CS 596 Quiz#2 March 20, 2000 NAME : solution

There are 10 questions. All questions carry 10 points. Some formulas are given on the last page.

1. Define miss rate and miss penalty.

   Miss rate is the fraction of cache access that results in a miss.

   Miss penalty is the number of memory stall cycles needed for a cache miss.

2. What are the three different types of cache mapping (depending on where a memory block can be placed in cache) called?

   Suppose we have a type of cache where a memory block can be mapped to one particular location in the cache. There are 32 memory blocks numbered 0 through 31 in the memory. The cache can hold total of 4 blocks in locations numbered 0 through 3. Where will memory block number 10 go in the cache in this case?

Direct mapped, N-way set associative, and fully associative.

since (10 MOD 4) = 2, memory block #10 will go to cache location # 2.

3. What are the two principles based on which caches are designed?

   We know that in C a matrix is stored row wise, whereas in Fortran a matrix is stored column wise. Loop interchange is sometimes done to make sure that matrices are accessed in correct order in either C or Fortran. Which principle of cache design is utilized when loop interchange is done?

   Which principle of cache design is utilized in loop fusion?

Caches are designed based on spatial locality and temporal locality principles.
Loop interchange uses spatial locality principle since data made to access the way it is brought into cache.
Loop fusion uses temporal locality principle since once data is brought into cache it is used multiple times.

4. Explain what are Random Replacement and Least Recently Used strategies. Where are they used ?

Random replacement : To spread allocation uniformly candidate blocks are randomly selected from cache blocks
LRU : To reduce the chance of throwing out information that will be needed soon, accesses to blocks are recorded. The block replaced is the one that has been unused for the longest time.

The above two are used in fully associative or set associative caches.

5. Suppose cache miss penalty is 60 clock cycles. All instructions normally take 3.0 clock cycles i.e. ignoring memory stalls. Assume miss rate is 3% and there is average of 1.25 memory references per instruction. What are the impact on performances when
   (i) there is no cache miss i.e. perfect cache

(ii) cache miss is taken into account
(iii) there is no cache at all

(i) Cache hit time is included in the CPI of CPU.
CPU execution time =
IC * clock cycle time * (CPI of CPU + memory reference per instruction * miss rate * miss penalty)
Since we have perfect cache there is not memory stall cycles, hence
CPU execution time = IC * clock cycle time * (3.0)

(ii) CPU execution time = IC * clock cycle time * (3.0 + 1.25*0.03*60) = IC*clock cycle time*5.25

(iii) Since there is no cache at all, we have miss rate of 100%, hence
CPU execution time = IC * clock cycle time * (3.0 + 1.25 * 60) = IC * clock cycle time * 78

6. We have two different cache organizations as explained below. First calculate the average memory access time for each and based on the results conclude which is better. Next calculate the CPU performance for each and based on the results conclude which is better. Finally comment on which is a better cache design overall based on the above two calculations.
   Assume for both caches: the CPI for the perfect cache is 2.0; there are 1.3 memory reference per instruction.
   Cache1 : miss penalty is 70 ns, hit time is 1 clock cycle; miss rate is 1.4%, clock cycle time is 2 ns
   Cache2 : miss penalty is 70 ns, hit time is 1 clock cycle; miss rate is 1.0%; clock cycle time is 1.10 times the clock cycle of cache1

Average memory access time for cache1 = 2 ns + 0.014 * 70 ns = 2.98 ns (since hit time is 1 clock cycle)
Clock cycle time of cache2 machine is 1.10 time that of cache1 machine i.e. clock cycle time of cache2 machine = 2 * 1.10 = 2.2 ns
Average memory access time for cache2 = 2*1.10 + 0.01 * 70 = 2.9 ns (since hit time is 1 clock cycle)
Hence comparing the above two, cache2 is better than cache1.

CPU execution time of cache1 = IC * (2.0*clock cycle time + 1.3*0.014*70)
= IC * (4.0 + 1.3*0.014*70) ns = 5.27 *IC ns
CPU execution time of cache2 = IC * (2 * clock cycle time + 1.3 * 0.01 * 70)
= IC* (2.2 + 1.3*0.01*70) = 5.31 IC
Comparing the above two timing results, cache1 machine is better than cache2 machine.
Since CPU execution time is the best measure of machine performance, we conclude that overall cache1 machine is a better machine.

7. What are the three different kind of hazards that can stall a pipelined architecture?

Structural hazard, data hazard, and control hazard.

8. Suppose 30% of the instructions are loads, and half the time the instruction following a load instruction depends on the result of the load. If this hazard creates a single-cycle delay, how much faster is the ideal pipelined machine (with CPI of 1) that does not delay the pipeline than the real pipeline? Ignore any stalls other than pipeline stalls.

The ideal machine will be faster by the ratio of CPI. Say we have 100 loads and for ideal machine that should have taken 100 cycles. But 50 of the loads create hazards and an additional single cycle delay

for each of those 50 loads. So total number of cycles for 100 loads is 150 clock cycles. So CPI for loads is 1.5.
Since 30% of instructions are loads, we have average CPI for the machine with hazard = (0.3*1.5 + 0.7*1.) = 1.15
So the ideal machine is 1.15 times faster.

9. Answer either C or Fortran case.

C case : Write a 4 times unrolled version of the following loop across the index of the inner loop. Is there any data dependency problem in the unrolled loop?

```
for (j = 1 ; j < 512 ; j = j + 1 )
        for  ( k = 0 ; k < 512 ; k = k + 1 )
                x[k][j] = 2 * x[k][j-1] ;
```

Explain if there is any conflict between ILP (obtained by unrolling the loop) and cache hit in the unrolled loop.

Fortran case: Write a 4 times unrolled version of the following loop across the index of the outer loop. Is there any data dependency problem in the unrolled loop ?

```
do k = 1, 512
        do j = 2, 512
                x(k,j) = 2* x(k,j-1)
```

Explain if there is any conflict between ILP (obtained by unrolling loop) and cache hit in the unrolled loop.

C case

```
for (j = 1 ; j < 512 ; j = j + 1 )
        for  ( k = 0 ; k < 512 ; k = k + 4 ) {
                x[k][j] = 2 * x[k][j-1] ;
                x[k+1][j] = 2 * x[k+1][j-1] ;
                x[k+2][j] = 2 * x[k+2][j-1] ;
                x[k+3][j] = 2 * x[k+3][j-1] ; }
```

There is not data dependency in the above unrolled loop since nothing on the RHS is computed in the above in the LHS.
There is cache conflict since data is stored row wise and data is accessed column wise.

The fortran loop can be unrolled similarly over k. There is no data dependency in fortran either. And similar to C, there is cache conflict since in fortran data is stored column wise and data is accessed row wise.

10. Unroll the following loop as many times as possible without causing data dependency (answer either Fortran or C case).

```
C code : for (j = 6,  j <= 100, j = j +1) {
              y[j] = y[j -5] + y[j] ;
         }

Fortran code : do j = 6, 100
                    y(j) = y(j-5) + y(j)
               end do
```

The loop can only be unrolled 5 times before date dependency happens. This is shown below :

```
for (j = 6; j <= 100, j = j + 5) {

   y[j] = y[j-5] + y[j] ;
   y[j+1] = y[j-4] + y[j+1] ;
   y[j+2] = y[j-3] + y[j+2] ;
   y[j+3] = y[j-2] + y[j+3] ;
   y[j+4] = y[j-1] + y[j+4] ; }
```

If we unrolled one more time then we will have,

```
   y[j+5] = y[j] + y[j+5] ;
```

So we need y[j] on the RHS which is calculated in the same loop on the LHS ; hence we have data dependency.

Formulas you may need :

average memory access time = Hit time + Miss rate * Miss penalty

CPU time =

IC * ($CPI_{execution}$ + (memory access per instruction)* (miss rate) * (miss penalty))*clock cycle time