Trabalho seminarios II
Aluno : Gustavo Lopes Rodrigues
Professor : Saulo Augusto de Paula Pinto

Questão 1
a) Toda vez que o programa é rodado , a saída do mesmo é diferente.
Ex:
-
Oi mundo! Sou a thread 6
Oi mundo! Sou a thread 0
Oi mundo! Sou a thread 7
Oi mundo! Sou a thread 2
Oi mundo! Sou a thread 3
Oi mundo! Sou a thread 5
Oi mundo! Sou a thread 4
Oi mundo! Sou a thread 1
-

-
Oi mundo! Sou a thread 6
Oi mundo! Sou a thread 0
Oi mundo! Sou a thread 2
Oi mundo! Sou a thread 5
Oi mundo! Sou a thread 4
Oi mundo! Sou a thread 3
Oi mundo! Sou a thread 1
Oi mundo! Sou a thread 7
-

a justificativa do porquê isso acontece, é devido ao fato de que: quando o paralelismo se inicia, varias threads são criadas, e a ordem que elas são criadas é arbitrária, ou seja, dependendo de quanto for a carga de cada núcleo, alguns vão executar a criação da thread mais rápido do que outros, consequentemente cada thread vai ter um número diferente , logo, a execução do código vai sempre resultar um resultado diferente.

b) A minha máquina possui 8 núcleos, sabendo isso, justifica porque durante a execução do programa os threads vão de 0 a 7.

Modelo da máquina:

Memória RAM: 8 Gb(7,7)
Processador: Intel® Core™ i5-8250U CPU @ 1.60GHz × 8
Placa de vídeo: Intel® UHD Graphics 620 (Kabylake GT2)
Tipo de Sistema : 64 bits

Questão 2

O padrão “fork-join” divide as tarefas entre os diferentes threads e depois junta-os no final.No caso desse exercicio, isso pode ser facilmente notado na funcao do tipo void “openMP2”, na execução ela imprime na tela o numero da thread e um laço “for” para imprimir numeros de 0 a 2. Ao executar o programa , é percebível pelas saidas que alguns threads apenas imprimem o numero do seu thread, mas não executam o laço “for” enquanto que outros threads podem executar uma ou mais vezes o laço “for”, em alguns casos os numeros imprimidos nem vao de 0 a 2, as vezes as threads imprimem só um dos numeros.

Exemplo 1 :
Região sequencial 1. Thread master. Número de threads: 1
Numero de processadores: 8
Entrando na primeira região paralela...
Oi mundo! Sou a thread 0
Oi mundo! Sou a thread 4
Oi mundo! Sou a thread 2
Oi mundo! Sou a thread 3
Oi mundo! Sou a thread 5
Oi mundo! Sou a thread 7
Oi mundo! Sou a thread 6
Oi mundo! Sou a thread 1
Fim da primeira região paralela...

Região sequencial 2. Thread master. Número de threads: 1
Numero de processadores: 8
Entrando na segunda região paralela...
Sou a thread 6
Sou a thread 1
i = 0
i = 1
i = 2
i = 0
i = 1
Sou a thread 3
i = 0
Sou a thread 4
Sou a thread 0
i = 0
Sou a thread 5
i = 2
Sou a thread 2
i = 1
i = 0
Sou a thread 7
i = 1
i = 2

i = 0
i = 2
i = 0
i = 1
i = 2
i = 1
i = 2
i = 0
i = 1
i = 2
i = 1
i = 2
Fim da segunda região paralela...

Região sequencial 3. Thread master. Número de threads: 1
Numero de processadores: 8
Terminando o programa…

Exemplo 2 :

Região sequencial 1. Thread master. Número de threads: 1
Numero de processadores: 8
Entrando na primeira região paralela...
Oi mundo! Sou a thread 0
Oi mundo! Sou a thread 6
Oi mundo! Sou a thread 2
Oi mundo! Sou a thread 1
Oi mundo! Sou a thread 3
Oi mundo! Sou a thread 4
Oi mundo! Sou a thread 5
Oi mundo! Sou a thread 7
Fim da primeira região paralela...

Região sequencial 2. Thread master. Número de threads: 1
Numero de processadores: 8
Entrando na segunda região paralela...
Sou a thread 6
i = 0
i = 1
i = 2
Sou a thread 7
i = 0
i = 1
i = 2
Sou a thread 2
i = 0
i = 1
i = 2

Sou a thread 1
i = 0
i = 1
i = 2
Sou a thread 3
i = 0
i = 1
i = 2
Sou a thread 4
i = 0
i = 1
i = 2
Sou a thread 5
i = 0
i = 1
i = 2
Sou a thread 0
i = 0
i = 1
i = 2
Fim da segunda região paralela...

Região sequencial 3. Thread master. Número de threads: 1
Numero de processadores: 8
Terminando o programa…

Questão 3

a) O motivo pelo qual alguns valores estão faltando, é porque todos os threads estão tentando acessar o mesmo espaço de memória, e como resultado, o valor de ‘i’ fica igual a ‘lixo’, fazendo com que os valores não sejam imprimidos corretamente na tela

b) Comparando o primeiro e o segundo código, ambos possui o mesmo número de threads e usam o mesmo tipo de paralelismo, a diferença é que no primeiro caso, não há memoria compartilhada entre os threads, enquanto que no segundo código, todos os threads usam a variável do tipo inteiro ‘i’.

Questão 4

a) No último código, cada um dos cinco threads usados são responsáveis por fazer um dos laços do for, ou seja, para cada execução do for , um thread recebe o trabalho de executá-lo.

b) O thread de número 0 ficou com uma iteração acrescentada, ao aumentar o numero de interações no laço “for”

c) Cada vez que o laço for tem interações icrementadas, os threads vão ganhando mais uma interação, no inicio cada um tinha uma interação, depois disso , o thread 0 ganhou uma interação, depois o thread

1 ganhou uma interação, depois o thread 2 ganhou uma interação e assim por diante. Quando todos os threads tem duas interações, a partir dai, todos os threads voltam a ganhar uma interação.

d) O máximo de threads que o meu computador pode ter é 1037 nesse programa, usando a mesma máquina:

Memória RAM: 8 Gb(7,7)
Processador: Intel® Core™ i5-8250U CPU @ 1.60GHz × 8
Placa de vídeo: Intel® UHD Graphics 620 (Kabylake GT2)
Tipo de Sistema : 64 bits