Οι σημειώσεις αυτές βασίζονται στο:

A “hands-on” introduction to OpenMP by Tim Mattson

http://www.openmp.org/wp-content/uploads/Intro_To_OpenMP_Mattson.pdf

Υπολογισμός του αριθμού π με τον τύπο:

$$\int_0^1 \frac{4}{1+x^2} dx = \pi$$

1. False sharing.

```c
#include <stdio.h>
#include <omp.h>
static long num_steps = 100000000;
double step;

void main() {
  for (int NUM_THREADS = 1; NUM_THREADS <= 8; NUM_THREADS++) {
    double start = omp_get_wtime(), finish;
    int i, nthreads;
    double pi, sum[NUM_THREADS];
    step = 1.0 / (double)num_steps;
    omp_set_num_threads(NUM_THREADS);
#pragma omp parallel
    {
      int i, id, nthrds;
      double x;
      id = omp_get_thread_num();
      nthrds = omp_get_num_threads();
      if (id == 0)
        nthreads = nthrds;
      for (i = id, sum[id] = 0.0; i < num_steps; i = i + nthrds) {
        x = (i + 0.5) * step;
        sum[id] += 4.0 / (1.0 + x * x);
      }
    }
    for (i = 0, pi = 0.0; i < nthreads; i++)
      pi += sum[i] * step;
    finish = omp_get_wtime();
    printf("Threads requested=%d Threads given=%d --> pi = %.9f time elapsed = "
           "%.4f\n",
           NUM_THREADS, nthreads, pi, finish - start);
  }
}
```

mattson_06a.c

```
gcc mattson_06a.c -o mattson_06a –fopenmp
```

```
./mattson_06a
```

```
Threads requested=1 Threads given=1 --> pi = 3.141592654 time elapsed = 0.8180
Threads requested=2 Threads given=2 --> pi = 3.141592654 time elapsed = 0.4790
Threads requested=3 Threads given=3 --> pi = 3.141592654 time elapsed = 0.3910
Threads requested=4 Threads given=4 --> pi = 3.141592654 time elapsed = 0.3940
Threads requested=5 Threads given=5 --> pi = 3.141592654 time elapsed = 0.3800
Threads requested=6 Threads given=6 --> pi = 3.141592654 time elapsed = 0.4050
Threads requested=7 Threads given=7 --> pi = 3.141592654 time elapsed = 0.3760
Threads requested=8 Threads given=8 --> pi = 3.141592654 time elapsed = 0.4280
```

2. Με padding έτσι ώστε να αποφευχθεί το false sharing.

```c
#include <stdio.h>
#include <omp.h>
#define PAD 8
static long num_steps = 100000000;
double step;

void main() {
  for (int NUM_THREADS = 1; NUM_THREADS <= 8; NUM_THREADS++) {
    double start = omp_get_wtime(), finish;
    int i, nthreads;
    double pi, sum[NUM_THREADS][PAD];
    step = 1.0 / (double)num_steps;
    omp_set_num_threads(NUM_THREADS);
#pragma omp parallel
    {
      int i, id, nthrds;
      double x;
      id = omp_get_thread_num();
      nthrds = omp_get_num_threads();
      if (id == 0)
        nthreads = nthrds;
      for (i = id, sum[id][0] = 0.0; i < num_steps; i = i + nthrds) {
        x = (i + 0.5) * step;
        sum[id][0] += 4.0 / (1.0 + x * x);
      }
    }
    for (i = 0, pi = 0.0; i < nthreads; i++)
      pi += sum[i][0] * step;
    finish = omp_get_wtime();
    printf("Threads requested=%d Threads given=%d --> pi = %.9f time elapsed = "
           "%.4f\n",
           NUM_THREADS, nthreads, pi, finish - start);
  }
}
```

mattson_06b.c

```
gcc mattson_06b.c -o mattson_06b –fopenmp
```

./mattson_06b
Threads requested=1 Threads given=1 --> pi = 3.141592654 time elapsed = 0.8450
Threads requested=2 Threads given=2 --> pi = 3.141592654 time elapsed = 0.4930
Threads requested=3 Threads given=3 --> pi = 3.141592654 time elapsed = 0.3590
Threads requested=4 Threads given=4 --> pi = 3.141592654 time elapsed = 0.2750
Threads requested=5 Threads given=5 --> pi = 3.141592654 time elapsed = 0.2440
Threads requested=6 Threads given=6 --> pi = 3.141592654 time elapsed = 0.2300
Threads requested=7 Threads given=7 --> pi = 3.141592654 time elapsed = 0.2140
Threads requested=8 Threads given=8 --> pi = 3.141592654 time elapsed = 0.2150

3. Με χρήση critical section

```c
#include <stdio.h>
#include <omp.h>
static long num_steps = 100000000;
double step;

void main() {
  for (int NUM_THREADS = 1; NUM_THREADS <= 8; NUM_THREADS++) {
    double start = omp_get_wtime(), finish, pi = 0.0;
    step = 1.0 / (double)num_steps;
    omp_set_num_threads(NUM_THREADS);
#pragma omp parallel
    {
      int i, id = omp_get_thread_num(), nthrds;
      double x, sum;
      for (i = id, sum = 0.0; i < num_steps; i = i + NUM_THREADS) {
        x = (i + 0.5) * step;
        sum += 4.0 / (1.0 + x * x);
      }
#pragma omp critical
      pi += sum * step;
    }
    finish = omp_get_wtime();
    printf("Threads=%d --> pi = %.9f time elapsed = "
           "%.4f\n",
           NUM_THREADS, pi, finish - start);
  }
}
```

mattson_08a.c

Αντί για `#pragma omp critical` μπορεί να χρησιμοποιηθεί το `#pragma omp atomic`

gcc mattson_08a.c -o mattson_08a –fopenmp
./mattson_08a
Threads=1 --> pi = 3.141592654 time elapsed = 0.7320
Threads=2 --> pi = 3.141592654 time elapsed = 0.3810
Threads=3 --> pi = 3.141592654 time elapsed = 0.2780
Threads=4 --> pi = 3.141592654 time elapsed = 0.2470
Threads=5 --> pi = 3.141592654 time elapsed = 0.2260
Threads=6 --> pi = 3.141592654 time elapsed = 0.2150

```
Threads=7 --> pi = 3.141592654 time elapsed = 0.1960
Threads=8 --> pi = 3.141592654 time elapsed = 0.2170
```

4. Παράλληλοι βρόχοι και reduction

```c
#include <stdio.h>
#include <omp.h>
static long num_steps = 100000000;

void main() {
  for (int NUM_THREADS = 1; NUM_THREADS <= 8; NUM_THREADS++) {
    double start = omp_get_wtime(), finish, step, pi = 0.0, x, sum = 0.0;
    step = 1.0 / (double)num_steps;
    omp_set_num_threads(NUM_THREADS);

    #pragma omp parallel for private(x) reduction(+ : sum)
    for (int i = 0; i < num_steps; i++) {
      x = (i + 0.5) * step;
      sum += 4.0 / (1.0 + x * x);
    }
    pi += sum * step;

    finish = omp_get_wtime();
    printf("Threads=%d --> pi = %.9f time elapsed = "
           "%.4f\n",
           NUM_THREADS, pi, finish - start);
  }
}
```

mattson_08b.c

```
gcc mattson_08b.c -o mattson_08a –fopenmp
./mattson_08b
Threads=1 --> pi = 3.141592654 time elapsed = 0.7860
Threads=2 --> pi = 3.141592654 time elapsed = 0.3890
Threads=3 --> pi = 3.141592654 time elapsed = 0.3020
Threads=4 --> pi = 3.141592654 time elapsed = 0.2420
Threads=5 --> pi = 3.141592654 time elapsed = 0.2300
Threads=6 --> pi = 3.141592654 time elapsed = 0.2260
Threads=7 --> pi = 3.141592654 time elapsed = 0.1980
Threads=8 --> pi = 3.141592654 time elapsed = 0.2000
```