Roteiro – Otimização

Atividade 1

Considere o código C abaixo:

```
#include <stdio.h>

double powern (double d, unsigned n)
{
  double x = 1.0;
  unsigned j;

  for (j = 1; j <= n; j++)
    x *= d;

  return x;
}

int main (void)
{
  double sum = 0.0;
  unsigned i;

  for (i = 1; i <= 100000000; i++)
    {
      sum += powern (i, i % 5);
    }

  printf ("sum = %g\n", sum);
  return 0;
}
```

Compile usando:
$ gcc -Wall -fopt-info -O1 test.c -lm
$ gcc -Wall -fopt-info -O2 test.c -lm
$ gcc -Wall -fopt-info -O3 test.c -lm
$ gcc -Wall -fopt-info -O3 -funroll-loops test.c -lm

Faça uma tabela com a diferença nos tempos.
Vale a pena o uso das diretivas de otimização?
Que otimizações foram realizadas pelo compilador (flag: -fopt-info)?

Atividade 2

Compare o tempo de execução para os códigos a seguir:

```
for(int i = 0; i < N; i++)
{
    z += x/y;
}
```

```
double denom = 1/y;
for(int i = 0; i < N; i++)
{
   z += x*denom;
}
```

Para diferentes tamanhos de N, monte uma tabela com os tempos de execução. Qual é o melhor código e por que?

Atividade 3

Compare o tempo de execução para os códigos a seguir (colocar dentro de um laço):

```
//compilar normalmente, sem e com parâmetros de otimização
int foo(a, b)
{
      a = a - b;
      b++;
      a = a * b;
      return a;
}
```

//compilado com -finline-functions

```
#define foo(a, b) (((a)-(b)) * ((b)+1))
```

Para diferentes tamanhos de N, monte uma tabela com os tempos de execução. Qual é o melhor código e por que?

Atividade 4

1) Entrar em https://godbolt.org/

2) Escolher o compilador x-84-64 clang 16.0

3) Clicar no '+ Add New' e adicionar 'Optimization'

3) Testar os seguintes códigos com diferentes flags de compilação:

```
int loop()
{
    int a[100];
    int sum=0;

    for (int i = 0; i < 100; i++)
    {
        a[i] = i*2 + 10;
    }

    for (int i = 0; i < 100; i++)
    sum+=a[i];

    return sum;
}
```

O que é possível verificar com a flag -O3?