Problem 8.1

A CPU designer has to decide on whether or not to add a new microarchitecture enhancement to improve performance (ignoring power costs) of a block (coarse-grain) multithreaded processor. In this processor a thread switch occurs only on a L2 cache miss. The cost of a thread switch is 60 cycles (time before a new thread can start executing). Assume that there are always enough ready threads to switch to on a cache miss. Also, it is given that the current L2 cache hit rate is 50%. The new microarchitectural block is a cache hit/miss predictor. The new predictor predicts whether a memory reference is going to hit or miss in L2 (note not L1) cache. The predictor is used to decide when to switch threads. If the predictor predicts a cache miss thread switching is initiated early. There are four scenarios to consider:

(a) The predictor predicts a L2 cache miss and the true outcome is also a L2 cache miss. In this case thread switching is initiated early and the thread switching cost is reduced to 20 cycles (from 60 cycles in the baseline).

(b) The predictor predicts a L2 cache miss and the true outcome is a L2 cache hit. In this case an unnecessary thread switch has been initiated which increases the thread switching overhead to 120 cycles due to unnecessary pipeline flushes.

(c) The predictor predicts a L2 cache hit and the true outcome is also a L2 cache hit. In this case no thread switching is initiated and there is no gain or loss.

(d) The predictor predicts a L2 cache hit and the true outcome is also a L2 cache miss. This is a case of lost opportunity for an early thread switch and the machine pays the 60 cycle baseline switching penalty.

Given these four scenarios what should be the predictor accuracy before the designer can be certain that this new microarchitectural block leads to a break-even point in performance. If the L2 cache hit rate of the base machine is improved from 50% to 80% how does that impact predictor's accuracy requirements before achieving break-even point in performance?

GPGPU Problem

A CUDA kernel A is launched with 10 blocks of 1024 threads on a GPU with 10 streaming multiprocessors (SMs) taking 27 seconds (s) to run. In this exercise we consider several scenarios in which the kernel is launched on a GPU that has 9 SMs but is otherwise equivalent to the first GPU. In both GPUs the maximum number of threads per SM and the maximum number of threads per block is 1024. Each SM can execute up to two warps in parallel. Assume also that the CUDA kernel does not make use of any `__shared__` memory.

(a) Suppose that kernel A is launched again with 10 blocks but this time on the GPU with 9 SMs. What is the execution time for kernel A on the 9-SM GPU?

(b) How long does it take for kernel A to run on the 9-SM GPU if launched with 9 blocks of 1024 threads, but doing the same amount of work as the 10-block launch.

(c) A different CUDA kernel B is launched with 10 blocks of 32 threads taking 72s on the 10-SM GPU. How long would the 10-block launch take on the 9-SM GPU?

(d) Given the following CUDA kernel code. Each instruction (A to E) costs 10 cycles. The warp size in the 10-SM machine is set to 32. The foo and bar vector sizes are 2048.

```
__global__ void ComputeW (float *foo, float *bar) {
  float v, w;
  int tid = blockIdx.x * blockDim.x + threadIdx.x;
A: v = foo[tid];
B: if(tid < blockDim.x/2)
C:     v++;
   else
D:          v--;
E: w = bar[tid] + v;
}
```

Assume that the programmer set <<<64, 32>>>, which means 64 blocks and the block size is 32, what will be the execution time of the kernel?

If the configuration is changed to <<<32, 64>>>, what will be the new execution time?

ReExam 2018 Problem 4

Consider a future 8-way CMP on which you can power off some cores to allow the rest to operate at a higher frequency. You can use either 1, 2, 4, or 8 cores at the respective frequencies:

- when only one core is running it operates at 0.3ns clock cycle
- when two cores are running they operate at 0.4ns clock cycle
- when 4 cores are running they operate at 0.5ns clock cycle
- when all 8 cores are running they operate at 1ns clock cycle

Consider a partially parallel application which has a serial part that is 1000 Instructions and a parallel part that can be parallelized at will which is 2000 Instructions. Parallelizing however requires you to create threads in the serial part of the application and 100 Instructions are added for every thread you create. The serial and parallel part of the applications all run one instruction per cycle regardless of the frequency.

<u>Your task is to:</u>

(a) Calculate the execution time of the application for the above processor when using 1,2,4,8 cores at their respective frequency without reconfiguring the processor while the application is running. Which configuration is better?

(b) What if you could reconfigure the processor to run the serial part with one core at 0.3 ns clock cycle and then configure it to 2 or 4 or 8 cores for the parallel part. What is the execution time for each case? Always consider that creating each thread takes 100 Instructions that are added to the serial part (and run on the single core at 0.3ns clock cycle).

ReExam 2018 Problem 6 (6p)

A system architect has three choices for an on-chip interconnection network: a uni-directional ring (UR), a bi-directional ring (BR) and an NxN mesh network (NM). In addition, the architect has two choices for providing coherence between L1 caches: snoop-based (SB) or directory-based (DB). There are 16 cores on the chip. All cores have private L1 caches and they share a L2 cache that is divided into 16 banks, with one bank attached to each core. The latency to communicate between two cores connected directly by a link is 1 cycle. For directory-based coherence the access time to the directory is 5 cycles. Assume that there is no contention in any link or in the access to the directory.

(a) Which combination of design choices provides the shortest latency to provide coherence? In this context, latency is considered to be the time it takes to reach all potential destination cores (not including acknowledgments). Consider the worst and average case for a coherence message to reach its destination(s)

(b) Now consider 256 cores and a directory access time of 15 cycles, which combination of design choices provides the shortest latency to provide coherence in this case?

Please organize the results in a table as follows:

Case (a) 16 cores

Latency	**Coherence Type**	**UR**	**BR**	**NM**
Worst Case	SB			
Worst Case	DB			
Average	SB			
Average	DB			

Case (b) 256 cores

Latency	**Coherence Type**	**UR**	**BR**	**NM**
Worst Case	SB			
Worst Case	DB			
Average	SB			
Average	DB			