Pseudocódigo:

**Thread0**

**Variaveis Compartilhadas:**

Barreira = semáforo do tipo barreira inicializado em 0

Elementos[n] = vetor de n elementos a serem processados pela thread1

P = num\_processadores();

maxV = aleatório de 1 a 10;

maxM = aleatório de 1 a 10;

maxN = aleatório de 1 a 10;

n\_elementos = (maxM \* maxN)/p;

resto = (maxM \* maxN) % p;

A = gerar\_matriz\_aleatoria(maxM,maxV);

B = gerar\_matriz\_aleatoria(maxV,maxN);

K = 0;

Para i de 0 até maxM

Para j de 0 até maxN

Criar elemento “e” com: A[i], coluna de B, i e j;

Elementos.adicionar(e);

cont++;

se cont == n\_elementos

se resto <> 0

j++

se j >= maxN

i++;

se i >= maxM

pare a repetição;

senão

j = 0;

fimse

fimse

Criar elemento “e” com: A[i], coluna de B, i e j;

Elementos.adicionar(e);

Resto = resto – 1;

Fimse

Threads[k] = nova thread1(barreira,elementos,n);

K++;

Cont = 0;

Resetar o vetor Elementos;

fimse

Fimpara

Fimpara

Para i de 0 até p

Threads[i].start();

Fimpara

Barreira.esperar(p);

Para i de 0 ate p

Para j de 0 ate tamanhoMax de threads[i].Elementos

C[threads[i].Elementos[j].getM()][threads[i].Elementos[j].getN()] = threads[i].Elementos[j].get\_resultado();

fimpara

fimpara

printar\_matriz(C);

gerar\_arquivo(A);

gerar\_arquivo(B);

gerar\_arquivo(C);

**Thread1:**

**Variaveis compartilhadas:**

Barreira = semáforo do tipo barreira inicializado em 0

Elementos[n] = vetor de n elementos a serem processados pela thread1

Para i de 0 ate tamanhoMax de Elementos

Elementos[i].calcular\_resultado();

Fimpara

Barreira.sinalizar();

**Elemento:**

Classe elemento

{

Privado linhasdeA[];

Privado colunasdeB[];

Privado m;

Privado n;

Privado resposta;

Publico calcula\_resultado()

{

Para i de 0 até tamanhoMax de linhasdeA

Resposta = resposta + linhasdeA[i] \* colunasdeB[i];

fimpara

}

}