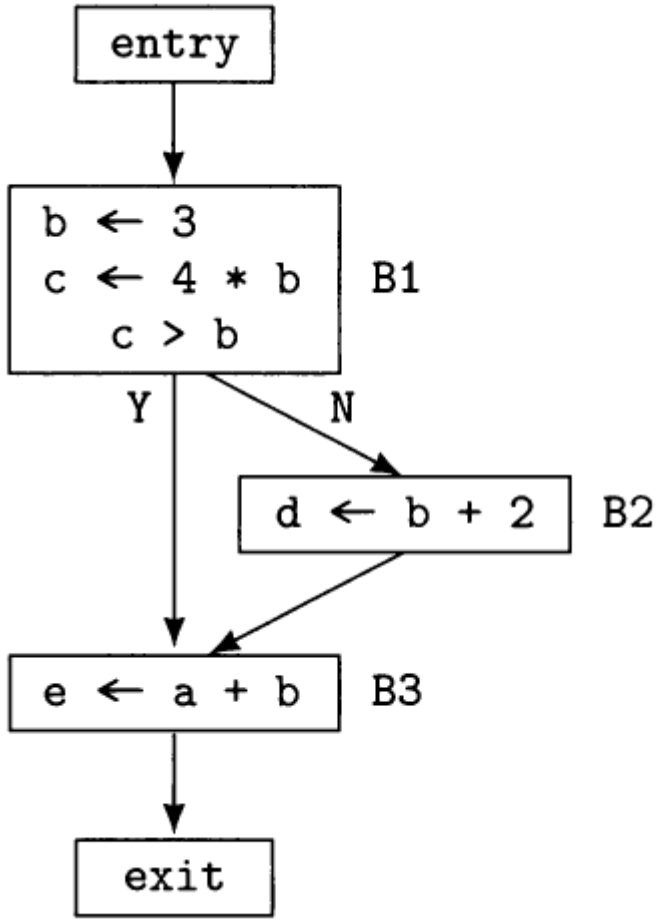
Pode-se reescrever:

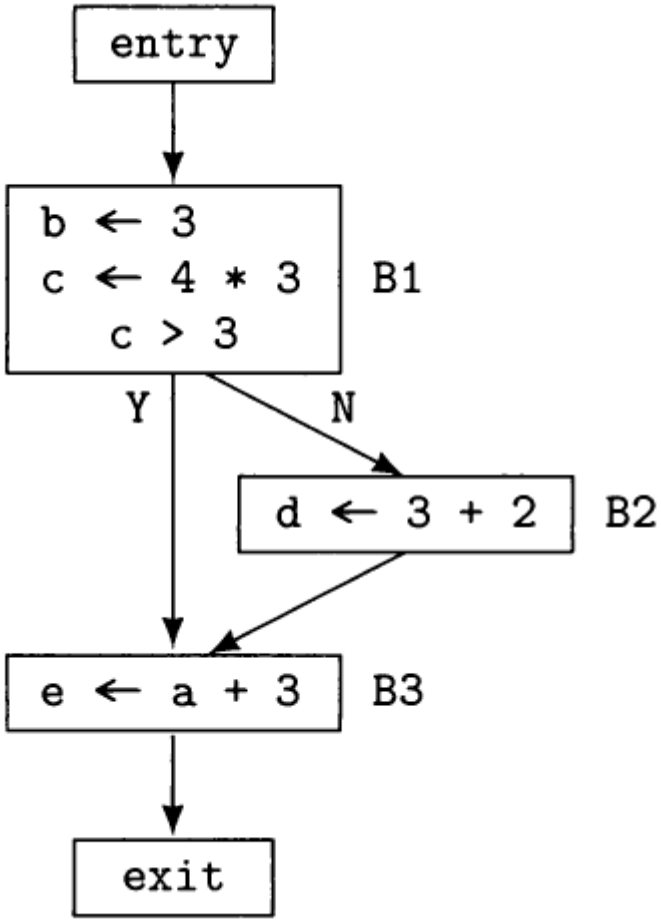
i_2: `y = c + x`

Constant Propagation



Constant Propagation





Dead Code Elimination

Se existem instruções do tipo:

```
i:  t  =  x + y
i:  t  =  M[x]
```

De maneira que **t** não está vivo após **i**,
então **i** pode ser eliminada.

Instruções com efeito colateral:

- Podem provocar alteração no resultado do programa se forem removidas
- O código pode funcionar com o otimizador desligado e não funcionar com ele ligado.

Dead Code Elimination

$$
\begin{aligned}
x &\leftarrow b + c \\
y &\leftarrow a + x \\
u &\leftarrow b + c \\
v &\leftarrow a + u \\
& f(v)
\end{aligned}
$$