

Comprehensive Examination Computer Architecture Spring 2001

Problem 1 (10 points)

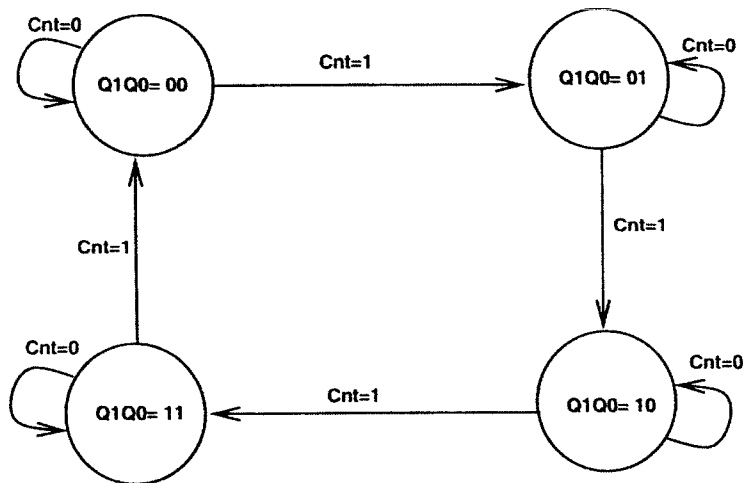
You are given the following function:

	yz	00	01	11	10
wx	00			1	1
01			1	1	
11			1	1	
10		1	1	1	1

Derive the minimal cover implementation in sum-of-products form, identify the static hazards and correct there (if any).

Problem 2 (10 points)

You are given the following state diagram, which is already state encoded, as shown; and only one transition input, *Cnt*, exists. (The output logic is not shown; you do not have to worry about implementing the output logic.) Please implement the *Next-State* logic for this state diagram, using *D* flip-flops.



Problem 3 (15 points):

A particular pipelined, in-order, scalar processor has a 9-stage pipeline, with branches resolved (that is, both branch direction and branch target are known) in the Oh stage, and instruction fetch beginning in the 15' stage. It has separate instruction and data caches, but in each case the cost of a miss is 25 cycles (stores typically do not stall the processor), but the pipeline stalls on the first *use* of the missed data. For program P,

- the instruction cache miss rate is 2%
- the data cache miss rate is 4%
- the branch prediction accuracy is 90%
- loads are 20% of all instructions
- conditional branches are 30% of all instructions.
- the typical distance between a load and its first use is 5 instructions.
(state any other assumptions you make)

Assuming an ideal CPI (perfect caches, perfect branch prediction) of 1.0, what is the actual CPI?

Would you get better speedup from an optimization that halved the data cache miss rate, or one that halved the branch misprediction rate? Give numbers.

How would you expect your answer to change if it were an out-of-order processor? (explain without numbers)?

Problem 4 (20 points): Cache memories.

- (a) In a few sentences, describe the significance of the following implementation policies: *multilevel inclusion*, *non-blocking* (also known as *lockup free*), and *sub-block placement*.
- (b) You are given the following loop nest that repeatedly sweeps over a one-dimensional array of N integer words.

The inner loop accesses every k^{th} element of the array. k is referred to as the *stride*.

```
int x[N] ;
int k = ...;           // input by the user

while ( do for a very long time ){
    for ( int i =0; i<N; i+=k )
        x[i] = i;
}
```

- a. Assume that you have a direct-mapped cache with a total capacity of N words and a line size 8 words. List all values of k for which the loop nest exhibits cache re-use due to *spatial locality*.
- b. We now decrease the capacity of our cache by one-half, keeping the organization and line size of the cache the same as in (a). What is the *miss rate* when $k=4$?
- c. Finally, we construct a fully associative cache. We keep the same line size (8 words) and the same total size of the cache ($N/2$ words) as in (b). For which values of k will the loop nest exhibit cache re-use due to *temporal locality*? Express the values as a range of integers.

Problem 5 (15 points):

Consider the following code (in this code, the destination operand is the first operand listed):

```
loop:
  LW    r3, 1000(r2)
  SUB   r1, r3, r4
  ADDI  r5, r3, #5
  LW    r8, 0(r1)
  SUBI  r2, r2, #4
  ADD   r1, r3, r3
  SW    0(r8), r1
  bnez  r2, loop
```

For the following questions, please state all assumptions clearly.

- (a) Assuming it was executed on a scalar (ie, 1 instruction per pipe stage per cycle) MIPSlike 5-stage pipeline (IF ID EX MEM WB), what would be the steady-state CPI of this loop (assuming it was executed many times, with perfect branch prediction)? Also assume the architecture does not have a branch delay slot.
- (b) What is the maximum speedup you could expect from a superscalar processor with the same pipeline (assuming unlimited resources)?
- (c) What is the maximum speedup you could expect from a superscalar out-of-order processor with register renaming (assuming unlimited resources)?

Problem 6 (10 points):

You are designing a special-purpose processor for a particular application, and thus have more freedom than is available in a general-purpose design. Give me two reasons (scenarios?) why you might not want to include hardware support for virtual memory (e.g. TLBs) and perhaps not support VM at all. At least one should involve performance.

Give me two reasons (scenarios?) why you would want to include virtual memory. At least one should involve performance.