

**UNIVERSIDADE FEDERAL DA BAHIA**

**DEPARTAMENTO DE CIÊNCIA DE COMPUTAÇÃO**

**LUCAS MENEZES PEREIRA**

**MATHEUS OLIVER DE CARVALHO CERQUEIRA**

**RELATÓRIO DE COMPARAÇÃO DE PROCEDIMENTOS C/ASSEMBLY/JAVA PARA OPERAÇÕES MATEMÁTICAS COM NOTAÇÃO DE PONTO FLUTUANTE**

Salvador

2017

1. **APRESENTAÇÃO**

C é uma das linguagens de programação mais populares, sendo predominante no ensino básico de programação devido à semelhança semântica e sintática com a língua inglesa, o que torna seu entendimento acessível a grande parte da população (linguagem de alto nível). Assim, a maioria das arquiteturas prevê um compilador C.

Assembly, no entanto é uma notação voltada ao comando direto à máquina, que é utilizado por arquiteturas específicas de computador, para programar códigos entendidos por dispositivos computacionais, como microprocessadores e microcontroladores. Nesse sentido, linguagens de alto nível surgem como uma camada, em que o programador utiliza procedimentos, que são traduzidos pelo compilador em linguagem de máquina, que é interpretada pelo dispositivo.

Portanto, como a linguagem C tem, internamente em suas funções, comandos em Assembly, os programas podem ser construídos mesclando as linguagens, com intenção de otimizar o código em trechos específicos. A otimização é muito útil para processamento de quantidades extraordinárias de dados, ou quando a precisão das medidas de entradas e saída deve ser máxima.

A representação de números em ponto flutuante surgiu como alternativa para que métodos computacionais pudessem abrangir essa precisão e extensão que diversas aplicações como Astronomia e Astronáutica, Meteorologia e Computação gráfica requerem. Essa notação será utilizada para expressar resultados das operações.

A linguagem JAVA, baseada em C, é uma das linguagens de programação mais utilizadas, possuindo diversas bibliotecas e interface que facilitam o trabalho dos programadores. Seu ponto forte é a facilidade de execução de um mesmo programa em diversos sistemas operacionais.

Essas três linguagens serão analisadas neste relatório do ponto de vista de tempo de execução.

1. **MATERIAIS UTILIZADOS**
   1. **Sistema Operacional:**
   * Windows 10 64 bits, com processador em x64.
   1. **Hardware:**
   * 4GB de memória RAM (Utilizável: 3,89GB);
   * Intel Core i3-3010M CPU @2.40GHz.
   1. **Software:**

* Netbeans.

1. **OBJETIVOS**
   1. **Geral**

Comparar as execuções de operações matemáticas entre números em notação de Ponto Flutuante para cada linguagem (C, Assembly e JAVA) e perceber como o código pode ser modificado, a partir da integração de linguagens, para deixar a execução mais rápida.

* 1. **Específicos**

Considerando as funções abaixo, para um número x real:

* Módulo;
* Exponencial Natural;
* Logaritmo Natural;
* Raiz Quadrada;
* Cosseno;
* Seno;
* Tangente;
* Arco Cosseno;
* Arco Seno;
* Arco Tangente.

Deve-se:

* Desenvolver programa em C para o cálculo de cada função matemática;
* Desenvolver programa em JAVA para o cálculo de cada função matemática;
* Desenvolver programa em C/ASSEMBLY para o cálculo de cada função. Neste caso, procedures em ASSEMBLY que realizam o cálculo das funções serão desenvolvidas e chamadas no programa em C;
* Comparar o tempo de execução dos programas desenvolvidos na linguagem C e na linguagem JAVA e indicando o ganho de desempenho, expressando os motivos para tal. Considera-se o ganho como a relação entre o tempo de execução em C e o tempo de execução em C/ASSEMBLY.

1. **METODOLOGIA**

Cada objetivo será cumprido individual e sequencialmente. Como padrão para os programas, a seguinte estrutura será aplicada:

* Inclusão de bibliotecas;
* Declaração de variáveis globais, sendo um contador inteiro para iteração, que compreenderá o intervalo de [-100000;100000[ (Duzentas mil iterações) e servirá como entrada para as funções; e uma variável do tipo “double” (dupla precisão, ou precisão estendida), para receber os resultados das funções para cada iteração. Sabe-se que esses valores não funcionarão integralmente para todas as funções, mas, sendo a comparação entre linguagens, o número de iterações bem-sucedidas é o mesmo.
* Execução do programa contendo a iteração e a contagem dos tempos. O princípio da cronometragem será utilizar funções nativas para pegar o instante de tempo imediatamente antes da primeira iteração, o imediatamente após a última, realizar a subtração desses instantes e exibir o intervalo de tempo em milissegundos. Tal precisão facilitará o processo de comparar os tempos, sendo suficiente para as duzentas mil contagens a cada operação matemática.

A execução da mesma estrutura para os programas de linguagens diferentes é a terceira garantia de um julgamento justo, sendo a primeira o fato de serem rodados na mesma máquina e a segunda é que a execução dos programas será feita na mesma IDE (Netbeans). Pode-se adicionar um quarto critério, que foi garantir que nenhum outro programa, além da IDE estivesse em primeiro ou segundo plano de execução na máquina, havendo a tendência de utilização quase semelhante da CPU, de disco e de memória para todas as linguagens.

Dada a estrutura de programação, foi estabelecido um total de cinco execuções de cada operação, de cada linguagem, com objetivo de estimar melhor o tempo de execução médio (o que não aconteceria se os valores de tempo de cada função fossem constantes).

1. **OPERACIONAL**
   1. **Linguagem C**

A inclusão das bibliotecas foi dada por:

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <conio.h>*

*#include <math.h>*

*#include <sys\timeb.h>*

A fim de tornar a execução mais rápida, cada função foi transformada eu um programa (seria o mesmo se as partes não utilizadas em cada análise de função fosse comentada). Assim, como cada função corresponde a um programa, as operações foram construídas internamente ao programa, de forma que a única diferença entre as funções está nas duas linhas internas ao for(), e, consequentemente, no nome das variáveis que guardam o valor de retorno das funções. Essa diferença será explicitada nos códigos a seguir, com as palavras em **negrito**.

Função Módulo de X:

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double* ***MODULO****;­*

*for(i=-100000;i<100000;i++){*

***MODULO*** *=* ***abs****(i);*

*printf("\n%lf",* ***MODULO****);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Exponencial Natural de X:

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double* ***EXPONENCIAL****;­*

*for(i=-100000;i<100000;i++){*

***EXPONENCIAL*** *=* ***exp****(i);*

*printf("\n%lf",* ***EXPONENCIAL****);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Logaritmo Natural de X:

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double* ***LOGARITMO****;­*

*for(i=-100000;i<100000;i++){*

***LOGARITMO*** *=* ***log****(i);*

*printf("\n%lf",* ***LOGARITMO****);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Raiz de X:

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double* ***RAIZ****;­*

*for(i=-100000;i<100000;i++){*

***RAIZ*** *=* ***sqrt****(i);*

*printf("\n%lf",* ***RAIZ****);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Cosseno de X:

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double* ***COSSENO****;­*

*for(i=-100000;i<100000;i++){*

***COSSENO*** *=* ***cos****(i);*

*printf("\n%lf",* ***COSSENO****);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Seno de X:

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double* ***SENO****;­*

*for(i=-100000;i<100000;i++){*

***SENO*** *=* ***sen****(i);*

*printf("\n%lf",* ***SENO****);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Tangente de X:

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double* ***TANGENTE****;­*

*for(i=-100000;i<100000;i++){*

***TANGENTE*** *=* ***tan****(i);*

*printf("\n%lf",* ***TANGENTE****);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Arco Cosseno de X:

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double* ***ARCOCOSSENO****;­*

*for(i=-100000;i<100000;i++){*

***ARCOCOSSENO*** *=* ***acos****(i);*

*printf("\n%lf",* ***ARCOCOSSENO****);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Arco Seno de X:

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double* ***ARCOSENO****;­*

*for(i=-100000;i<100000;i++){*

***ARCOSENO*** *=* ***asen****(i);*

*printf("\n%lf",* ***ARCOSENO****);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Arco Tangente de X:

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double* ***ARCOTANGENTE****;­*

*for(i=-100000;i<100000;i++){*

***ARCOTANGENTE*** *=* ***atan****(i);*

*printf("\n%lf",* ***ARCOTANGENTE****);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Para esses códigos, foram repetidas 5 execuções para cada, e os resultados serão apresentados posteriormente, na seção RESULTADOS E DISCUSSÕES

* 1. **Linguagem JAVA**

No caso da linguagem JAVA, não foram incluídas bibliotecas no início do programa, mas as chamadas das funções ocorreram dentro das iterações (como em *abs = Math.abs(i)*;), devido à própria estrutura de JAVA. Excetuando isso, os programas seguem a mesma estrutura dos criados em C, desde a ordem de declaração de variáveis, até a contagem de tempo. O destaque em negrito, utilizado na seção de C, foi mantido aqui, para, novamente, evidenciar as semelhanças estruturais dos programas, entre si, e entre linguagens distintas.

Função Módulo de X:

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***abs****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***abs*** *= Math.****abs****(i);*

*System.out.println(****abs****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

Função Exponencial Natural de X:

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***exp****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***exp*** *= Math.****exp(****i);*

*System.out.println(****exp****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

Função Logaritmo Natural de X:

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***log****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***log*** *= Math.****log****(i);*

*System.out.println(****log****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

Função Raiz de X:

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***sqrt****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***sqrt*** *= Math.****sqrt****(i);*

*System.out.println(****sqrt****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

Função Cosseno de X:

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***cos****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***cos*** *= Math.****cos****(i);*

*System.out.println(****cos****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

Função Seno de X:

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***sin****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***sin*** *= Math.****sin****(i);*

*System.out.println(****sin****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

Função Tangente de X:

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***tan****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***tan*** *= Math.****tan****(i);*

*System.out.println(****tan****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

Função Arco Cosseno de X:

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***acos****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***acos*** *= Math.****acos****(i);*

*System.out.println(****acos****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

Função Arco Seno de X:

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***asin****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***asin*** *= Math.****asin****(i);*

*System.out.println(****asin****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

Função Arco Tangente de X:

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***atan****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***atan*** *= Math.****atan****(i);*

*System.out.println(****atan****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

No caso do ASSEMBLY, foram elaborados dois métodos de associação à linguagem C, sendo Inline (In) o método de utilizar ASEMBLY no próprio programa em C, e Módulos separados (Sep) a chamada de procedures externas ao programa principal em C.

O método Inline é preferível quando se tem pouca instrução ASSEMBLY para embarcar. A sintaxe de ASSEMBLY para In, diferentemente do método Sep, que utiliza notação INTEL, é a arquitetura AT&T, que possui algumas diferenças. No código, essas diferenças podem influenciar no tempo de execução.

Para o método de módulos separados, mais comum quando há muitas instruções, é preciso passar o .asm (extensão do arquivo ASSEMBLY embarcado) por um montador, antes de compilar o arquivo.C. Ao utilizar as funções embarcadas, é necessário declarar “extern” antes da função.

* 1. **Linguagem ASSEMBLY associada a programa em C**

Como será utilizada a linguagem C, serão mantidas as bibliotecas utilizadas anteriormente:

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <conio.h>*

*#include <math.h>*

*#include <sys\timeb.h>*

Entretanto, em vez de utilizar funções internas à main(), os programas foram escritos com chamadas a funções fora do main(), mas pertencentes ao arquivo C. Isso gera um gasto de poucos milissegundos em relação à técnica anterior, pois, quando os comandos são internos ao main(), o compilador “lê” o código sequencialmente, enquanto a chamada da função vai fazer a busca para o endereço em que a função foi alocada, e depois retornar o valor ao último endereço antes da execução.

Função Módulo de X:

*double ASMabs(double x) {*

*double ans;*

*asm(*

*"fldl %1;" //carrega x em st0*

*"fabs;" //abs*

*:"=st"(ans) //saida: st0*

*:"m"(x) //entrada: x (na memoria)*

*);*

*return ans;*

*}*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double MODULO;*

*for(i=-100000;i<100000;i++){*

*MODULO = ASMabs(i);*

*printf("\n%lf", MODULO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Exponencial Natural de X:

*double ASMexp(double x) {*

*double ans;*

*double e = 2.71828182846;*

*asm(*

*"fldl %1;"*

*"fldl %2;"*

*"fyl2x;"*

*"fld1;"*

*"fld %%st(1);"*

*"fprem;"*

*"f2xm1;"*

*"faddp;"*

*"fscale;"*

*"fxch %%st(1);"*

*"fstp %%st"*

*:"=st"(ans)*

*:"m"(x), "m"(e)*

*);*

*return ans;*

*}*

*void main(){*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double EXPONENCIAL;*

*for(i=-100000;i<100000;i++){*

*EXPONENCIAL = ASMexp(i);*

*printf("\n%lf", EXPONENCIAL);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Logaritmo Natural de X:

*double ASMln(double x) {*

*double ans;*

*asm("fldl %1;"*

*"fld1;"*

*"fxch;"*

*"fyl2x;"*

*"fldl2e;"*

*"fxch;"*

*"fdivp;"*

*:"=st"(ans)*

*:"m"(x)*

*);*

*return ans;*

*}*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double LOGARITMO;*

*for(i=-100000;i<100000;i++){*

*LOGARITMO = ASMln(i);*

*printf("\n%lf", LOGARITMO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Raiz de X:

*double ASMsqrt(double x){*

*double ans;*

*asm(*

*"fldl %1;"*

*"fsqrt"*

*:"=st"(ans)*

*:"m"(x)*

*);*

*return ans;*

*}*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double RAIZ;*

*for(i=-100000;i<100000;i++){*

*RAIZ = ASMsqrt(i);*

*printf("\n%lf", RAIZ);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Cosseno de X:

*double ASMcos(double x) {*

*double ans;*

*asm("fldl %1;"*

*"fcos"*

*:"=st"(ans)*

*:"m"(x)*

*);*

*return ans;*

*}*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double COSSENO;*

*for(i=-100000;i<100000;i++){*

*COSSENO=ASMcos(i);*

*printf("\n%lf", COSSENO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Seno de X:

*double ASMsin(double x) {//WORKING*

*double ans;*

*asm("fldl %1;"*

*"fsin"*

*:"=st"(ans)*

*:"m"(x)*

*);*

*return ans;*

*}*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double SENO;*

*for(i=-100000;i<100000;i++){*

*SENO = ASMsin(i);*

*printf("\n%lf", SENO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Tangente de X:

*double ASMtan(double x) {//WORKING*

*double ans;*

*asm("fldl %1;"*

*"fptan;"*

*"fstp %%st;"*

*:"=st"(ans)*

*:"m"(x)*

*);*

*return ans;*

*}*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double TANGENTE;*

*for(i=-100000;i<100000;i++){*

*TANGENTE = ASMtan(i);*

*printf("\n%lf", TANGENTE);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Arco Cosseno de X:

*double ASMacos(double x) {//WORKING*

*double ans;*

*asm("fldl %1;"*

*"fldl %1;"*

*"fmulp;"*

*"fld1;"*

*"fsubp;"*

*"fsqrt;"*

*"fldl %1;"*

*"fpatan;"*

*:"=st"(ans)*

*:"m"(x)*

*);*

*return ans;*

*}*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double ARCOCOSSENO;*

*for(i=-100000;i<100000;i++){*

*ARCOCOSSENO = ASMacos(i);*

*printf("\n%lf", ARCOCOSSENO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Arco Seno de X:

*double ASMasin(double x) {*

*double ans;*

*asm(*

*"fldl %1;"*

*"fldl %1;"*

*"fldl %1;"*

*"fmulp;"*

*"fld1;"*

*"fsubp;"*

*"fsqrt;"*

*"fpatan;"*

*:"=st"(ans)*

*:"m"(x)*

*);*

*return ans;*

*}*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double ARCOSENO;*

*for(i=-100000;i<100000;i++){*

*ARCOSENO = ASMasin(i);*

*printf("\n%lf", ARCOSENO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Arco Tangente de X:

*double ASMatan(double x) {//WORKING*

*double ans;*

*asm("fldl %1;"*

*"fld1;"*

*"fpatan;"*

*:"=st"(ans)*

*:"m"(x)*

*);*

*return ans;*

*}*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double ARCOTANGENTE;*

*for(i=-100000;i<100000;i++){*

*ARCOTANGENTE = ASMatan(i);*

*printf("\n%lf", ARCOTANGENTE);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Para o ASSEMBLY (Sep)

//ASSEMBLY EXTERNO

*; double ans;*

*; asm("fldl %1;" //carrega x em st0*

*; "fabs;" //abs*

*; :"=st"(ans) //saida: st0*

*; :"m"(x) //entrada: x (na memoria)*

*; );*

*; return ans;*

*section .text*

*global \_ASMabs*

*\_ASMabs:*

*enter 0,0*

*fld qword [ebp + 8]*

*fabs*

*leave*

*ret*

//C

*extern double ASMabs(double x);*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double MODULO;*

*for(i=-100000;i<100000;i++){*

*MODULO = ASMabs(i);*

*printf("\n%lf", MODULO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Exponencial Natural de X:

//ASSEMBLY EXTERNO

*;double ASMexp(double x) {//WORKING*

*; double ans;*

*; double e = 2.71828182846;*

*; asm("fldl %1;"*

*; "fldl %2;"*

*; "fyl2x;"*

*; "fld1;"*

*; "fld %%st(1);"*

*; "fprem;"*

*; "f2xm1;"*

*; "faddp;"*

*; "fscale;"*

*; "fxch %%st(1);"*

*; "fstp %%st"*

*; :"=st"(ans)*

*; :"m"(x), "m"(e)*

*; );*

*; return ans;*

*;}*

//C

*extern double ASMexp(double x);*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double EXPONENCIAL;*

*for(i=-100000;i<100000;i++){*

*EXPONENCIAL = ASMexp(i);*

*printf("\n%lf", EXPONENCIAL);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Logaritmo Natural de X:

//ASSEMBLY EXTERNO

*;double ASMln(double x) {//WORKING*

*; double ans;*

*; asm("fldl %1;"*

*; "fld1;"*

*; "fxch;"*

*; "fyl2x;"*

*; "fldl2e;"*

*; "fxch;"*

*; "fdivp;"*

*; :"=st"(ans)*

*; :"m"(x)*

*; );*

*; return ans;*

*;}*

*section .text*

*global \_ASMln*

*\_ASMln:*

*enter 0, 0*

*fld qword [ebp + 8]*

*fld1*

*fxch*

*fyl2x*

*fldl2e*

*fxch*

*fdivp*

*leave*

*ret*

//C

*extern double ASMln(double x);*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double LOGARITMO;*

*for(i=-100000;i<100000;i++){*

*LOGARITMO = ASMln(i);*

*printf("\n%lf", LOGARITMO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

*package pontoflutuante;*

*public class Pontoflutuante{*

*public static void main(String[] args) {*

*int i;*

*double* ***log****;*

*long initialTime = System.currentTimeMillis();*

*for(i=-100000;i<100000;i++){*

***log*** *= Math.****log****(i);*

*System.out.println(****log****);*

*}*

*long endTime = System.currentTimeMillis();*

*System.out.println("Tempo total em milissegundos: " +(endTime-initialTime));*

*}*

*}*

Função Raiz de X:

//ASSEMBLY EXTERNO

*;double ASMsqrt(double x) {//WORKING*

*; double ans;*

*; asm("fldl %1;"*

*; "fsqrt"*

*; :"=st"(ans)*

*; :"m"(x)*

*; );*

*; return ans;*

*;}*

*section .text*

*global \_ASMsqrt*

*\_ASMsqrt:*

*enter 0,0*

*fld qword[ebp + 8]*

*fsqrt*

*leave*

*ret*

//C

\*

*extern double ASMsqrt(double x);*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double RAIZ;*

*for(i=-100000;i<100000;i++){*

*RAIZ = ASMsqrt(i);*

*printf("\n%lf", RAIZ);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Cosseno de X:

//ASSEMBLY EXTERNO

*;double ASMcos(double x) {//WORKING*

*; double ans;*

*; asm("fldl %1;"*

*; "fcos"*

*; :"=st"(ans)*

*; :"m"(x)*

*; );*

*; return ans;*

*;}*

*section .text*

*global \_ASMcos*

*\_ASMcos:*

*enter 0, 0*

*fld qword [ebp + 8]*

*fcos*

*leave*

*ret*

//C

*extern double ASMcos(double x);*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double COSSENO;*

*for(i=-100000;i<100000;i++){*

*COSSENO=ASMcos(i);*

*printf("\n%lf", COSSENO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Seno de X:

//ASEMBLY EXTERNO

*;double ASMsin(double x) {//WORKING*

*; double ans;*

*; asm("fldl %1;"*

*; "fsin"*

*; :"=st"(ans)*

*; :"m"(x)*

*; );*

*; return ans;*

*;}*

*section .text*

*global \_ASMsin*

*\_ASMsin:*

*enter 0,0*

*fld qword [ebp + 8]*

*fsin*

*leave*

*ret*

//C

*extern double ASMsin(double x);*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double SENO;*

*for(i=-100000;i<100000;i++){*

*SENO = ASMsin(i);*

*printf("\n%lf", SENO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Tangente de X:

//ASSEMBLY EXTERNO

*;double ASMtan(double x) {//WORKING*

*; double ans;*

*; asm("fldl %1;"*

*; "fptan;"*

*; "fstp %%st;"*

*; :"=st"(ans)*

*; :"m"(x)*

*; );*

*; return ans;*

*;}*

*section .text*

*global \_ASMtan*

*\_ASMtan:*

*enter 0, 0*

*fld qword [ebp + 8]*

*fptan*

*fstp ST0*

*leave*

*ret*

//C

*extern double ASMtan(double x);*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double TANGENTE;*

*for(i=-100000;i<100000;i++){*

*TANGENTE = ASMtan(i);*

*printf("\n%lf", TANGENTE);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Arco Cosseno de X:

//ASSEMBLY EXTERNO

*;double ASMacos(double x) {//WORKING*

*; double ans;*

*; asm("fldl %1;"*

*; "fldl %1;"*

*; "fmulp;"*

*; "fld1;"*

*; "fsubp;"*

*; "fsqrt;"*

*; "fldl %1;"*

*; "fpatan;"*

*; :"=st"(ans)*

*; :"m"(x)*

*; );*

*; return ans;*

*;}*

*section .text*

*global \_ASMacos*

*\_ASMacos:*

*enter 0, 0*

*fld qword [ebp + 8]*

*fld qword [ebp + 8]*

*fmulp*

*fld1*

*fsubp*

*fsqrt*

*fld qword [ebp + 8]*

*fpatan*

*leave*

*ret*

//C

*extern double ASMacos(double x);*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double ARCOCOSSENO;*

*for(i=-100000;i<100000;i++){*

*ARCOCOSSENO = ASMacos(i);*

*printf("\n%lf", ARCOCOSSENO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Arco Seno de X:

//ASSEMBLY EXTERNO

*;double ASMasin(double x) {//WORKING*

*; double ans;*

*; asm("fldl %1;"*

*; "fldl %1;"*

*; "fldl %1;"*

*; "fmulp;"*

*; "fld1;"*

*; "fsubp;"*

*; "fsqrt;"*

*; "fpatan;"*

*; :"=st"(ans)*

*; :"m"(x)*

*; );*

*; return ans;*

*;}*

*section .text*

*global \_ASMasin*

*\_ASMasin:*

*enter 0, 0*

*fld qword [ebp + 8]*

*fld qword [ebp + 8]*

*fld qword [ebp + 8]*

*fmulp*

*fld1*

*fsubp*

*fsqrt*

*fpatan*

*leave*

*ret*

//C

*extern double ASMasin(double x);*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double ARCOSENO;*

*for(i=-100000;i<100000;i++){*

*ARCOSENO = ASMasin(i);*

*printf("\n%lf", ARCOSENO);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

Função Arco Tangente de X:

//ASSEMBLY EXTERNO

*;double ASMatan(double x) {//WORKING*

*; double ans;*

*; asm("fldl %1;"*

*; "fld1;"*

*; "fpatan;"*

*; :"=st"(ans)*

*; :"m"(x)*

*; );*

*; return ans;*

*;}*

*section .text*

*global \_ASMatan*

*\_ASMatan:*

*enter 0, 0*

*fld qword [ebp + 8]*

*fld1*

*fpatan*

*leave*

*ret*

//C

*extern double ASMatan(double x);*

*void main() {*

*struct timeb start, end;*

*int diff;*

*ftime(&start);*

*int i;*

*double ARCOTANGENTE;*

*for(i=-100000;i<100000;i++){*

*ARCOTANGENTE = ASMatan(i);*

*printf("\n%lf", ARCOTANGENTE);*

*}*

*ftime(&end);*

*diff = (int) (1000.0 \* (end.time - start.time)+(end.millitm - start.millitm));*

*printf("\n\nTempo total em milissegundos: %u\n", diff);*

*system("pause");*

*}*

1. **RESULTADOS E DISCUSSÕES**

Para a linguagem C, foram obtidos os seguintes valores:



Tabela 1 – Tempos de execução em milissegundos para linguagem C.

Para a linguagem JAVA, foram obtidos os seguintes valores:



Tabela 2 – Tempos de execução em milissegundos para linguagem JAVA­­.

Para o Programa C, com procedures elaboradas em ASSEMBLY(In), foram obtidos os seguintes valores:



Tabela 3 – Tempos de execução em milissegundos para ASSEMBLY (In).

Para o Programa C, com procedures elaboradas em ASSEMBLY (Sep), foram obtidos os seguintes valores:



Tabela 4 – Tempos de execução em milissegundos para ASSEMBLY (Sep).

De acordo com os resultados explicitados na tabela de valores médios e gráfico correspondente abaixo:



Tabela 5 – Tempos médios das linguagens.

Gráfico 1 – Comparação de tempos médios de execução das linguagens observadas.

É possível perceber a regularidade dos intervalos. Em todas as funções, a ordem crescente de tempos foi: C < C/ASSEMBLY (In) < JAVA < C/ASSEMBLY (Sep). Os resultados das operações dentro das linguagens, no caso de C e C/ASSEMBLY (In), também foram bem homogêneos.

Os Ganhos considerados ente C e C/ASSEMBLY (In) foram como se pode observar na tabela:



Tabela 6 – Ganhos em Tempo de Execução.

Antes de elucidar sobre os resultados, é preciso conhecer um pouco mais as linguagens.

C é uma linguagem imperativa e procedural, para implementação de sistemas. Foi desenvolvida para ser compilada, fornecendo acesso de baixo nível à memória e baixos requerimentos do hardware. Também foi desenvolvida para ser uma linguagem de alto nível, para maior reaproveitamento do código. C foi útil para muitas aplicações que foram codificadas originalmente em Assembly. De forma sucinta, a linguagem C já utiliza internamente rotinas em Assembly e o compilador utiliza o código-fonte em C para gerar um executável em código nativo.

A linguagem JAVA, entretanto, tem seu código fonte compilada para um Bytecode, que, em vez de passar diretamente ao Sistema Operacional, é enviado para a JVM (Java Virtual Machine), que garante a leitura de um mesmo programa JAVA para diversos sistemas. É a JVM quem passa o programa para o sistema para ser lido pela máquina.

Assim, o JAVA possui uma etapa a mais, tornando o processo de compilação e execução maior que o realizado na linguagem C. Assim, era de se esperar que o tempo gasto pelo programa em C fosse bem menor que o necessário para o JAVA. Essa diferença se torna irrisória em máquinas com maior capacidade de memória RAM, pois, devido à própria evolução da linguagem, o JAVA é capaz de executar mais processos em paralelo (programação multi-thread) sendo, neste caso, necessário um maior intervalo de iteração para que o C volte a ser mais rápido.

Para o ASSEMBLY, percebe-se que o Inline resultou em tempos muito próximos (pouco maiores que C puro), o que era de se esperar, pois a linguagem C, por si só, é construída em ASSEMBLY, e totalmente Otimizada, sendo a melhor possibilidade dentro dos limites da máquina utilizada. Quanto ao ASSEMBLY em módulos separados, percebe-se que as medidas de tempo superaram os valores obtidos para a linguagem JAVA, o que pode indicar que os testes não foram adequados.

O primeiro ponto a ser considerado é que as funções matemáticas para o ASSEMBLY foram implementadas diretamente, enquanto para C e JAVA, foram utilizadas funções prontas. A justificativa para tal é que as funções “acos”, “asin”, e “atan” não são nativas de ASSEMBLY e, portanto, não seria possível analisá-las diretamente, sendo necessário misturar os métodos, o que tornaria o método de comparação injusto. Logo, A comparação foi específica entre as formas In e Sep, que, como é possível observar, resultou em grandezas bem distintas.

Outra consideração a ser feita é que o instante final de tempo foi medido somente ao final do loop. Sabendo que o sistema operacional tende a manipular e evitar loops, sendo a primeira forma de contornar isso, a contagem acumulada de instantes finais dentro de cada iteração. Há ainda a possibilidade de medir diretamente a frequência do clock da máquina, que toma o número de ciclos por operação, que resultaria em um valor mais verdadeira.

Portanto, a comparação do ganho da linguagem C foi feita apenas para o ASSEMBLY Sep e o resultado foi menor que 100%, indicando o o tempo gasto no puro C é menor.

1. **CONCLUSÃO**

Tem-se que o ganho é de cerca de 64%, o que torna mais prático utilizar operações diretamente em Linguagem C, dispensando a necessidade de elaborar um programa “misto” (utilizando funções matemáticas C e embarcando ASSEMBLY simultaneamente). Diante dos resultados apresentados e métodos de medição utilizados, C tem um tempo menor, sendo a opção mais considerável para cálculos matemáticos extensos utilizando notação de ponto flutuante.

1. **REFERÊNCIAS**

IBM. Disponível em: <https://www.ibm.com/developerworks/br/java/tutorials/j-introtojava1/index.html>. Acesso em 16 de Dezembro de 2017.

Tutorials Point. Disponível em: <https://www.tutorialspoint.com/c\_standard\_library/math\_h.htm/>. Acesso em 10 de Dezembro de 2017

Microsoft. Disponível em: < https://msdn.microsoft.com/pt-br/library/windows/desktop/ms724390(v=vs.85).aspx>. Acesso em 13 de Dezembro de 2017.

Oracle. Disponível em: <https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html>. Acessado em 12 de Dazembro de 2017.