Homework 7

Name: Yiqiao Jin
UID: 305107551

5.3

5.3 For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache.

Tag	Index	Offset
31–10	9–5	4–0

5.3.1 [5] <§5.3> What is the cache block size (in words)?

5.3.2 [5] <§5.3> How many entries does the cache have?

5.3.3 [5] <§5.3> What is the ratio between total bits required for such a cache implementation over the data storage bits?

Starting from power on, the following byte-addressed cache references are recorded.

Address											
0	4	16	132	232	160	1024	30	140	3100	180	2180

5.3.4 [10] <§5.3> How many blocks are replaced?

5.3.5 [10] <§5.3> What is the hit ratio?

5.3.1

5 offset bits indicates a block size of 2^5 bytes, or 32 bytes。This means 8 words per block (1 word is 4 bytes).

5.3.2

5 index bits means 2^5 sets, or 32 sets. We know that there is 1 entry per set for direct mapped cache, so there are 32 entries in total.

5.3.4

Address	Tag Index Offset	H/M
0	00 00000 00000	M
4	00 00000 00100	H
16	00 00000 10000	H
132	00 00100 00100	M
232	00 00111 01000	M
160	00 00101 00000	M

Address	Tag Index Offset	H/M
1024	01 00000 00000	M
30	00 00000 11110	M
140	00 00100 01100	H
3100	11 00000 11100	M
180	00 00101 10100	H
2180	10 00100 00100	M

For addresses 1024, 30, and 3100, the blocks are replaced. So 3 blocks are replaced in total.

5.3.5

There are 4 hits for 12 instructions. So hit rate is $\frac{4}{12}$, or $\frac{1}{3}$.

5.5 Media applications that play audio or video files are part of a class of workloads called "streaming" workloads; i.e., they bring in large amounts of data but do not reuse much of it. Consider a video streaming workload that accesses a 512 KiB working set sequentially with the following address stream:

0, 2, 4, 6, 8, 10, 12, 14, 16, ...

5.5.1 [5] <§§5.4, 5.8> Assume a 64 KiB direct-mapped cache with a 32-byte block. What is the miss rate for the address stream above? How is this miss rate sensitive to the size of the cache or the working set? How would you categorize the misses this workload is experiencing, based on the 3C model?

5.5.2 [5] <§§5.1, 5.8> Re-compute the miss rate when the cache block size is 16 bytes, 64 bytes, and 128 bytes. What kind of locality is this workload exploiting?

5.5.1

For a 32-byte block, we draw in 32 contiguous addresses upon each cache miss.

For this access pattern, there will be one miss for every 16 accesses, giving a miss rate of $\frac{1}{16}$. Each miss is a compulsory miss, meaning that the block is NOT seen by the cache in previous accesses of data memory.

5.5.2

For 16-byte blocks, we draw in 16 contiguous addresses upon each cache miss. A single block will contain 8 accesses. There is 1 cold miss for every 8 accesses, giving a miss rate of $\frac{1}{8}$

For 64-byte blocks, there is 1 cold miss for every 32 accesses, giving a miss rate of $\frac{1}{32}$

For 128-byte blocks, there is 1 cold miss for every 64 accesses, giving a miss rate of $\frac{1}{64}$

In this case, we exploit spatial locality - if we use an entry, its adjacent entries are more like to be used in the near future.

5.6 In this exercise, we will look at the different ways capacity affects overall performance. In general, cache access time is proportional to capacity. Assume that main memory accesses take 70 ns and that memory accesses are 36% of all instructions. The following table shows data for L1 caches attached to each of two processors, P1 and P2.

	L1 Size	L1 Miss Rate	L1 Hit Time
P1	2 KiB	8.0%	0.66 ns
P2	4 KiB	6.0%	0.90 ns

5.6.1 [5] <§5.4> Assuming that the L1 hit time determines the cycle times for P1 and P2, what are their respective clock rates?

5.6.2 [5] <§5.4> What is the Average Memory Access Time for P1 and P2?

5.6.3 [5] <§5.4> Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 and P2? Which processor is faster?

For the next three problems, we will consider the addition of an L2 cache to P1 to presumably make up for its limited L1 cache capacity. Use the L1 cache capacities and hit times from the previous table when solving these problems. The L2 miss rate indicated is its local miss rate.

L2 Size	L2 Miss Rate	L2 Hit Time
1 MiB	95%	5.62 ns

5.6.4 [10] <§5.4> What is the AMAT for P1 with the addition of an L2 cache? Is the AMAT better or worse with the L2 cache?

5.6.2

P1: AMAT = hit time + miss rate * penalty = 0.66 + 0.08 * 70 = 6.26

P2: AMAT = 0.90 + 0.06 * 70 = 5.10

5.6.4

P1: New AMAT = 0.66 + 0.08 * (5.62 + 0.95 * 70) = 6.43

The new AMAT is worse due to the high miss rate of L2 Cache.

5. ***Why, oh why, must we do TCPI? (40 points)***: We are going to assess branch and cache performance on the pipelined datapath from class – we have full data forwarding. Our peak CPI is 1.0. Assume that 30% of instructions are branches, and that we have a single cycle branch hazard on this processor. Our branch predictor **always** guesses *not taken*. 50% of branches are not taken. Our processor has an instruction cache and data cache – both take a single cycle to access. The instruction cache miss rate is 10% and the data cache miss rate is 30%. The next level of the memory hierarchy is an L2 cache with a miss rate of 20% and an access time of 10 cycles, this is in addition to the L1 cache latency. Main memory has an access time of 80 cycles, this is in addition to the latency of the L1 and L2 caches. 20% of instructions are loads, and stores do not stall the processor on a cache miss. 3/5ths of loads have dependent instructions following them. Our target application executes 1,000,000 instructions. The processor clock runs at 2 GHz.

   a. Calculate the average memory access time (AMAT) of the processor:

      AMAT: 8.8 cycles

L1 miss penalty: $0.2 \times 80 + 10 = 26$

AMAT = $1 + 0.3 \times 26 = 8.8$

   b. Calculate TCPI for our target application on our processor.

      TCPI: 5.43

Base CPI = $1 + \underbrace{0.3 \times 0.5}_{\text{Branch}} + \underbrace{0.6 \times 0.2}_{\text{load}} = 1.27$

MemCPI = $\underbrace{1 \times 0.1 \times 26}_{\text{instruction cache}} + \underbrace{0.2 \times 0.3 \times 26}_{\text{data cache}} = 4.16$

(1 = miss rate; 0.2 = % loads; 0.3 = miss rate)

TCPI = Base CPI + Mem CPI = 5.43

Amortized CPI: $1 + \underbrace{0.2}_{\text{loads}} = 1.2$

operations: $10^6 + \underbrace{0.2 \times 10^6}_{\text{loads}} = 1.2 \times 10^6$

c. Suppose 1/6th of all branches are procedure calls. Each procedure call (i.e. a jal instruction) in our application also has a return (i.e. a jr instruction). These will all be mispredicted because we always guess not taken. One approach to reducing branch hazards in such a case is to *in-line* the procedure call. The compiler basically takes the instructions in the body of the procedure call and replaces **all** calls to that procedure with these instructions. This means that instead of the code:

```
        add $s0, $s0, $t1
        jal Target
        add $s0, $s0, $t2
        jal Target
        …..
        …..
        …..
        …..
Target: lw $t3, 0 ($s0)
        addi $t3, $t3, 200
        sw $t3, 0 ($s0)
        jr $ra
```

We would have the code:

```
        add $s0, $s0, $t1
        lw $t3, 0 ($s0)
        addi $t3, $t3, 200
        sw $t3, 0 ($s0)
        add $s0, $s0, $t2
        lw $t3, 0 ($s0)
        addi $t3, $t3, 200
        sw $t3, 0 ($s0)
        …..
        …..
        …..
        …..
```

The benefit in this simple example is that we avoid four branches (two jal's and two jr's), but the size of the instruction text segment in memory (i.e. the size of the actual program we are running) has increased. Now, instead of the lw, addi, and sw being in one place in the text segment, they are in two places. This can increase the miss rate of the instruction cache.

Suppose that we try in-lining on our processor. In order for performance to improve, the cost of increasing the instruction cache miss rate must not exceed the benefit of reducing branch hazards. Using TCPI as the CPI in the equation for Execution Time, provide an upper bound on the miss rate of the instruction cache to improve performance when using in-lining. Assume that the L2 cache's miss rate does not change.

The instruction cache miss rate must be <= 0.1197

% jal = % jr = $\frac{1}{6}$

So totally $\frac{1}{6}+\frac{1}{6}=\frac{1}{3}$, or 33.3% branches are removed when inlining

New # branches: $0.3\times10^6\times(1-\frac{2}{3}) = 0.2\times10^6$ instructions

10^6 instructions in all

less $\frac{1}{3}$ × 0.3×10^6 instructions

	Old	New	ratio
R-type	0.5×10^6	0.5×10^6	$\frac{5}{9}$
Load/Store	0.2×10^6	0.2×10^6	$\frac{2}{9}$
Branch	0.3×10^6	0.2×10^6	$\frac{2}{9}$
Total	1×10^6	0.9×10^6	1

0.15×10^6 taken (old) → 0.05×10^6 taken (new) (25%)

0.15×10^6 NOT taken (75%)

Predict NOT taken:

Base CPI = $1+\frac{2}{9}\times\underbrace{0.25}_{\text{mis-prediction}}\times\underbrace{1}_{\text{single cycle hazard}}+\underbrace{\frac{2}{9}\times0.6}_{\text{lw}}\approx1.189$

Let x = Instruction Miss Rate

$\text{memCPI} = (x)(1)(26)+(\frac{2}{9})(0.3)(26) = 26x+\frac{26}{15}\approx 26x+1.733$

(0.3: data miss rate)

$TCPI = 1.189+26x+1.733 = 2.922+26x$ (old CPI)

$ET = TCPI\cdot IC = (2.922+26x)\cdot0.9\cdot10^6 \le 5.43\cdot10^6$ (old IC)

Instruction miss rate: $x \le 0.1197$