**1. Compile e execute o programa simples.c, disponíıvel nos exemplos de Pthreads. No Linux, isso pode ser feito com os comandos:**

****

**2. Modifique o programa simples.c para criar, além das threads que executam PrintHello(), outras NUM\_THREADS+3 threads que imprimam o quadrado do número recebido como parâmetro. (No total, portanto, o programa deve criar 2\*NUM\_THREADS+3 threads.)**

**3. Modifique o programa simples.c para que cada thread retorne o quadrado do número recebido como parâmetro. O programa principal deve imprimir a soma dos valores de retorno de todas as threads.**

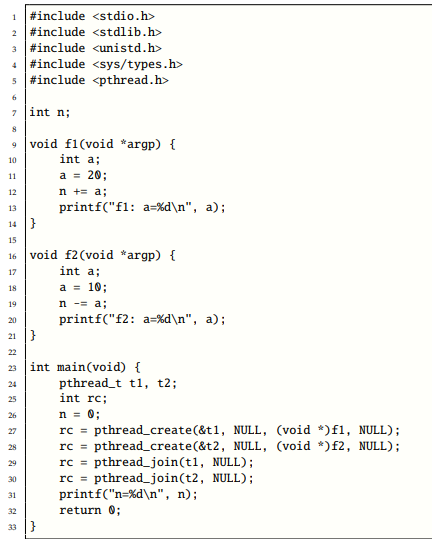
**4. O programa conta.c, disponível nos exemplos de Pthreads, conta o números de elementos pares e de elementos ímpares em um vetor de inteiros. Adapte esse programa conta.c para criar duas threads, uma para contar os pares e outra para contar os ímpares. Ambas as threads devem executar a função conta(). O vetor v pode ser definido como uma variável global, sendo o único parâmetro da função o tipo de número a ser contado. Execute o programa algumas vezes e veja se há diferença no tempo de execução das duas versões (sequencial e multithread).**

**5. Modifique o programa desenvolvido no exercício anterior de modo que tanto o vetor quanto o tipo de número sejam passados como parâmetro para a função conta().**

**6. Escreva um programa que conte ate 2 31 (1<<31). O programa deve ter N threads, e cada thread deve contribuir 231/N para o total. A contagem em cada thread deve ser em incrementos unitários (i.e., total\_thr++), e o valor final deve ser retornado para main() e somado ao total global. Caso a divisão de 2 31 por N seja inexata, o resto deve ser acrescido em main(), também em incrementos unitários. Verifique se, incrementando N, há variação no tempo de execução do programa. 1**

O arquivo pthread-ex-5.c implementa uma solução em que N e passado como parâmetro pela linha de comando (por exemplo, para N = 2, deve-se executar o programa com ./pthread-ex-5 2). A figura abaixo mostra o tempo de execução (em segundos) para N variando entre 1 e 8, em uma máquina com 4 núcleos, considerando a média de 5 rodadas. Observa-se que o tempo cai entre 1 e 4 threads (número de núcleos disponíveis), e a partir desse ponto ele torna-se relativamente estável.

**7. Considere o código abaixo:**

****

**O que será impresso pelo código acima?**

(a) f1: a=20

f2: a=10

n=10

(b) f2: a=10

f1: a=20

n=10

(c) f1: a=20

f2: a=10

n=-10

(d) f2: a=10

f1: a=20

n=-10

(e) f1: a=20

f2: a=10

n=20

(f) f2: a=10

f1: a=20

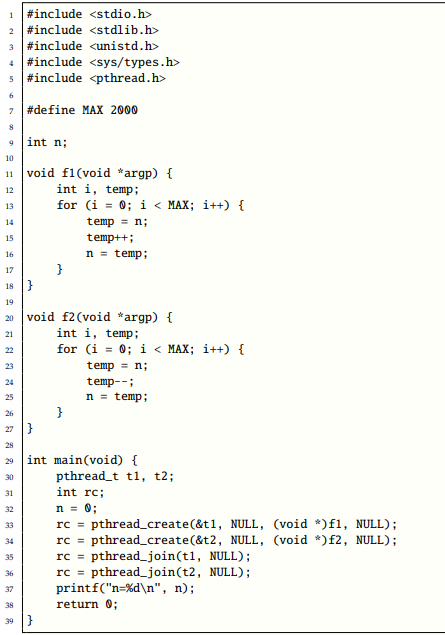
n=20

As saídas (a) e (b) são as mais prováveis, e envolvem apenas a mudança de ordem dos printf()s das linhas 13 e 20. As saídas (c) a (f) ocorrem com menor frequência, pois envolvem sequências mais improváveis de execução. Para visualizar tais sequências, primeiro é preciso considerar como as linhas 12 (n += a) e 19 (n -= a) seriam traduzidas para linguagem de máquina (REG e um registrador de propósito geral):



Como essas instruções podem ser intercaladas, uma thread pode incrementar ou decrementar o valor desatualizado da variável compartilhada. Por exemplo, a saída (c) poderia ser produzida pela sequência 19a, 12, 13, 19b, 19c, 20, 31, e a saída (e) pela sequência 12a, 19, 12b, 12c, 13, 20, 31.

**8. Considere o código abaixo:**

****

**Qual será o valor de n impresso na linha 37?**

**Caso haja mais de um valor possível, enumere as possibilidades e justifique sua resposta.**

O valor final de n fica entre −2000 e +2000