Renato Toshiaki Shibata RA:082674

MC723 – Lucas Wanner

Exercício 1

Relatório

Compilando primo.c sem nenhuma opção de compilação extra:

```
gcc -Wall -pg primo.c -o primo
```

Logo em seguindo executando-o:

```
./primo
```

E então utilizando o gprof:

```
gprof primo gmon.out > analysis.txt
```

Temos como saída no analysis.txt:

```
Each sample counts as 0.01 seconds.
 %   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
101.23      0.32     0.32        1   323.95   323.95  primo
```

Ou seja, 323.95 milisegundos para sua execução

Compilando com a opção O0 e usando os mesmos comandos na seção anterior:

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
101.23      0.31     0.31        1   313.83   313.83  primo
```

Compilando com a opção O1:

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
101.23      0.33     0.33        1   334.07   334.07  primo
```

Compilando com a opção O2:

```
Each sample counts as 0.01 seconds.
no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
```

Compilando com a opção O3:

```
Each sample counts as 0.01 seconds.
 no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
```

Ao utilizar o parâmetro -mtune na compilação, é possível utilizar vários valores para meu processador, i5-5220U, como:

-mtune=generic

Produz código otimizado para processadores IA32/AMD64/EM64T.
Recomendável usar apenas quando não se sabe a CPU no qual o código irá rodar.
Assim que novos processadores são empregados no mercado, o comportamento desta opção irá mudar. Portanto se atualizar para uma nova versão de GCC, o código gerado irá mudar para refletir os processadores que são mais comuns quando a versão do GCC foi lançada.

Após compilar usando:

```
gcc -Wall -pg -mtune=generic primo.c -o primo
```

E seguindo os mesmo passos na seção anterior foi obtido em analysis.txt:

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
101.23      0.32     0.32        1   323.95   323.95  primo
```

-mtune=native

Este determina o tipo de processador da máquina em que se está compilando. Desse modo, irá produzir código otimizado para a máquina local sob restrições do conjunto de instruções selecionado.

Após compilar usando:

```
gcc -Wall -pg -mtune=native primo.c -o primo
```

E seguindo os mesmo passos na seção anterior foi obtido em analysis.txt:

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
101.23      0.33     0.33        1   334.07   334.07  primo
```

-mtune=i686

O mesmo que *generic*, mas quando usado com -march=i686, o conjunto de instruções de PentiumPro será usado então o código irá ser executado em toda família de chips i686.

Após compilar usando:

```
gcc -Wall -pg -mtune=i686 -march=i686 primo.c -o primo
```

E seguindo os mesmo passos na seção anterior foi obtido em analysis.txt:

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
101.23      0.31     0.31        1   313.83   313.83  primo
```

Com esses resultados podemos ver que o tempo não melhorou substancialmente, permaneceram os mesmos.

Então após separar em 3 arquivos: main.c , calc_primo.c e primo.h, cujos códigos são respectivamente:

```
//main.c
#include <primo.h>

int main(){

  int n = 104395301;

  if (primo(n))
    printf("%d é primo.\n", n);
  else
    printf("%d não é primo.\n", n);
}
-----------------------------------------------------
//calc_primo.c
#include <stdio.h>
#include <primo.h>

int primo(int n)
{
  int i;

  for(i = 2; i < n; i ++)
    if (n % i == 0)
      return 0;

  return 1;
}
-----------------------------------------------------
//primo.h
int primo(int n);
-----------------------------------------------------
```

Então criei um arquivo chamado makefile que tem como conteúdo:

```
main: main.c calc_primo.c
        gcc -o main main.c calc_primo.c -O0 -I.
```

Usei o parametro -O0 porque esse foi o menor valor obtido de tempo dentre as outra opções de otimização.

Assim ao se executar o seguinte comando:

```
make -f makefile
```

E então usar o gprof para main, o resultado foi:

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
101.23      0.31     0.31                             _fini
```

Isso corresponde ao mesmo valor em segundos, 0.31, do valor quando otimizado com -O0 e sem usar o makefile. A única diferença é que aqui não mostra os valores em milisegundos.