**Disciplina**: Performance em Sistemas Ciberfisicos

**Professor:** Guilherme Schnirmann

**Nome Estudante: Ary Felipe Farah e Silva**

**Atividade Prática / Relatório**

**Desempenho – Exercícios**

1. Considere duas máquinas com conjuntos de instruções diferentes. Os computadores tem uma frequência de 300 MHz. As medições a seguir foram registradas executando programas de um benchmark:

|  |  |  |
| --- | --- | --- |
| **Tipo de instrução** | **Número de instruções (milhões)** | **Ciclos por tipo de instrução** |
| **Máquina A** | | |
| Aritmética e lógica | 8 | 1 |
| Load e store | 4 | 3 |
| Desvios | 2 | 4 |
| Outros | 4 | 3 |
| **Máquina B** | | |
| Aritmética e lógica | 10 | 1 |
| Load e store | 8 | 2 |
| Desvios | 2 | 4 |
| Outros | 4 | 3 |

1. Calcule o CPI médio para cada máquina

**A**: 8x1 + 4x3 + 2x4 + 4x3 = 40 milhões de ciclos / 18 milhões de instruções ~ 2.23

**B**: 10x1 + 8x2 + 2x4 + 4x3 = 46 milhões de ciclos / 24 milhões de instruções ~ 1.92

1. Calcule os tempos de execução

**A** - 40x~~10^6 ciclos~~

300x~~10^6 ciclos por segundo~~

40/300 = 0.14

**B** - 46x~~10^6 ciclos~~

300x~~10^6 ciclos por segundo~~

46/300 = 0.15

1. Calcule a taxa MIPS para cada máquina (pesquise o que é essa taxa).

**MIPS - Milhões de Instruções por Segundo**

Avalia a capacidade de processamento de uma máquina

MIPS = nº instruções executadas / tempo de execução em segundos

**A** – 18x~~10^6~~ / 0.14~~x10^6~~ = 18/0.14 ~ 128 MIPS

**B** – 24x~~10^6~~ / 0.15x~~10^6~~ = 24/0.15 ~ 160 MIPS

1. Sabendo que o desempenho de cada máquina é calculado por 1/(tempo de execução), calcule o desempenho de cada máquina e compare.

**A** = 1/0.14 = 7.14 execuções por segundo

**B** = 1/0.15 = 6.67 execuções por segundo

1. A razão entre os desempenhos é conhecida como ***speed-up.*** Calcule e interprete.

Indica quantas vezes uma máquina é mais rápida que a outra

**Speed-up** = Desempenho A / Desempenho B

**s = 7.14 / 6.67 = 1.07** = a máquina A é aproximadamente 7% mais rápida que a máquina B, completando tarefas 1.7 vezes mais rapidamente.

1. Explique os três tipos de hazards em pipelines. Utilize exemplos para fundamentar suas respostas.

**Hazard Estrutural** – quando o hardware não suporta a execução paralela (ex: memória com uma porta só).

**Solução**: Atrasar o ciclo com conflito

**Hazard de Controle** – desvios / alterações de fluxo no programa, podendo fazer o pipeline carregar instruções desnecessárias ou fora de ordem.

**Solução**: Branch prediction / delayed branch

**Hazard de Dados** – Instrução depende do resultado de outra instrução que ainda está no pipeline, resultando em dependência de dados ainda não produzidos ou em trânsito

**Solução**: forwarding/bypassing

1. Pesquise, explique e exemplifique o que são as “bolhas” ou *stalls* no contexto de pipelining.

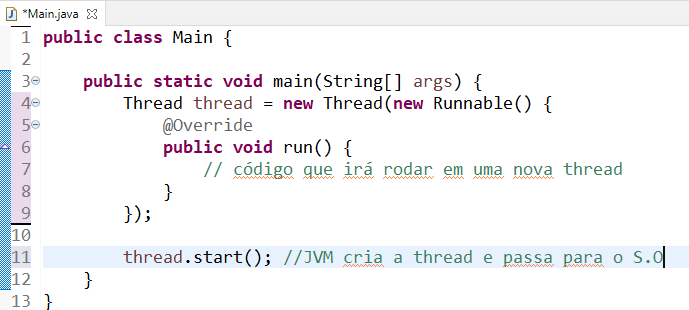
Bolhas (stalls) são pausas temporárias (ciclos de clock) no pipeline para resolver conflitos, como os hazards citados acima. Durante essa pausa, não são processadas novas instruções, mas sim um vazio, como uma bolha de ar.

1. Explique de forma objetiva o que são os computadores superescalares.

São sistemas de procesamento que possuem múltiplas unidades de execução, podendo executar várias instruções em paralelo desde que não exista dependência entre elas. Então, os computadores superescalares exploram o paralelismo a nível de instrução para aumentar o desempenho.

1. **INTRODUÇÃO A THREADS EM JAVA**

*Em JAVA todas as propriedades e métodos estão encapsulados no corpo JDK. Assim, para criarmos uma thread instanciamos um objeto do tipo Thread. Passamos um objeto da classe que implementa a interface Runnable. O que inserirmos no método run será executado (desde que a thread seja agendada pelo S.O).*

**

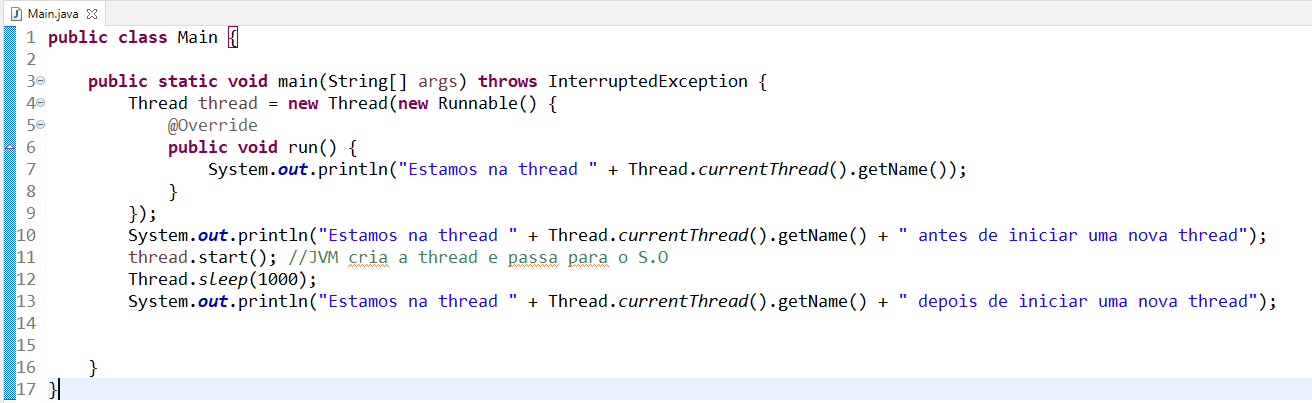
Replique o código (caso faça em python, pesquise como retornar o nome da thread). Atribua um nome a thread utilizando o método ***setName*().** Ainda, atribua uma prioridade utilizando ***setPriority().*** *Faça um teste utilizando a constante* ***Thread.MAXPRIORITY.***Apresente o print dos seus testes com o nome dado e a prioridade.

Texto

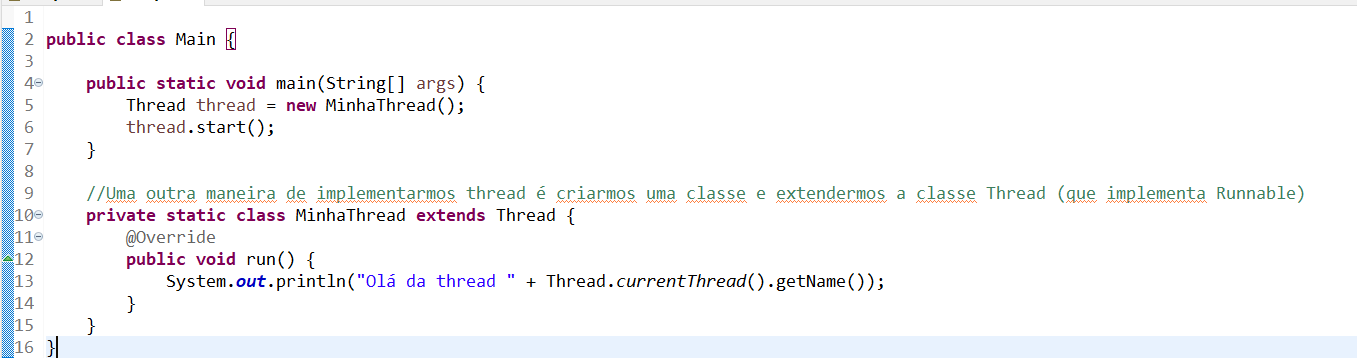
Descrição gerada automaticamente

Texto

Descrição gerada automaticamente



**Uma outra maneira de implementarmos thread é criarmos uma classe e extendermos a classe Thread (que implementa Runnable). Agora conseguimos utilizar “this”.**



Texto

Descrição gerada automaticamente**Crie um programa em que duas threads são criadas (t1 e t2). No caso do JAVA, instancie dois objetos da sua classe Thread (MinhaThread) em um classe chamada Processo, por exemplo. Inicialize as duas threads e coloque um breakpoint no método run(). Observe e reporte quantos Threads existem.**

**Exercício:**

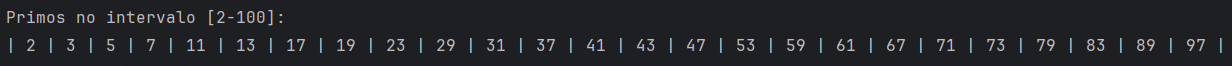
**Faça um programa que dado um intervalo (numérico), imprima todos os números primos existentes.**

**Exemplo:**

**Intervalo [2;10]**

**Primos: 2,3,5,7**

1. **Faça apenas o processo (sem threads)**



1. **Criar um thread que imprime os primos no intervalo**



1. **Criar 2 threads: Uma para primeira metade (começo até metade) e outra para (metade +1 até final).**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

1. **Criar vários threads, um para cada sub-intervalo.**

**Exemplo: 5-50 ; 50-100; 100-150; 150-200**

Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente

1. **Faça testes com intervalos grandes (na casa de milhões). Qual dos exemplos anteriores é o caso mais rápido?**

**Início = 2**

**Fim = 1000000**

**Como vemos abaixo, o programa com duas threads foi o mais rápido.**

1. **Utilize o powershell para medir o tempo: *measure-command { java meu programa}. Para isso crie um programa para cada versão e meça o tempo para intervalos iguais.***

Uma imagem contendo Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente**Primos: PrimosThread:**

Texto

Descrição gerada automaticamente com confiança médiaTexto

Descrição gerada automaticamente com confiança baixa**PrimosDuasThreads PrimosNThreads (4)**