Flynn Taxonomy

1. Explain SISD and give an example.

   Single Instruction Single Data; each instruction is executed in order, acting on a single stream of data. For example, traditional computer programs.

2. Explain SIMD and give an example.

   Single Instruction Multiple Data; each instruction is executed in order, acting on multiple streams of data. For example, the SSE Intrinsics.

3. Explain MISD and give an example.

   Multiple Instruction Single Data; multiple instructions are executed simultaneously, acting on a single stream of data. There are no good modern examples.

4. Explain MIMD and give an example.

   Multiple Instruction Multiple Data; multiple instructions are executed simultaneously, acting on multiple streams of data. For example, map reduce or multithreaded programs.

Average Memory Access Time

AMAT stands for Average Memory Access Time. It refers to the time necessary to perform a memory access on average. It does NOT refer to the time necessary to execute an instruction which accesses memory. The AMAT of a simple system with only a single level of cache may be calculated as:

$$\text{AMAT} = (\text{hit time}) + (\text{miss rate}) * (\text{miss penalty})$$

This formula can be extended to more complicated memory hierarchies by replacing miss penalty with the AMAT for the next level in the memory hierarchy. Specifically for the i^{th} level of memory:

$$\text{AMAT}_i = (\text{hit time})_i + (\text{miss rate})_i * (\text{miss penalty})_{i+1}$$

1. Calculate the AMAT for a system with the following properties:

   - L1$ hits in 1 cycle with local hit rate 50%
   - L2$ hits in 10 cycles with local hit rate 75%
   - L3$ hits in 100 cycles with local hit rate 90%
   - Main memory always hits in 1000 cycles

   $$\text{AMAT} = 1 + (1 - 0.5)(10 + (1 - 0.75)(100 + (1 - 0.9)(1000))) = 31$$

2. Calculate the AMAT for a system with the following properties:

   - L1$ hits in 1 cycle with global miss rate 50%
   - L2$ hits in 10 cycles with global miss rate 25%
   - L3$ hits in 100 cycles with global miss rate 10%
   - Main memory always hits in 1000 cycles

$$\text{AMAT} = 1 + 0.5(10) + 0.25(100) + 0.1(1000) = 131$$

Cache Performance

1. Suppose our processor has a separate L1 instruction cache and data cache. Our $\text{CPI}_{\text{ideal}}$ is 3 clock cycles, whereas memory access takes 125 cycles. Our I$ miss rate is 4%, while our D$ miss rate is 8%. Unfortunately, 60% of our instructions happen to be loads or stores. What is our processor's $\text{CPI}_{\text{stall}}$?

$$\text{CPI}_{\text{stall}} = \text{CPI}_{\text{ideal}} + (\text{L1 instruction miss cycles}) + (\text{L1 data miss cycles})$$
$$= 3 + (0.04)(125) + (.6)(.08)(125) = 14$$

2. To improve the performance of our processor, we add a unified L2 cache between the two L1 caches and main memory. Our L2 cache has a hit time of 25 cycles and a global miss rate of 3%. What is our new $\text{CPI}_{\text{stall}}$?

$$\text{CPI}_{\text{stall}} = \text{CPI}_{\text{ideal}} + \sum_i ((\text{L}i \text{ instruction miss cycles}) + (\text{L}i \text{ data miss cycles}))$$
$$= 3 + (0.04)(25) + (.6)(.08)(25) + (.03)(125) + (.6)(.03)(125) = 11.2$$

Cache Question

1. The following C function `sum_iter` is run on a 32-bit MIPS machine. On this system, textttstruct Nodes are aligned to two-word boundaries, since `sizeof(struct Node)` is 8. Assume the total space taken up by the linked list is greater than (and a multiple of) the cache size.

```
struct Node {
  int n;
  struct Node *next;
};
int sum_iter(struct Node *head) {
  int sum = 0;
  while (head != NULL) {
    sum += head - >n; // load from head+0
    head = head - >next; // load from head+4
  }
  return sum;
}
```

Given a direct-mapped data cache with 13 index bits and 7 offset bits, how many words are in a cache block and how many bytes of data does this cache hold?

$$\text{block size} = 2^{(\text{offset bits})} = 2^7 \text{ bytes} = 32 \text{ words}$$
$$\text{cache size} = (\text{block size}) * 2^{(\text{index bits})} = 2^7 * 2^{13} = 2^{20} \text{ bytes} = 1 \text{ MiB}$$

2. What is the lowest and highest possible cache hit rates for the `while` loop in `sum_iter`, and under what conditions do each occur?

   The lowest is 50%, because every time it loads `head+0`, `head+4` is pulled into the same block. This can occur when every load kicks out the previously loaded block.

   The highest is $\left(\frac{32-1}{32}\right) * 100\% = 96.875\%$, when the entire block pulled in with one load is made up of various other nodes that have not yet been requested. This does not necessarily imply that the nodes are right next to each other in memory. Instead, it implies that when a block is kicked out, every node in that block has already been accessed.