**Assignment 1**

**PROBLEM 1.** [15 points] The instruction-type breakdown of three programs $P_1$, $P_2$ and $P_3$ are given below:

<table>
<thead>
<tr>
<th>Number of instructions</th>
<th>ALU</th>
<th>Load</th>
<th>Store</th>
<th>Branch</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>700</td>
<td>500</td>
<td>200</td>
<td>50</td>
<td>1450</td>
</tr>
<tr>
<td>$P_2$</td>
<td>700</td>
<td>1330</td>
<td>200</td>
<td>50</td>
<td>2280</td>
</tr>
<tr>
<td>$P_3$</td>
<td>700</td>
<td>500</td>
<td>2275</td>
<td>50</td>
<td>3525</td>
</tr>
</tbody>
</table>

1.1 [5 points] We are running $P_1$, $P_2$ and $P_3$ on a processor for which any ALU instruction uses 1 clock-cycle and any Branch instruction uses 3 clock-cycles. Suppose that the numbers of clock-cycles for Load and Store instructions are unknown and, that we want to determine them experimentally. Assume that $P_2$ and $P_3$ run respectively 2 and 3 times slower than $P_1$. Deduce the numbers of clock-cycles for a Load instructions and a Store instruction (Ignoring memory-stall cycles).

1.2 [5 points] Compute the execution time of Program $P_1$ for a clock rate of 3 GHz. (Still ignoring memory-stall cycles).

1.3 [5 points] Compute the execution time of Program $P_1$ on the same processor, now taking into accounts memory-stall cycles. To do so, we assume that

- the processor has a 100 cycle miss penalty,
- Program $P_1$ has 2% instruction-cache miss rate, and 3 %data-cache miss rate,

**PROBLEM 2.** [15 points] Let us consider a computer with a L1, L2 and L3 cache memory hierarchy and with the following characteristics:

- Processor Clock Frequency = 1 GHz,
- Hit Time L1 = 1 clock cycle,
- Hit Time L2 = 4 clock cycles,
- Hit Time L3 = 8 clock cycles,
- Miss Penalty L3 = 15 clock cycles.

Assume that, on this computer, we are running a program $P$ for which Hit Rate L1 = 95%, Hit Rate L2 = 92% and Hit Rate L3 = 90%.

2.1 [5 points] How much is the Global Miss Rate for the three levels of cache (that is the product of the miss rates at levels L1, L2 and L3)?

2.2 [5 points] How much is the AMAT?

2.3 [5 points] Given a MAPI (Memory Access Per Instruction, that is, the percentage of Load/Store instructions) of 78% and a CPIideal of 3, how much is the CPIstall when the hit rate in L3 becomes 100%
PROBLEM 3. [30 points] The following four questions are using this simple cache memory illustrated below.

We specify its key features, which all similar to cache features seen in class:
- byte addressable memory,
- direct-mapped cache,
- the cache has a capacity of 32Kbyte and each block holds 64 bytes, thus the cache can store up to 512 blocks,
- assuming that each variable of type int in a C program uses 4 bytes, this cache can store up to 2^{9} \times 2^{4} = 2^{13} int values,
- byte addresses map into cache as follows:
  - bottom 6 bits are used as offset in a cache block,
  - next 9 bits determine the cache block,
  - top 17 bits give the tag.

For each of the following four questions, we estimate the cache miss rate of a fragment of a program written in the programming language C. Each program fragment reads and/or writes arrays. We assume that:
- the only data being stored in cache is coefficients from those arrays,
- the cache is initially empty when the execution of the program fragment starts,
- the arrays are laid out sequentially in memory, that is, two consecutive array elements are stored in consecutive memory words,
- the arrays are aligned in memory, that is, the first element of each array is at the beginning of a block.

3.1 [7 points]

```c
#define S (1<<20)
int A[S];
// Thus size of A is 2^20
for (i = 0; i < S; i++) {
}
```

What are the total numbers of cache misses and cache hits? What kind of locality does it have, if any? What kind of cache misses?

3.2 [7 points]
```c
#define S (1<<20)
#define T (1<<12)
#define R (1<<8)
int A[S];
// Thus size of A is 2^(20)
for (j = 0; j < R; j++) {
    for (i = 0; i < T; i++) {
    }
}

What are the total numbers of cache misses and cache hits? What kind of locality does it have, if any? What kind of cache misses?

3.3 [8 points]

#define S (1<<20)
#define T (1<<14)
#define R (1<<6)
int A[S];
// Thus size of A is 2^(20)
for (j = 0; j < R; j++) {
    for (i = 0; i < T; i++) {
    }
}

What are the total numbers of cache misses and cache hits? What kind of locality does it have, if any? What kind of cache misses?

3.4 [8 points]

#define S (1<<19)
int A[S];
int B[S];
int tmp1;
int tmp2;
// Thus, in the main memory, the cache lines of
// B are just after all the cache lines of A
for (i = 0; i < S; i++) {
    tmp1 = A[i];
    tmp2 = B[i];
    A[i] = tmp2;
    B[i] = tmp1;
}
```
What are the total numbers of cache misses and cache hits? What kind of locality does it have, if any? What kind of cache misses?

**PROBLEM 4.** [40 points] Download the archive `perf_examples.tgz` from the course website. Compile the programs `mm_naive.c` and `mm_tiled.c` using the Makefile.

4.1 [10 points] Tune the program `mm_tiled.c` by varying the values of the parameters `BLOCK_X`, `BLOCK_Y` and `BLOCK_Z` defined in the code at the beginning of the source file. Note that those parameters must be powers of 2, no greater than 1024. What are the best values of `BLOCK_X`, `BLOCK_Y` and `BLOCK_Z` on your computer?

4.2 [10 points] What do the two programs do? Which approach does each program take?

4.3 [5 points] Show the CPU info of the machine that you are using for your measurements. Under Linux, you will find this information in `/proc/cpuinfo`.

4.4 [15 points] Choose proper performance metrics and use `perf` to measure them for both programs. Consider to vary the values of `x`, `y`, `y` (keeping `x = y = z`) to successive powers of two: 512, 2014, 2048, 4096, etc. Which program is faster? Explain briefly why one is faster than the other.

**PROBLEM 5.** [20 points]

Below is a list of 32-bit word addresses in the main memory:

<table>
<thead>
<tr>
<th>Address</th>
<th>Tag</th>
<th>Index</th>
<th>Offset</th>
<th>Hit/Miss</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0000</td>
<td>011</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>194</td>
<td>0100</td>
<td>001</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>0001</td>
<td>111</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000</td>
<td>011</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>196</td>
<td>0100</td>
<td>010</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>0010</td>
<td>111</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>197</td>
<td>0100</td>
<td>010</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0001</td>
<td>011</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000</td>
<td>011</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>0011</td>
<td>011</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>195</td>
<td>0100</td>
<td>001</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>265</td>
<td>1000</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

As in Problem , and as in the lectures, the memory is byte addressable.

5.1 [10 points] Consider a direct-mapped cache with 2-word cache lines. Indicate whether each reference below is either a hit or a miss, assuming the cache is initially empty and is large enough to accommodate all the cache lines referred below.
5.2 [10 points] Consider now a 2-way set associative cache with 2-word cache lines. Indicate whether each reference below is either a hit or a miss, assuming the cache is initially empty and is large enough to accommodate all the cache lines referred below. Use LRU (least recently used) replacement if necessary.

<table>
<thead>
<tr>
<th>Address</th>
<th>Tag</th>
<th>Index</th>
<th>Offset</th>
<th>Hit/Miss</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0000</td>
<td>011</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>194</td>
<td>0110</td>
<td>001</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>0001</td>
<td>111</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000</td>
<td>011</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>196</td>
<td>0110</td>
<td>010</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>0011</td>
<td>111</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>197</td>
<td>0110</td>
<td>010</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0001</td>
<td>011</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000</td>
<td>011</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>0011</td>
<td>011</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>195</td>
<td>0110</td>
<td>001</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>265</td>
<td>1000</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>