Problem 1 Assume a GPU architecture that contains 10 SIMD processors. Each SIMD instruction has a width of 32 and each SIMD processor contains 8 lanes for single-precision arithmetic and load/store instructions, meaning that each nondiverged SIMD instruction can produce 32 results every 4 cycles. Assume a kernel that has divergent branches that causes, on average, 80% of threads to be active. Assume that 70% of all SIMD instructions executed are single-precision arithmetic and 20% are load/store. Because not all memory latencies are covered, assume an average SIMD instruction issue rate of 0.85. Assume that the GPU has a clock speed of 2 GHz.

(a) Compute the throughput, in $GFlops/s$, for this kernel on this GPU.

(b) Assume that you have the following choices:

- Increasing the number of single-precision lanes to 12.
- Increasing the number of SIMD processors to 15 (assume this change doesn't affect any other performance metrics and that the code scales to the additional processors) .
- Adding a cache that will effectively reduce memory latency by 40%, which will increase instruction issue rate to 0.95.

What is speedup in throughput for each of these improvements?

Problem 2 In this problem, we will compare the performance of a vector processor with a hybrid system that contains a scalar processor and a GPU-based coprocessor. In the hybrid system, the host processor has superior scalar performance to the GPU, so in this case all scalar code is executed on the host processor while all vector code is executed on the GPU. We will refer to the first system as the vector computer and the second system as the hybrid computer. Assume that your target application contains a vector kernel with an arithmetic intensity of 0.5 FLOPs per DRAM byte accessed; however, the application also has a scalar component that must be performed before and after the kernel in order to prepare the input vectors and output vectors, respectively. For a sample dataset, the scalar portion of the code requires 300 ms of execution time on both the vector processor and the host processor in the hybrid system. The kernel reads input vectors consisting of 300 MB of data and has output data consisting of 100 MB of data. The vector processor has a peak memory bandwidth of 30 GB/s and the GPU has a peak memory bandwidth of 150 GB/s. The hybrid system has an additional overhead that requires all input vectors to be transferred between the host memory and GPU local memory before and after the kernel is invoked. The hybrid system has a direct memory access (DMA) bandwidth of 10 GB/s and an average latency of 10 ms. Assume that both the vector processor and GPU are performance bound by memory bandwidth. Compute the execution time required by both computers for this application.

Problem 3 Assume that we have a function for an application of the form $F(i,p)$, which gives the fraction of time that exactly i processors are usable given that a total of p processors are available. This means that

$$\sum_{i=1}^{p} F(i,p) = 1$$

Assume that when i processors are in use, the applications run i times faster.

(a) Rewrite Amdahl's Law so that it gives the speedup as a function of p for some application.

(b) An application A runs on single processor for a time T seconds. Different portions of its running time can improve if a larger number of processors is used. The figure below provides the details. How much speedup will A achieve when on 8 processors?

(c) Repeat for 32 processors and an infinite number of processors.

Fraction of T	20%	20%	10%	5%	15%	20%	10%
Processors (P)	1	2	4	6	8	16	128

Problem 4 In this exercise, we will examine several loops and analyze their potential for parallelization

a. Does the following loop have a loop-carried dependency?

```
for (i=0;i <100;i++) {
A[i] = B[2*i+4]; /* S1 */
B[4*i+5] = A[i]; /* S2 */
}
```

b. In the following loop, find all the true dependences, output dependences, and antidependences. Eliminate the output dependences and antidependences by renaming.

```
for (i=0;i <100;i++) {
A[i] = A[i] * B[i]; /* S1 */
B[i] = A[i] + c; /* S2 */
A[i] = C[i] * c; /* S3 */
C[i] = D[i] * A[i]; /* S4 */
```

c. Consider the following loop:

```
for (i=0;i <100;i++) {
A[i] = B[i] + C[i]; /* S1 */
B[i+1] = D[i] + E[i]; /* S2 */
C[i+1] = D[i] * E[i] /* S3 */
}
```

Are there dependences between **S1** and **S2**? Is this loop parallel? If not, show how to make it parallel.

Problem 5 Assume a hypothetical GPU with the following characteristics:
- Clock rate 1.5 GHz
- Contains 16 SIMD processors, each containing 16 single-precision floating-point units
- Has 100 GB/s off-chip memory bandwidth

Without considering memory bandwidth, what is the peak single-precision floating-point throughput for this GPU in $GFlops/s$, assuming that all memory latencies can be hidden? Is this throughput sustainable given the memory bandwidth limitation?