7º Exercício Prático
Desenvolvido no Laboratório

Objetivo

Utilizar a otimização automática do compilador GCC, comparando alguns dos códigos otimizados da aula passada com os mesmos códigos, mas também otimizados com a opção de otimização **-O2** do GCC.

Materiais

1. Compilador GCC
2. Arquivo Acessos.cpp
3. Arquivo com textos: biblia-em-txt.txt.

Desenvolvimento

O experimento de hoje, como nos do 4º e 5º Laboratório, usa um programa que faz a contagem de ocorrências de uma letra em um arquivo de texto. No desenvolvimento deste experimento serão analisadas as vantagens no tempo quando se utiliza a otimização disponível no compilador GCC.

Executando o programa sem otimização

1. Compile o programa Acessos.cpp
    - gcc -g Acessos.cpp -o Acessos -masm=intel
2. Execute o programa.
    - ./Acessos
3. Observe a saída do programa Acessos e responda:

   Quantas vezes a letra ‘a’ aparece no texto?

   Ocorrências da letra ‘a’: _____

   Qual foi o tempo observado na execução?

   Tempo 1 (sem -O2): _____ s

Otimização com a opção do GCC “-O2”

1. Compile o programa Acessos.cpp
    - gcc -g **-O2** Acessos.cpp -o Acessos -masm=intel
    - O “-O2” define que o compilador deve otimizar com nível 2. Outros níveis serão abordados em atividades futuras.
2. Execute o programa.
    - ./Acessos
3. Observe a saída do programa Acessos e responda:

Quantas vezes a letra ‘a’ aparece no texto?

Ocorrências da letra ‘a’: _____

Qual foi o tempo observado na execução?

Tempo 2 (com -O2): _____ s

Revisando as otimizações

Nesta otimização a função *fgetc* que é chamada muitas vezes deve ser substituída pela função *fgets.*

1. Abra o arquivo Acessos.cpp em um editor de texto
2. Altere a função contaLetra() com o código abaixo

```
int contaLetra(FILE *pArq, char c) {
  int cont = 0;
  char linha[MAX_CARACTERES_POR_LINHA];
  while (fgets(linha, MAX_CARACTERES_POR_LINHA, pArq)) {
    for (int i = 0; i < strlen(linha); i++) {
      if (linha[i] == c) {
        cont++;
      }
    }
  }
  return cont;
}
```

Quantas vezes a letra ‘a’ aparece no texto?

Ocorrências da letra ‘a’: _____

Qual foi o tempo observado na execução?

Tempo 3 (sem -O2): _____ s

Tempo 4 (com -O2): _____ s

Segunda otimização

A função *strlen* é chamada em cada interação e a cada chamada ele busca na cadeia de caracteres, para reduzir o tempo observado no programa anterior, pode-se chamar a função *strlen* uma única vez logo após a leitura da linha.

1. Altere o programa da primeira otimização chamando o *strlen* uma única vez para cada linha. Quantas vezes a letra ‘a’ aparece no texto?

Ocorrências da letra ‘a’: _____

Qual foi o tempo observado na execução?

Tempo 5 (sem -O2): _____ s

Tempo 6 (com -O2): _____ s

Terceira otimização

Vamos tentar agora fazer uma única leitura do arquivo, colocar todos os dados na memória e então fazer a contagem da letra.

1. Altere a função contaLetra() do programa anterior com o código abaixo

```
int contaLetra(FILE *pArq, char c) {
  int cont = 0;
  int tam = fread(tudo, MAX_CARACTERES_NA_BIBLIA, 1, pArq);
  for (int i = 0; i < MAX_CARACTERES_NA_BIBLIA; i++) {
    if (tudo[i] == c) {
      cont++;
    }
  }
  return cont;
}
```

Quantas vezes a letra ‘a’ aparece no texto?

Ocorrências da letra ‘a’: _____

Qual foi o tempo observado na execução?

Tempo 7 (sem -O2): _____ s

Tempo 8 (com -O2): _____ s

Última otimização

Muitas vezes é possível utilizar linguagem de montagem para otimizar uma seção de um programa. Altere a busca na função contaLetra para que esta busca seja feita em linguagem de montagem. **ATENÇÃO: o código abaixo foi melhorado em relação ao código em linguagem de montagem apresentado no Laboratório 5, assim, use a apresentado neste documento.**

1. Altere a função contaLetra() do programa anterior com o código abaixo.

```
int contaLetra(FILE *pArq, char c) {
  int cont = 0;
  int tam = fread(tudo, MAX_CARACTERES_NA_BIBLIA, 1, pArq);
  asm("   mov   rdi, %[tudo]    \n"
      "   mov   rcx, %[MAX_CARACTERES_NA_BIBLIA]\n"
      "   mov   al, %[c]        \n"
      "   xor   rdx, rdx        \n"
      "   xor   rbx, rbx        \n"
      "repete:                  \n"
      "   cmp   [rdi+rcx], al   \n"
      "   sete  dl              \n"
      "   add   rbx, rdx        \n"
      "   sub   rcx,1           \n"
      "   jnz   repete          \n"
      "   nop                   \n"
      "   mov   %[cont], ebx    \n"
      : [cont] "=r"(cont) // resultado em cont
      : [tudo] "r"(tudo), // vetor com texto
        [c] "r"(c),       // caractere a procurar
        [MAX_CARACTERES_NA_BIBLIA] "i"(MAX_CARACTERES_NA_BIBLIA)
      : "rax", "rbx", "rcx", "rdx", "rdi");
  return cont;
}
```

Quantas vezes a letra ‘a’ aparece no texto?

Ocorrências da letra ‘a’: _____

Qual foi o tempo observado na execução?

Tempo 9 (sem -O2): _____ s

Tempo 10 (com -O2): _____ s

Avaliando os resultados

Envie a avaliação dos resultados como descrito no arquivo “Avaliacao Dos Resultados.pdf”, mas inclua uma comparação entre os pares de tempos com e sem a opção “-O2”.

Conclusão

Os compiladores podem fazer algumas otimizações no código, mas muitas outras otimizações dependem da atenção dos programadores.