CS M151B: Homework 6 Solutions

5.3.1

Block size in bytes can be determined from the number of offset bits. 5 offset bits means block size is 2^5 or 32 bytes. 32 bytes/block * (1 word / 4 bytes) = 8 words/block

5.3.2

The cache is direct mapped, which means there is one entry per set. The number of sets can be determined by the number of index bits. 5 index bits means 2^5 or 32 sets. Because there is 1 entry per set, there are 32 entries total.

5.3.4

	Tag	Index	Offset	Hit/Miss?	Replaced?
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	
1024	…01	00000	00000	M	Y
30	…00	00000	11110	M	Y
140	…00	00100	01100	H	
3100	…11	00000	11100	M	Y
180	…00	00101	10100	H	
2180	…10	00100	00100	M	

4 blocks are replaced.

5.3.5

Hit rate = no. hits / total no. of access = 4 / 12 = 1/3

5.5.1

Because the block size is 32 bytes, the block corresponds to 32 contiguous addresses. Thus, for this particular access pattern of 0, 2, 4, … where each access is incremented by 2, the pattern will span a single cache block in 16 accesses. Based on this access pattern and the block size of 32 bytes, for every 16 accesses, there will be one miss. This results in a miss rate of 1/16. This miss occurs because the block has not previous been seen in the cache, making it a cold or compulsory miss.

5.5.2

For a 16 byte block size:

- 8 accesses will span a single block with a single cold miss for every 8 accesses.

- Miss rate: 1/8

For a 64 byte block size:

- 32 accesses will span a single block.
- Miss rate: 1/32

For a 128 byte block size:

- Miss rate: 1/64

This is exploiting spatial locality, or the idea that if an item is referenced, nearby items are like to be referenced in the near future.

5.6.2

AMAT = hit time + miss rate * miss penalty
Time and penalty can be in seconds or cycles.

L1 hit time determines the cycle time.

P1:
L1 miss rate: 8%
L1 hit time/cycle time: .66 ns
Clock rate: 1/.66 * 10^9 = 1.515 GHz
Memory access time: 70ns
Memory access cycles: 70ns * 1.515 * 10^9 cycles/second = 106.05

AMAT(cycles) = 1 + .08*106.05 = 9.484

P2:
L1 miss rate: 6%
L1 hit time/cycle time: .9 ns
Clock rate: 1/.9 * 10^9 = 1.111 GHz
Memory access time: 70ns
Memory access cycles: 70ns * 1.111 * 10^9 cycles/second = 77.77

AMAT(cycles) = 1 + .06*77.77 = 5.6662

5.6.4

L2 miss rate: 95%
L2 hit time: 5.62
L2 access cycles = 5.62*1.515*10^9 = 8.5143

New penalty for missing L1: (8.5143 + .95*106.05)
New AMAT(cycles) = 1 + .08*(8.5143 + .95*106.05) = 9.751

AMAT actually becomes slightly worse due to the high miss rate of the L2 cache.

5. Why, oh why, must we do TCPI?
a.

10^6 instructions
30% instructions are branches

- Predict not taken
- Branch hazard stall is 1 cycle, assume no new branch data hazards
- 50% of the branches are actually not taken
- $.3*10^6$ branches total.
- $.15*10^6$ branches mispredicted

20% of instructions are loads

- Full forwarding, which means 1 stall cycle for load use dependency
- 3/5 (60%) of loads are immediately followed by a dependent
- $.2*10^6$ loads total

D$ miss rate: 30%
I$ miss rate: 10%
L2 miss rate: 20%
L2 access time: 10 cycles
Memory access time: 80 cycles

AMAT (Data Memory) = 1 + .3*(10 + .2*80) = 8.8
AMAT (Insn Memory) = 1 + .1*(10 + .2*80) = 3.6

AMAT is an average over all memory accesses. There are 10^6 instructions and $.2*10^6$ loads, which means a total of $1.2*10^6$ memory operations.

Total AMAT = .2/1.2 * AMAT(Data) + 1/1.2 * AMAT(Insn)

…but actually the professor was only looking for AMAT(Data)

b.

TCPI = BaseCPI + MemCPI
BaseCPI = PeakCPI + Data Hazard CPI (DHCPI) + Control Hazard CPI (CHCPI)
MemCPI = D$CPI + I$CPI

PeakCPI = 1
DHCPI = (% of instructions that cause load hazard) * (load stall penalty)
= .2 * .6 * 1 = .12
CHCPI = (% of instructions that are branches) * (% of branches that are mispredicted) * (branch stall penalty)
= .3 * .5 * 1 = .15
BaseCPI = 1 + .12 + .15 = 1.27

D$CPI = (% of insns that go to D$) * (D$ miss rate) * (D$ miss penalty)
= .2 * .3 * (10 + .2 * 80) = 1.56
I$CPI = (I$ miss rate) * (I$ miss penalty)
= .1 * (10 + .2 * 80) = 2.6
MemCPI = 1.56 + 2.6 = 4.16

TCPI = 1.27 + 4.16 = 5.43

c.

Old_TCPI = 5.43

We have reduced the amount of mis-predicted branches to improve TCPI, but at the potential cost of the I$ miss rate. In order to determine what a viable I$ miss rate would be, we would like to consider the New_TCPI with the assumption that:

Old_ET > New_ET

If 1/6 of branches are procedure calls (jal), then another 1/6 of branches are (jr). Both jal and jr are always taken, which means that in our “always predict not taken” scheme all of these branches are mispredictions. We improve upon this by inlining and removing all jal and jr’s. This amounts to 2/6 or 1/3 of branches.

Originally, there were .3 * 10^6 branches. Now, we are eliminating 1/3 of them, or .1 *10^6 branches. Now:

New instruction count: 10^6 - .1*10^6 = .9*10^6
New branch count: .3*10^6 - .1*10^6 = .2*10^6

New branch %: .2 / .9 = 2/9

Originally .15 * 10^6 branches were predicted correctly and the same amount were predicted incorrectly. We are reducing the amount of mispredicted branches by .1 * 10^6 so now:

- Number of mispredicted branches: (.15 - .1) * 10^6 = .05 * 10^6
- Number of correctly predicted branches: .15 * 10^6
- New branch mispredication rate: .05 / .2 = ¼

The lw count stays the same (.2*10^6), but the percentage changes.

- New lw %: .2 / .9 = 2/9

PeakCPI = 1

DHCPI = (% of instructions that cause load hazard) * (load stall penalty)

= (2/9) * .6 * 1 = .133

CHCPI = (% of instructions that are branches) * (% of branches that are mispredicted) * (branch stall penalty)

= (2/9) * (1/4) * 1 = .0556

BaseCPI = 1 + .133 + .0556 = 1.1886

D$CPI = (% of insns that go to D$) * (D$ miss rate) * (D$ miss penalty)

= (2/9) * .3 * (10 + .2 * 80) = 1.733

I$CPI = (I$ miss rate) * (I$ miss penalty)

= *i* * (10 + .2 * 80) = 26*i*

MemCPI = 1.733 + 26*i*

New_TCPI = 1.1886 + 1.733 + 26*i* = 2.9216 + 26*i*

Old_ET > New_ET

Old_ET = Old_TCPI * Old_IC * Old_CT

= 5.43 * 10^6 * ½ * 10^{-9}

= .002715

New_ET = New_TCPI * New_IC * New_CT

= (2.9216 + 26*i*) * .9*10^6 * ½ * 10^{-9}

= .001315 + .0117*i*

.002715 > .001315 + .0117*i*

.1197 > *i*

To be a viable improvement, the instruction cache miss rate must be less than 11.97%.