Hacettepe Üniversitesi
Bilgisayar Mühendisliği Bölümü
BBM442 Koşut İşlem
Ara Sınav II 21-05-2015

Soru 1. Aşağıda CUDA uyumlu bir grafik işlemcinin çok sayıda çekirdeğini koşut olarak kullanan bir kod kesimi verilmiştir (NVIDIA'nın CUDA teknolojisi). Bu örnekte C dizisinin her elemanı bir izlek (thread) tarafından hesaplanmaktadır. Örnek kod, dizideki eleman sayısı ile izlek sayısının eşit olmasını gerektirmektedir. Dizideki eleman sayısının izlek sayısının 64 katı olması durumu için VecAdd işlevini (kernel) yeniden kodlayınız. Her izleğe ardışık 64 sayı atamakla, ardışık izleklere sayıları birer birer dağıtmak arasında ne fark olurdu?

```
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
int i = threadIdx.x;
C[i] = A[i] + B[i];
}

int main()
{
...
// Kernel invocation with N threads
VecAdd<<<1, N>>>(A, B, C);
...
}
```

Soru 2. Aşağıda matris çarpma algoritmasının OpenMP ile kodlanmış iki sürümü listelenmiştir. Bu iki kodun izlek yapısını çizimle de destekleyerek açıklayınız. İki kod arasında başarıma ilişkin ne farklılıklar vardır?

```
void matrix_multiply (int m, int n, int p, double **A, double **B, double **C)
{
   int i, j, k;
#pragma omp parallel for private(j,k,tmp)
   for (i = 0; i < m; i++)
      for (j = 0; j < n; j++) {
       double tmp = 0.0;
       for (k = 0; k < p; k++)
          tmp += A[i][k] * B[k][j];
       C[i][j] = tmp;
      }
   return;
}
```

```
void matrix_multiply (int m, int n, int p, double **A, double **B, double **C)
{
   int i, j, k;
for (i = 0; i < m; i++)
#pragma omp parallel for private(k,tmp)
      for (j = 0; j < n; j++) {
       double tmp = 0.0;
       for (k = 0; k < p; k++)
          tmp += A[i][k] * B[k][j];
       C[i][j] = tmp;
      }
   return;
}
```

Soru 3. Soru 1 ve 2'yi örnek olarak kullanıp, hızlanma (Speedup) ve etkinlik (Efficiency) terimlerini kısaca açıklayınız. Ölçeklenebilirlik (Scalability) kavramını açıklayınız.