Midterm Exam

This is an "open book" exam (referencing notes + slides is allowed).

You are to work individually -- no cooperation is allowed. All work must be your own.

Partial credit may be awarded. Always attempt a problem as best you can and write something down. Show all your thoughts/work.

Good Luck!

Part 1: Loop Transformations (20 points)

For the following loop code, perform loop unrolling with an unrolling factor of 4, and show the correct transformed code.

```
//N is a user input
int N = atoi(argv[1]);
int p = 1;
int A[N], B[N];
for (int i=0; i<N; i++) {
    p = p * (A[i] * B[i]);
}
```

```
int N = atoi(argv[1]);
int p = 1;
int A[N], B[N];

int remainder = N % 4;
for (int i=0; i<remainder; i++) {
    p = p * (A[i] * B[i]);
}
for (int i=remainder; i < N; i+=3) {
    p = p * (A[i] * B[i]);
    p = p * (A[i+1] * B[i+1]);
    p = p * (A[i+2] * B[i+2]);
    p = p * (A[i+3] * B[i+3]);
}
```

In the previous question, which other loop transformation technique did you need to use to correctly implement loop unrolling?

Loop Peeling

Identify whether the Loop Fission applied to the following code is a safe or unsafe transformation. If it is unsafe, describe or show why it is unsafe.

```
//Original Code
a[0]=5; a[1]=7;
for (int i=2; i<=16; i++) {
    S1: a[i] = b[i] + d[i];
    S2: c[i] = a[i-2] * 8;
}


//Code after Loop Fission
a[0]=5; a[1]=7;
for (int i=2; i<=16; i++) {
    S1: a[i] = b[i] + d[i];
}
for (int i=2; i<=16; i++) {
    S2: c[i] = a[i-2] * 8;
}
```

Safe

Identify whether the Loop Fusion applied to the following code is a safe or unsafe transformation. If it is unsafe, describe or show why it is unsafe.

```
//Original Code
for (int i=1; i<N; i++) {
    S1: a[i] = b[i-1];
}
for (int i=1; i<N; i++) {
    S2: b[i] = c[i] + d[i];
}


// Code after Loop Fusion
for (int i=1; i<N; i++) {
    S1: a[i] = b[i-1];
    S2: b[i] = c[i] + d[i];
}
```

Unsafe. In the original code, S1 reads all elements of b[] before S2 writes new values into b[]. In the fused code, new values are written into b[] by S2 before they are read (in the next loop iteration) by S1.

Part 2: Profiling & Amdahl's Law (15 points)

Answer true/false for the following:

Analyzing the profiling output of a flat profile will allow for understanding the amount of execution time spent in a function named B( ) when called by a function named C( ) as opposed to when function B( ) is called by functions X( ) or Y( ).

- ○ True
- ○ **False**

In order to understand whether the L2 cache miss rate was high for an application, which of the following would be more appropriate to collect?

- ○ A. Assembly code dumps
- ○ B. Performance profile data
- ○ C. Execution time measurements
- ○ **D. Performance counter data**

Assume we have a new application that we are developing, and we want to improve its performance. To help us determine where to focus our optimization efforts, we have exercised a profiling tool on our application, and found that one function accounts for 40% of the total execution time. After examination of this function, we realize that there is an amazing code optimization that we can make that effectively makes this function take 0 time (all its execution time is hidden)! If we apply this optimization, what is the speedup that we can expect compared to the original code? If our original goal had been to speed up this application by 4X, would our new code reach this goal?

The speedup would be 1.67x (speedup = 1 / 0.6). We would not reach the 4x goal.

Part 3: Data Dependence Analysis (22 points)

For the code shown below, list all data dependencies. Clearly indicate the type of the dependence (true, anti, output) as well as the statements involved in the dependence and their loop index relationship. For example, S5[i,j] =>T S7[i+1,j+1].

Also, number each dependence (1 through X). You will use the number to refer to each dependence in the next question.

Hint: It may be helpful to sketch out the ITG on scratch paper as part of identifying the dependencies.

```
...
for (i=N-2; i>0; i--) { //Note the loop direction
  for (j=1; j<N-1; j++) {
    S1: c[i][j] = (4 + a[i][j]) / 2;
    S2: a[i][j] = a[i+1][j] + b[i][j+1];
    S3: b[i][j] = 2 * b[i+2][j+2]
  }
}
...
```

1. S1[i][j] =>A S2[i][j]
2. S2[i][j] =>T S2[i-1][j]
3. S2[i][j] =>A S3[i][j+1]
4. S3[i][j] =>T S3[i-2][j-2]

For each data dependence listed in the answer to the previous question, indicate whether it is loop-carried or loop-independent. For each dependence that is loop-carried, indicate whether it is loop-carried on the 'for i' loop or the 'for j' loop.

1. Loop independent
2. Loop carried, ‘for i’
3. Loop carried, ‘for j’
4. Loop carried, ‘for i’ and ‘for j’

Imagine we had an in-order processor with a maximum performance of 1 instruction per cycle (i.e. it is not a superscalar processor). Also, the processor can magically execute all instructions in a single cycle (even loads and stores). Answer true/false to the following statement:
Data dependencies between instructions in a program could hurt the performance of this program when running on this processor.

Concisely explain your rationale.

○ True

○ **False**

The processor cannot access more than one instruction each cycle. In addition, every instruction takes 1 cycle to execute. Therefore, its results are ready for a dependent instruction to execute on the very next cycle. Thus, a data dependence could not hurt performance of the code executing on this processor.

Part 4: Cache Operation & Performance (22 points)

Assume that we are running code on a processor that has only a very small, fully associative cache with 4 cache blocks of 64 bytes each. The cache uses LRU (least recently used) replacement policy. This cache is backed up by physical memory (i.e. there is only this single level of cache between the processor and memory).

A[27], hit

Continuing with the previous question, assume the cache has an access latency of 2 processor cycles and a miss penalty of 100 cycles (the time to access memory, which is in addition to the 2 cycles to access the cache and detect a miss).

Calculate the cache miss rate and Average Access Time for the sequence of access hits / misses made by this code that you determined in the previous question. Note: you can just show the equation – you do not need to simplify down to a single number.

Miss rate = 4/10 = 0.4
AAT = 2 + 100 * 0.4 = 42 cycles

Assume we have a cache with the following design:

- 64B cache blocks
- 16 sets
- 4-way set associative

If we access the cache with the following address (in hex), which cache set will be accessed?

0xA374DB2

Offset = 6 bits (for 64B cache blocks)
Set index = 4 bits (for 16 sets)
Cache set = (0xA374DB2 >> 6) & 0xF = **6**

Part 5: Miscellaneous (21 points)

In the latency measurement program that we constructed in class, we used a “stride”

We learned that loop unrolling reduces the number of instructions required to execute a loop (i.e. we have fewer loop management instructions). Given this benefit of loop unrolling, describe a reason why we (and the compiler) do not fully unroll all loops in our program.

Loop unrolling increases the size of the code in the compiled program. The primary negative impact of the larger code size would be a worse instruction cache hit rate.

Consider the use of Vector instructions (or SIMD). Which of the following types of parallelism is this expressing?

- A. Pipelined parallelism
- B. Instruction level parallelism (ILP)
- C. Thread-level parallelism
- **D. Data parallelism**

Assume a CPU has one processor core with an L1 and L2 cache. The caches use 128-byte cache blocks. Also assume that the L2 cache can support up to 10 outstanding cache misses at a time (i.e. up to 10 in-flight requests to memory). If the average latency of each miss is 200 cycles, what is the maximum memory bandwidth the processor can achieve (you may assume the memory system itself has unlimited bandwidth).

Max B/W = (10 * 128B) / 200 cycles = 1280/200 = 6.4 bytes per cycle

Consider a matrix multiplication program, like what we looked at in assignment #2. Would you expect the performance of matrix multiplication to be more sensitive to the latency or the bandwidth of the cache and memory subsystem of the CPU? Briefly and clearly explain your reason(s) why.

Bandwidth. The program consists of independent memory accesses. Since they are independent, the processor can initiate multiple in-flight accesses and thus exposes the bandwidth of the cache/memory subsystem rather than the access latency.