CS3350 Assignment 1

MingCong Zhou - 250945414

January 31, 2020

Exercise 1

Define the processor-memory gap. How have computer architectures changed to combat this gap? What metric looks to be minimized by these changes? Be specific with this metric; of course minimizing one metric might also decrease another, higher-level metric.

Solution 1

What is the Processor Memory gap?
Memory grows little by little every year, while CPU grows rapidly every year(doubling the amount, by Moore's Law). This situation result in memory can't follow closely to the speed of CPU, which is namely the Processor Memory gap.
How to solve memory wall?
We use the hierarchy of memory to solve the memory wall issue(cache, main memory, secondary).
What metric looks to be minimized by these changes?
Latency looks to minimized. Memory hierarchy follows the principle of locality. We are less likely to fetch data through secondary memory after we introduced the hierarchy of memory.

Exercise 2

Consider the following function fib, which computes a Fibonacci number of a given index. Discuss the instruction locality of the function. Is there good locality or bad locality? Is there spatial locality or temporal locality? Explain why. For ease of discussion, you may consider a particular execution of the function, say, fib(4).

```
int fib(int n) {
   if (n < 2) {
      return n;
   }
   int n1 = fib(n-1);
   int n2 = fib(n-2);
   return n1 + n2;
```

```
}
```

Solution 2

Temporal Locality: n.
fib function would modify the 'n' variable in each iteration.
There is NO Spatial Locality:
In the above algorithm, we are neither contiguous access memory or sequential access memory.
Notice: even though we are reusing n1 and n2 every time by the recursive call. However, we are not reusing it. We are keep overwriting the space. Consider, fib(4):
int n1 = fib(3), int n2 = fib(2), each space is independent from each other.

Exercise 3

Discern a single formula for the average memory access time for a computer with three levels of cache. The hit rates of those levels are 90%, 85%, and 80%, respectively. The latency of main memory is 500 clock cycles. Leave anything in the AMAT formula which has not been given a specific value as a parameter.

Solution 3

AMAT = Time for Hit + Miss Rate * Miss Penalty
Miss Penalty = Lantency of main memory = 500 clock cycles
L1 Miss Rate = 1 - 0.9 = 0.1
L2 Miss Rate = 1 - 0.85 = 0.15
L3 Miss Rate = 1 - 0.8 = 0.2
AMAT = L1 Hit Time + 0.1 * L1 Miss Penalty
L1 Miss Penalty = L2 Hit Time + 0.15 * L2 Miss Penalty
L2 Miss Penalty = L2 Hit Time + 0.2 * 500
AMAT = L1 Hit Time + 0.1 * (L2 Hit Time + 0.15(L3 Hit Time + 100))

Exercise 4

The following tables summarizes the instructions present in some program (Table 1) as well as two different processors (Table 2).

Operation	*Instruction Count*
ALU	2000
Load	1000
Store	600
Branch	400

Table 1: Program Instructions

Operation	*Processor* 1	*Processor* 2
ALU	3	1
Load	4	6
Store	3	3
Branch	2	2

Table 2: Processor Instruction Cycles

(a) 5 Marks Calculate the ideal CPI of this program running on Processor 1.
Solution: CPI_{ideal} on Processor 1

$$CPI_{ideal} = \frac{\sum(InstructionCount * InstrictionCycle)}{TotalProgramInstruction}$$

$$CPI_{ideal} = \frac{2000 * 3 + 1000 * 4 + 600 * 3 + 400 * 2}{2000 + 1000 + 600 + 400} = 3.15$$

(b) 5 Marks Calculate the ideal CPI of this program running on Processor 2.
Solution: CPI_{ideal} on Processor 2

$$CPI_{ideal} = \frac{2000 * 1 + 1000 * 6 + 600 * 3 + 400 * 2}{4000} = 2.65$$

(c) 5 Marks Assuming that Processor 1 operates at 2.8 GHz and Processor 2 operates at 2.4 GHz calculate the CPU time for the program when run on each processor. On which processor is the program faster? (Hint: for easy calculations, 1GHz = 1 * 109Hz)
Solution: Given: Processor 1 CR = 2.8 GHz, Processor 2 CR = 2.4 GHz

$$CPUTimeProcessor1 = \frac{IC * CPI}{CR} = 4000 * 3.152.8 * 10^9 = 4.5 * 10^{-6}s$$

$$CPUTimeProcessor2 = \frac{IC * CPI}{CR} = 4000 * 2.652.4 * 10^9 = 4.4 * 10^{-6}s$$

Processor 2 is slightly faster than Processor 1

(d) 5 Marks In terms of the three factors calculating CPU time (instruction count, CPI, clock frequency) explain why the processor from part (c) is the faster one. How could a programmer change the program described in Table 1 so that it would run faster on the other processor?
Solution:
Explain why the processor from part (c) is the faster one:
Processor 2 has a lower clock frequency than Processor 1
This make the CPI_{ideal} in Processor 2 is less than processor 1.
Therefore processor 2 run faster than processor 1.
How could a programmer change the program?
We reduce the instruction in ALU while adding instrictions in Load to make the program run faster on processor 1.

(e) 10 Marks Calculate the CPU time of the program described in Table 1 running on Processor 1 (at 2.8 GHz) when also considering memory stall cycles (and thus have CPU time calculated using CPIstall). Assume this processor has one level of cache, an instruction miss rate of 2%, a data miss rate of 5% and a miss penalty of 100 cycles.

Solution: Processor 1 CR = 2.8 GHz

$$CPUTime = \frac{IC * CPI}{CR}$$

$$CPI_{stall} = CPI_{ideal} + AverageMemoryStallCycle$$

Access Count = $\frac{1000+600}{4000} = 0.4$

$AverageMemoryStallCycle = AccessCount * MissRate * MissPenalty$
Instruction Miss Rate = 2%
Data Miss Rate = 5%
Miss Penalty = 100 Cycles
Average Memory Stall Cycle = 1 * 0.02 * 100 + 0.4 * 0.05 * 100

$$CPI_{stall} = CPI_{ideal} + 4 = 3.15 + 4 = 7.15$$

$$CPUTime = \frac{4000 * 7.15}{2.8 * 10^9} = 1.02 * 10^{-5}$$

Exercise 5

Consider a 32-bit computer with a simplified memory hierarchy. This hierarchy contains a single cache and an unbounded backing memory. The cache has the following characteristics:

- Direct-Mapped, Write-through, Write allocate.
- Cache blocks are 16 words each.
- The cache has 128 sets.

(a) 4 Marks Calculate the cache's size in bytes.
Solution:
32 bit computer = 4 bytes computer
16 words per line = 16 * 4 = 64 bytes, each line contains 64 bytes (N = 1 Direct-Mapped)
there are 128 sets in the cache (R)
therefore C = BNR = 64 * 1 * 128 = 8192 bytes

(b) 12 Marks Consider the following code fragment in the C programming language to be run on the described computer. Assume that: program instructions are not stored in cache, arrays are cache-aligned (the beginning of the array aligns with the beginning of a cache line), ints are 32 bits, and all other variables are stored only in registers.

```
int N = 16384;
int A[N];
for (int i = 1; i < N; i += 2) {
   A[i-1] = A[i];
}
```

Determine the following:

(i) The number of cache misses.
Solution: int = 32 bit = 4 bytes
64/4 = 16 ints, therefore we have 16 integer per line Array Size: 16384 Loop Number: 16384 / 2 = 8192 times
Every 16 times would occur a cold miss.
16384 / 16 = 1024 cache misses.

(ii) The cache miss rate.
Solution: 1024/16384 = 6.25% Miss Rate

(iii) The type of cache misses which occur.
Solution: Compulsory Miss

(c) 14 Marks Consider the following code fragment in the C programming language to be run on the described computer. In addition to the assumptions of part (b), assume that that the array B immediately follows the array A in memory.

```
int N = 16384;
int A[N];
int B[N];
for (int i = 0; i < N; ++i) {
   A[i] = B[i];
}
```

(i) The number of cache misses.
Solution: number of cache misses = 16384 * 2 = 49152

(ii) The cache miss rate.
Solution: cache miss rate = 100

(iii) The type of cache misses which occur.
Solution: Since A[i] and B[i] will overwrite each other. Therefore, cold miss and conflict miss occur.

Exercise 6

Below is a sequence of references to 16-bit memory word addresses.

1, 18, 14, 16, 2, 3, 17, 18, 14, 15, 20, 21

(a) Consider an initially empty, direct-mapped cache with 2-word cache lines and capacity of 32 bytes. Using the sequence of address references above, determine if each address referenced results in a hit or a miss. If the reference results in a cache miss, say which type of cache miss (cold, conflict, capacity) You may use the table below to help answering this question, but for the purposes of marking, we do not care about the tag, index, or

block offset.

16 bits memory word address, 16 bits = 2 bytes
2 word(4 bytes) per line capacity = 32 bytes 32 / 4 = 8 lines 8 lines = 2^3, therefore we need 3 bits of index

(b) Consider an initially empty, 2-way set associative cache with 2-word cache lines, a capacity of 32 bytes, and an LRU (least recently used) eviction policy. Using the sequence of address references above, determine if each address referenced results in a hit or a miss. If the reference results in a cache miss, say which type of cache miss (cold, conflict, capacity) You may use the table below to help answering this question, but for the purposes of marking, we do not care about the tag, index, or block offset.

a:

Word Address	Tag	Index	Block Offset	Hit/Miss	Type of Miss
1	0000 0000 000	000	01	Miss	Cold
18	0000 0000 000	100	10	Miss	Cold
14	0000 0000 000	011	10	Miss	Cold
16	0000 0000 000	100	00	Hit	
2	0000 0000 000	000	10	Hit	
3	0000 0000 000	000	11	Hit	
17	0000 0000 000	100	01	Hit	
18	0000 0000 000	100	10	Hit	
14	0000 0000 000	011	10	Hit	
15	0000 0000 000	011	11	Hit	
20	0000 0000 000	101	00	Miss	Cold
21	0000 0000 000	101	01	Hit	

b:

Word Address	Tag	Index	Block Offset	Hit/Miss	Type of Miss
1	0000 0000 0000	00	01	Miss	Cold
18	0000 0000 0001	00	10	Miss	Cold
14	0000 0000 0000	11	10	Miss	Cold
16	0000 0000 0001	00	00	Miss	Cold
2	0000 0000 0000	00	00	Miss	Conflict
3	0000 0000 0000	00	10	Hit	
17	0000 0000 0001	00	11	Hit	
18	0000 0000 0001	00	01	Miss	Conflict
14	0000 0000 0000	11	10	Hit	
15	0000 0000 0000	11	11	Hit	
20	0000 0000 0001	01	00	Miss	Cold
21	0000 0000 0001	01	01	Hit	