Please write your name on top of EVERY PAGE.

1. (40 pts) Short answer questions.

   (a) Moore's law states that "the number of transistors in a dense integrated circuit doubles approximately every two years." Does this mean that the computing power of a processor doubles every two years? Justify your yes/no answer in one or two sentences.

   (b) Draw a schematic (from processor to RAM) of the cache-hierarhcy for a 2 CPU system with 2 cores per CPU.

   (c) Hyperthreading is Intel's implementation of simultaneous multithreading. Describe specifically what hyperthreading means with respect to:

      i. How are threads scheduled on processor cores?

      ii. How many cores does the operating system see?

   (d) What service does Amazon EC2 provide? (Respond in no more than 50 words.)

2. (20 pts) Amdahl's law and its relationship to speedup and scaleup. For a code with an Amdahl number of 0.98, i.e. 98% of the code is optimized and 2% not optimized:[†]

   (a) Draw a speedup chart that varies resources from 1 to 100. Please place and label one specific data point on the chart, i.e. (x, y) where x is resources and y is speedup.[‡]

   (b) What is the maximum speedup of this code given infinite computing resources? Add a dashed line to your speedup chart that shows this asymptotic limit.

   (c) Draw a scaleup chart that varies the resources from 1 to 100. Please place and label one specific data point on the chart, i.e. (x, y) where x is resources and y is scaleup.

[†] Instructor's note: When I ask you to draw charts, they should be approximate (not exact) but they need to travel through at least two points (1,1) and (x,y) as specified below.

[‡] Instructor's note: Neither x nor y need to be integers. They can be. They must solve your Amdahl's law equation.

3. (20 pts) OpenMP parallelization constructs and variable scoping.

```
int length;
unsigned int * ar;
unsigned int maxel = 0;

for (int i=0; i<length; i++)
{
  if (a[i] > maxel)
  {
    maxel = a[i];
  }
}
printf(”Max = %d\n”, max);
```

(a) Write an OpenMP parallel program that performs the same computation as the above code snippet. Good implementations (which yours should be) will minimize or eliminate writes to shared variables.

(b) Unless you specified otherwise, your program used the default OpenMP for loop scheduler static.

i. Describe how the static scheduler distributes loop iterations to threads.

ii. Why is static the optimal scheduler for this problem?

4. (20 pts) The Java random number generator uses a synchronized method to update its internal state. All methods are either synchronized or call the synchronized next function with prototype.

   protected synchronized int next(int bits) {....}

   Consider the following two implementations of a thread-parallel program:

Left (L) Implementation

```
import java.util.Random;

class Rnbl implements Runnable
{
  static int num_thrds = 4;
  static int trials = 10000000;
  int thread_id;

  Random generator = new Random ();

  public void run ()
  {
    for (int i=0;
            i<trials/num_thrds;
            i++ )
    {
      generator.nextInt();
    }
  }

  Rnbl ( int id )
  {
    this.thread_id = id;
  }

  public static void main ( )
  {
    Thread[] thrds = new Thread[num_thrds];
    for ( int i=0; i<num_thrds; i++ )
    {
      thrds[i] = new Thread (new Rnbl(i));
      thrds[i].start();
    }
  }
}
```

Right (R) Implementation

```
import java.util.Random;

class Rnbl implements Runnable
{
  static int num_thrds = 4;
  static int trials = 10000000;
  int thread_id;

  static Random generator;

  public void run ()
  {
    for (int i=0;
            i<trials/num_thrds;
            i++ )
    {
      generator.nextInt();
    }
  }

  Rnbl ( int id )
  {
    this.thread_id = id;
  }

  public static void main ( )
  {
    Thread[] thrds = new Thread[num_thrds];
    generator = new Random ();
    for ( int i=0; i<num_thrds; i++ )
    {
      thrds[i] = new Thread (new Rnbl(i));
      thrds[i].start();
    }
  }
}
```

(a) Which implementation would you expect to perform better (run faster). Briefly explain your answer. Include a sentence that states which of the factor(s) against parallelism (startup, skew interference) causes the difference in runtime?

(b) The implementation of Random synchronizes all function invocations within an object. In contrast, your homework synchronized all objects in a class. Characterize the amount of parallelism available with respect to the number of Random objects instantiated.

Scratch space.