



# Compiladores

Pequenos casos práticos



# Compiladores

- Transformar código texto em um binário executável.



# Compiladores

- Transformar código texto em um binário executável.
- Otimizar o executável.



# Otimizações

Alterar as instruções do programa final sem alterar a sua funcionalidade.



# Otimizações (Exemplos)

- Remoção de cálculos desnecessários
  - $X = 5; X + 10;$
  - $X = 15;$



# Otimizações (Exemplos)

- Remoção de cálculos desnecessários
  - `X = 5; X + 10;`
  - `X = 15;`
- Desenrolar laços
  - `for (i = 0; i < 4; i++) X[i] = i;`
  - `X[0] = 0; X[1] = 0; X[2] = 2; X[3] = 3;`



# Otimizações (Exemplos)

- Remoção de cálculos desnecessários
  - `X = 5; X + 10;`
  - `X = 15;`
- Desenrolar laços
  - `for (i = 0; i < 4; i++) X[i] = i;`
  - `X[0] = 0; X[1] = 0; X[2] = 2; X[3] = 3;`
- Expansão de funções
  - `int f1(int x) { return x*2; }`  
`int f2(int x) { return f1(x) + 10; }`
  - `int f2(int x) { return (x * 2) + 10; }`



# Otimizações (Exemplos)

- Remoção de cálculos desnecessários
  - `X = 5; X + 10;`
  - `X = 15;`
- Desenrolar laços
  - `for (i = 0; i < 4; i++) X[i] = i;`
  - `X[0] = 0; X[1] = 0; X[2] = 2; X[3] = 3;`
- Expansão de funções
  - `int f1(int x) { return x*2; }`
  - `int f2(int x) { return f1(x) + 10; }`
  - `int f2(int x) { return (x * 2) + 10; }`
- Remoção de variáveis desnecessárias
  - `X = Y + 5; Z = X * 2; X = Z - 2;`
  - `X = Y + 5; X = X * 2; X = X - 2;`





# Otimizações

Otimizações podem melhorar

- Tempo de execução
- Quantidade de memória usada para operações
- Tamanho ocupado pela parte executável do programa



# Otimizações

Otimizações podem melhorar

- Tempo de execução
- Quantidade de memória usada para operações
- Tamanho ocupado pela parte executável do programa
- Legibilidade de código



# Otimizações

Otimizações podem melhorar

- Tempo de execução
- Quantidade de memória usada para operações
- Tamanho ocupado pela parte executável do programa
- Legibilidade de código
  - Exemplo: Declarar várias variáveis para guardar valores intermediários pode ajudar a entender a operação feita. Sem otimização isso geraria um custo para o programa final, com otimização não faz diferença nenhuma.



# Lab01

- Nesse laboratório vocês vão receber 4 arquivos com o nome exemploX.c
- Cada um foi feito para exemplificar uma das 4 otimizações explicadas nos slides anteriores.
- Vocês devem: Compilar o programa com e sem otimização e analisar o executável gerado para identificar como e qual otimização foi feita.



# Lab01

- Compilar sem otimização:
  - `gcc -O0 exemploX.c -o exemploX_o0`
- Compilar com otimização:
  - `gcc -O3 exemploX.c -o exemploX_o3`
- Analisar o binário:
  - `Objdump -D exemploX_oY > exemploX_oY.dump`
  - `exemploX_oY.dump` vai conter uma versão do executável legível com as instruções ASM.



# Lab01: Analisar o binário

iCounter.py – Contem funções para extrair informações sobre as instruções usadas no binário.

- `(c1,c2) = iCounter.exec("exemploX_o0.dump", "exemploX_o3.dump")`

c1 e c2 vão ser vetores com tuplas contendo o nome de uma instrução e quantas vezes ela aparece no programa. O c1 contém instruções de exemploX\_o0.dump e c2 de exemploX\_o3.dump.

Ambos os vetores c1 e c2 contém as mesmas instruções, e estão ordenados por ordem alfabética pelo nome da instrução.



# Lab01: Entrega

O trabalho deve ser feito em dupla.

Deve ser entregue um PDF contendo o nome e RA de ambos os alunos da turma e as seguintes informações para cada arquivo exemploX.c

- Qual das 4 otimizações se destacou no arquivo.
- Um histograma com o uso de instruções da versão compilada sem e com otimização. A ordem das instruções no eixo X deve ser a alfabética, e ambos os gráficos devem mostrar o histograma para as mesmas instruções (isso pode ser facilmente obtido pela função iCounter.exec do arquivo em python).

O nome do PDF deve ser [RAaluno1]\_[RAaluno2].pdf