1. (30%) You are the design manager to implement a multicore system. All the cores in the system share the same memory space. To enhance the interconnection bandwidth, a designer from your team suggests using a NoC (network-on-chip) with a mesh topology to connect all the processors. A NoC adopts a packet-switch scheme, and uses multiple routers to perform the system interconnection. Multiple data transactions can be supported concurrently.
   (a) (10%) You plan to implement on-chip memory for each processing core in order to enhance the data access efficiency. The on-chip memory can be implemented as a cache or scratch-pad style buffer. The cache is managed a hardware controller with a predefined policy, while scratch-pad is managed by users. When would users apply cache? And when would users apply scratch-pad? Please specify an application scenario when this multicore design with on-chip cache can outperform the design with on-chip shared memory. Assume the cache and shared memory have equal capacity and access bandwidth.
   (b) (10%) Assume you choose to implement cache for each processing core. Consider the mesh-style NoC interconnection, can a snoopy cache coherence protocol work correctly on this system? If yes, please use an example to describe how it works. If no, please propose a mechanism to support the snoopy protocol. Please clearly specify your assumptions on each important aspect, such as system architecture, execution behavior, and etc.
   (c) [10%] Assume you successfully implement a snoop protocol to the multicore system. Consider the following code segment in this multicore system, where the initial values are A=B=C=0, Reg1=0, Reg2=0.

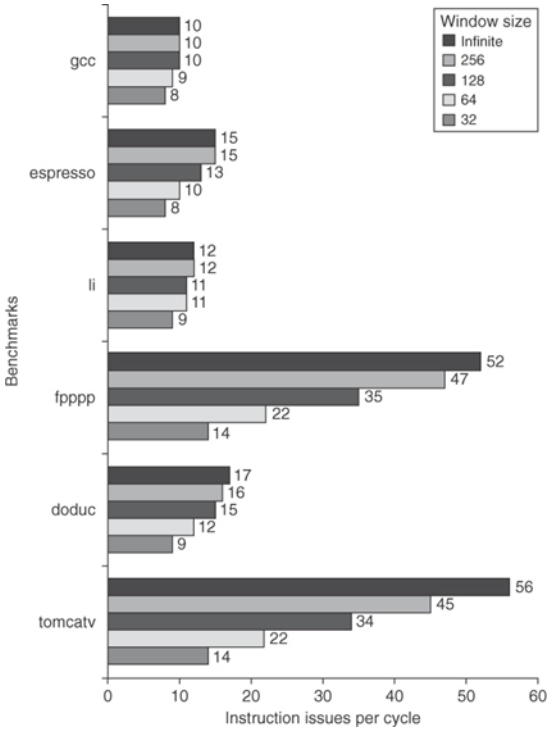
Processor 1	Processor 2	Processor 3	Processor 4

A = 1;
B = 1;
	A = 2;
	C = 1;
		While (B==0) {;}
		While (C==0) {;}
		Reg1=A;
			While (B==0) {;}
			While (C==0) {;}
			Reg2=A;

   If the final value of Reg1 is 1. Assume every function for each processor is working properly, and there is no unexpected error in the system. Please describe at least **two** system scenarios (e.g. cache organization, interconnect between processors, processor architecture, etc) that could make Reg2=2 at the end of the code.

2. (10%) Consider the following three hypothetical, but not atypical, processors, which we run with the SPEC gcc benchmark:
   (a) A simple MIPS two-issue static pipe running at a clock rate 4 GHz and achieving a pipeline CPI of 0.8. This processor has a cache system that yields 0.005 misses per instruction.
   (b) A deeply pipelined version of a two-issue MIPS processor with slightly smaller caches and a 5 GHz clock rate. The pipeline CPI of the processor is 1.0, and the smaller caches yield 0.0055 misses per instruction on average.
   (c) A speculative superscalar with a 64-entry window. It achieves one-half of the ideal issue rate measured for this window size. (Use the data in the following figure.) This processor has the smallest caches, which lead to 0.01 misses per instruction, but it hides 25% of the miss penalty on every miss by dynamic scheduling. This processor has a 2.5 GHz clock.

   Assume that the main memory time (which sets the miss penalty) is 50 ns. Determine the relative performance of these three processors.



3. (10%)
   (a) (5%) Suppose you want to achieve a speedup of 80 with 100 processors. What fraction of the original computation can be sequential?
   (b) (5%) Suppose we have an application running on a 32-core processor, which has a 200 ns time to handle reference to a remote memory. For this application, assume that all the references except those involving communication hit in the local memory hierarchy and the base CPI is 0.5. Once a remote reference request, the processor will be stalled. How much faster is the multicore processor if there is no communication versus if 0.2% of the instructions involve a remote reference? Suppose that the clock rate of each core is 3.3 GHz.

4. (20%) The following code implements the DAXPY operation, *Y = aX + Y*, for a vector length 100. Initially, R1 is set to the base address of array X and R2 is set to the base address of Y:

```
      DADDIU    R4, R1, #800;      R1 = upper bound for X
foo:  L.D       F2, 0(R1);         (F2) = X(i)
      MUL.D     F4, F2, F0;        (F4) = a*X(i)
      L.D       F6, 0(R2);         (F6) = Y(i)
      ADD.D     F6, F4, F6;        (F6) = a*X(i) + Y(i)
      S.D       F6, 0(R2);         Y(i) = a*X(i) + Y(i)
      DADDIU    R1, R1, #8;        increment X index
      DADDIU    R2, R2, #8;        increment Y index
      DSLTU     R3, R1, R4;        test: continue loop?
      BNEZ      R3, foo;           loop if needed
```

The functional unit latencies are shown in the following table. Assume a one cycle delayed branch that resolves in the ID stage. Assume that results are fully bypassed.

Instruction producing result	Instruction using result	Latency in clock cycles
FP multiply	FP ALU op	6
FP add	FP ALU op	4
FP multiply	FP store	5
FP add	FP store	4
Integer operations and all loads	Any	2

(a) (10%) Assume a single-issue pipeline. Unroll the loop as many times as necessary to schedule it without any stalls, collapsing the loop overhead instructions. How many times must the loop be unrolled? Show the instruction schedule. What is the execution time per element of the result?

(b) (10%) Assume a VLIW processor with instructions that contain five operations, as shown in following table.

Memory reference 1	Memory reference 2	FP operation 1	FP operation 2	Integer operation/branch

We will compare two degrees of loop unrolling. First, unroll the loop 6 times to extract ILP and schedule it without any stalls (i.e., completely empty issue cycles), collapsing the loop overhead instructions, and then repeat the process but unroll the loop 10 times. Ignore the branch delay slot. Show the two schedules. What is the execution time per element of the result vector for each schedule? What percent of the operation slots are used in each schedule? How much does the size of the code differ between the two schedules? What is the total register demand for the two schedules?

5. (15%)

(a) (10%) Let's consider a dual-core system connected by a bus-based shared memory. Each processor contains a private direct-mapped L1 cache. To solve the cache coherence problem, a 3-state (un-cached, shared, and exclusive) write-back directory-based coherence protocol is implemented. Suppose that **the block of the cache contains two words**. Assume further that two different words A1 and A2, mapped to the same cache block, have been read already by the processor P1 and P2, respectively, and at the end of the *n*th time epoch they are in shared state. Consider the following sequence of steps:

step	P1	P2
n+1	Write 10 to A2	
n+2		Read A1
n+3		Write 20 to A1
n+4	Read A2	
n+5		Write 30 to A2
n+6		Write 40 to A2
n+7	Write 50 to A1	
n+8		Read A2

Please identify each step as a true sharing miss, a false sharing miss, or a hit.

(b) (5%) Suppose that **the block of the cache is changed to one word only**. Please identify each step in (a) as a true sharing miss, a false sharing miss, or a hit.

6. (15%)

(a) (5%) What is fine-grained multithreading (Fine MT) and what the most advantage and disadvantage of the Fine MT are?

(b) (5%) What is coarse-grained multithreading (Coarse MT) and what the most advantage and disadvantage of Coarse MT are?

(c) (5%) What is simultaneous multithreading (SMT)? Please list at least two design challenges in SMT.