INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
Mid-Autumn Semester 2019-20

Date of Examination: 25/02/2020 ; Session(FN/AN) : **FN** ; Duration **2 hrs**, Marks = **60**
Subject Name : **HIGH PERFORMANCE PARALLEL PROGRAMMING,** Subject No: **CS61064**
Department: **Computer Science and Engineering;** Specific charts, graph paper, log book etc. required: **NO**
Special Instructions (if any): ANSWER ALL questions. In case of reasonable doubt, make assumptions and state them upfront. Marks will be deducted for sketchy proofs and claims without proper reasoning.

Section A (GPGPU) **[Section Full Marks: 35]**

1. Assume a GPU architecture that has 6 streaming multi-processors (SMs) and 160 scalar processors (SPs) per SM. Consider a CUDA kernel with launch parameters <<<(8,8,1),(64,1,1)>>>. Since the number of blocks being scheduled is much greater than the number of SMs, thread blocks will be scheduled in batches. The occupancy of an SM in a batch is equal to the number of active threads executing in that SM divided by the total number of SPs in the SM. What are the number of batches and what is the maximum occupancy of any SM during the kernel lifetime?
[5+5=10 Marks]
   A. 8, 40%
   B. 6, 80%
   C. 6, 40%
   D. 8, 80%

2. Consider a 2D Array **A** of **MxN** integers where each element is of the form **A[i*N+j]=(i*N+j)%8.** There exists a sum kernel called **add(A)** which returns the sum of all elements in **A.** Consider another kernel snippet, the code for which is given below.

```
__global__ void selectiveIncrement(int* A, int M,int N)
{
 int tidx = blockIdx.x*blockDim.x + threadIdx.x;
 int tidy = blockIdx.y*blockDim.y + threadIdx.y;
 int tid = tidx*N + tidy;
 if(A[tid]%2) A[tid]++;
 if(!A[tid]%3) A[tid]-=2;
}
```

[10 Marks]

Given M = 1024 and N=1024, what is the output of add(A) after A is processed by the above kernel? The launch parameters for `selectiveIncrement` are <<<(32,32,1),(32,32,1)>>>.

   A. 1665360
   B. 7864320
   C. 3670016
   D. 3275600

3. Assume there is a 1D input array **A** of **N** (where **N > 8)** integers, where each element is of form A[i]=2*i. Consider the following kernel code snippet executing on a GPU architectu where the warp size is 8.

```
__global__ void divBranch(int* A, int* B, int M, int N, int k){
      int tid = blockIdx.x * blockDim.x + threadIdx.x;
      if(A[tid]%8) {
            B[tid]+=sqrt(A[tid]);
            if(A[tid]%4) {
                  B[tid]+=sqrt(A[tid]); B[tid]+=sqrt(A[tid]);
                  B[tid]+=sqrt(A[tid]); B[tid]+=sqrt(A[tid]);
            }
            else {
                  B[tid]+=sqrt(A[tid]); B[tid]+=sqrt(A[tid]);
            }
      }
}
```

The above code represents a kernel with nested levels of divergence. During the lifetime of a **single warp** (executing 8 threads in lockstep), what are the total number of square root instructions that are executed.

A. 18
B. 16
C. 32
D. 26

[10 Ma

4. Consider that every warp has access to **eight** special function units (SFUs) which can perfo operation. For warp `0', what is the average utilization of these SFUs while executing the code snippet. Pick the valid range.

A. 40 – 50 %
B. 50 – 60%
C. 60 – 70%
D. None of the above

[5 Marks]

Section B (OpenMP) **[Section Full Marks: 25]**

System Spec: 4 processors with max 8 threads
Programming Language: OpenMP C

5. A list of OpenMP C code segments are given below. Check their correctness (syntax and logic). If incorrect then report the mistake and correct it. **[Marks 4+3+3+3=13]**

(a)

```
int arr[10];
for(int i = 0; i < 10; i++)
      arr[i] = i;
#pragma omp parallel for
for (int i = 1; i < 10; i++)
      arr[i] = arr[i - 1];
for(int i = 0; i < 10; i++)
      printf("\narr[%d] = %d", i, arr[i]);
```

(b)

```
int a;
#pragma omp parallel private(a)
{
      a = 0;
      #pragma omp barrier
      #pragma omp sections
      {
            #pragma omp section
            {
                  #pragma omp atomic
                  a+=100;
            }
            #pragma omp section
            {
                  #pragma omp atomic
                  a+=1;
            }
      }
      #pragma omp critical
      {
            printf("a = %d\n",a);
      }
}
```

(c) Make sure that ALWAYS ONLY ONE "Hello World" will be printed in each line.

```
#pragma omp parallel num_threads(2)
      printf("Hello World\n");
```

(d) Assume the existence of one correct fact(int) function.

```
#pragma omp parallel
{
      omp_set_num_threads(2);
      #pragma omp for
      for (int i = 0; i < 10; i++)
            printf("Fact of %d is %d\n",i,fact(i));
}
```

6. The pseudo-code of a serial iterative method is given below. Write down an efficient parallel version (in OpenMP C) of this. **[Marks=12]**

Input: Two strings ***Str1*** and ***Str2*** from the alphabet set {***A***, ***C***, ***T***, G}.
Output: Best alignment score.

```
for i=0 to length(Str1) {
      mat[i,0]=0;
}
for j=0 to length(Str2) {
      mat[0,j]=0;
}
for i=1 to length(Str1) {
      for j=1 to length(Str2) {
      diag=mat[i-1,j-1] + score;    /* score=1 if Str1[i]==Str2[j] else
      score=0 */
      top=mat[i-1,j] + gap;   /* gap = -1 */
      left=mat[i,j-1] + gap;  /* gap = -1 */
      mat[i,j]=maximum(diag,top,left);
      }
}
printf("Max Score= %d\n",mat[length(Str1),length(Str2)]);
```