Problem 1:

You are building a system around a processor with inorder execution that runs at 1.1 GHz and has a CPI of 0.7 excluding memory accesses. The only instructions that read or write data from memory are loads (20% of all instructions) and stores (5% of all instructions).

The memory system for this computer is composed of a split L1 cache that imposes no penalty on hits. Both the I-cache and D-cache are direct mapped and hold 32 KB each. The I-cache has a 2% miss rate and 32-byte blocks, and the Dcache is write through with a 5% miss rate and 16-byte blocks. There is a write buffer on the D-cache that eliminates stalls for 95% of all writes.

The 512 KB write-back, unified L2 cache has 64-byte blocks and an access time of 15 ns. It is connected to the L1 cache by a 128-bit data bus that runs at 266 MHz and can transfer one 128-bit word per bus cycle. Of all memory references sent to the L2 cache in this system, 80% are satisfied without going to main memory. Also, 50% of all blocks replaced are dirty. The 128-bit-wide main memory has an access latency of 60 ns, after which any number of bus words may be transferred at the rate of one per cycle on the 128-bit-wide 133 MHz main memory bus.

a) What is the average memory access time for instruction accesses?

b) What is the average memory access time for data reads?

c) What is the average memory access time for data writes?

d) What is the overall CPI, including memory accesses?

e) You are considering replacing the 1.1 GHz CPU with one that runs at 2.1 GHz, but is otherwise identical. How much faster does the system run with a faster processor? Assume the L1 cache still has no hit penalty, and that the speed of the L2 cache, main memory, and buses remains the same in absolute terms (e.g., the L2 cache still has a 15 ns access time and a 266 MHz bus connecting it to the CPU and L1 cache).

f) If you want to make your system run faster, which part of the memory system would you improve? Graph the change in overall system performance holding all parameters fixed except the one that you're improving. Parameters you might consider improving include L2 cache speed, bus speeds, main memory speed, and L1 and L2 hit rates. Based on these graphs, how could you best improve overall system performance with minimal cost?

Problem 2:

As caches increase in size, blocks often increase in size as well.

a) If a large instruction cache has larger data blocks, is there still a need for prefetching? Explain the interaction between prefetching and increased block size in instruction caches.

b) Is there a need for data prefetch instructions when data blocks get larger? Explain.

Problem 3:

Some memory systems handle TLB misses in software (as an exception), while others use hardware for TLB misses.

a) What are the trade-offs between these two methods for handling TLB misses?

b) Will TLB miss handling in software always be slower than TLB miss handling in hardware? Explain.

c) Are there page table structures that would be difficult to handle in hardware, but possible in software? Are there any such structures that would be difficult for software to handle but easy for hardware to manage?

d) Use the data from Figure 5.45 to calculate the penalty to CPI for TLB misses on the following workloads assuming hardware TLB handlers require 10 cycles per miss and software TLB handlers take 30 cycles per miss: (50% gcc, 25% perl, 25% ijpeg), (30% swim, 30% wave5, 20% hydro2d, 10% gcc).

e) Are the TLB miss times in part (d) realistic? Discuss.

f) Why are TLB miss rates for floating-point programs generally higher than those for integer programs?

Program	CPI	Cache misses per 1000 instructions		TLB misses per 1000 instructions
		I-cache	L2 cache	I-TLB
TPC-C-like	2.23	11.15	7.30	1.21
go	0.58	0.53	0.00	0.00
m88ksim	0.38	0.16	0.04	0.01
gcc	0.63	3.43	0.25	0.30
compress	0.70	0.00	0.40	0.00
li	0.49	0.07	0.00	0.01
ijpeg	0.49	0.03	0.02	0.01
perl	0.56	1.66	0.09	0.26
vortex	0.58	1.19	0.63	1.98
Average SPECint95	0.55	0.88	0.18	0.03
tomcatv	0.52	0.01	5.16	0.12
swim	0.40	0.00	5.99	0.10
su2cor	0.59	0.03	1.64	0.11
hydro2d	0.64	0.01	0.46	0.19
mgrid	0.44	0.02	0.05	0.10
applu	0.94	0.01	10.20	0.18
turb3d	0.44	0.01	1.60	0.10
apsi	0.67	0.05	0.01	0.04
fpppp	0.52	0.13	0.00	0.00
wave5	0.74	0.07	1.72	0.89
Average SPECfp95	0.59	0.03	2.68	0.09

Figure 5.45 CPI and misses per 1000 instructions for running a TPC-C-like database workload and the SPEC95 benchmarks (see Chapter 1) on the Alpha 21264 in the Compaq ES40. In addition to the worse miss rates shown here, the TPC-C-like benchmark also has a branch misprediction rate of about 19 per 1000 instructions retired. This rate is 1.7 times worse than the average SPECint95 program and 25 times worse than the average SPECfp95. Since the 21264 uses an out-of-order instruction execution, the statistics are calculated as the number of misses per 1000 instructions successfully committed. Cvetanovic and Kessler [2000] collected these data, but unfortunately did not include miss rates of the L1 data cache or data TLB. Note that their hardware performance monitor could not isolate the benefits of successful hardware prefetching to the instruction cache. Hence, compulsory misses are likely very low.