Computer Architecture Qual., Fall 2012

- This is a 2-hour, closed-book, closed-note test. Calculator allowed.
- This test has 4 questions for a total of 100 points.

1. [20 points] **Amdahl's law**

   The marketing department at your company has promised your customers that the next software release will show a 2× performance improvement. You have been assigned the task of delivering on that promise. You have determined that only 80% of the system can be improved. How much (i.e., what value of a factor) would you need to improve this part to meet the overall performance target?

2. [30 points] **Branch prediction**

   Consider the following C code with nested loops.

```
for (i=0; i<20; i++)
    for (j=0; j<3; j++)
        sum += a[i][j];
```

   When this C code is compiled into an assembly code, two branch instructions are needed for executing inner loop and outer loop, respectively. Consider only the inner loop branch for different branch predictors in the questions below. We assume that each predictor is initialized to *not-taken* state (or strongest not-taken state if multiple not-taken states exist).

   (a) When the 1-bit bimodal predictor is used, compute the mis-prediction rate.

   (b) When the 2-bit bimodal predictor is used, compute the mis-prediction rate.

   (c) When the (2, 1) correlating predictor is used, compute the mis-prediction rate. Note that an (m, n) correlating branch predictor uses the behavior of the most recent m executed branches to choose from 2^m predictors, each of which is an n-bit bimodal predictor.

3. [20 points] **Cache miss rate analysis**

Consider a direct mapped cache of size 64KB with block size of 16B. Furthermore, the cache is write-back and write-allocate. You will calculate the miss rate using this cache for three different functions below. The first function is implementation of a copy of the matrix (`copy_matrix`). The second and third functions are implementations of a horizontal flip and copy of the matrix (`copy_n_flip_matrix1` and `copy_n_flip_matrix2`).

```
void copy_matrix(int dest[ROWS][COLS], int src[ROWS][COLS]) {
    register int i, j;
    for (i=0; i<ROWS; i++) {
        for (j=0; j<COLS; j++) {
            dest[i][j] = src[i][j];
        }
    }
}

void copy_n_flip_matrix1(int dest[ROWS][COLS], int src[ROWS][COLS]) {
    register int i, j;
    for (i=0; i<ROWS; i++) {
        for (j=0; j<COLS; j++) {
            dest[i][COLS - 1 - j] = src[i][j];
        }
    }
}

void copy_n_flip_matrix2(int dest[ROWS][COLS], int src[ROWS][COLS]) {
    register int i, j;
    for (j=0; j<COLS; j++) {
        for (i=0; i<ROWS; i++) {
            dest[i][COLS - 1 - j] = src[i][j];
        }
    }
}
```

Use the following assumptions.

- `ROWS = 128` and `COLS = 128`.
- `sizeof(int) = 4`.
- The `src` matrix starts at address 0 and the `dest` matrix immediately follows it. Elements in an array are allocated in row-major order.
- The cache starts empty.
- Local variables and computations take place completely within the registers and do not spill onto the stack.

Compute the cache miss rate of each function.

4. [30 points] **Cache coherence**

Consider a 2-processor SMP using the write-invalidate MSI snoopy coherence protocol. Two processors, `P1` and `P2`, have one-level private caches with write-back policy.

We assume the following:

- Read and write hits generate no stall cycles. (i.e., Both hits take 1 cycle in processor.)
- Read and write misses generate 100 stall cycles and are satisfied by only memory.
- Write hits that generate an invalidate incur 10 stall cycles.
- A writeback of a block incurs additional 5 stall cycles.

The cache in each processor has 1 cache block, which size is 4 words. The caches are initially empty. The memory accesses occur strictly sequentially in textual order below. Two words in the memory accesses, `A1` and `A2`, are in the same cache block.

```
 1) P1 read A1
 2) P1 read A2
 3) P1 write A1
 4) P1 read A1
 5) P2 read A1
 6) P1 write A1
 7) P2 read A2
 8) P1 write A1
 9) P2 write A2
10) P1 read A2
```

Answer the questions below.

(a) Show the state transition for the cache block in the caches of `P1` and `P2` after each access.

(b) Determine the total time needed to execute the given memory access sequence.