Solution of Homework 3

Questions 1: Consider a five stage pipelined processor in which the branch target address is determined in the ID stage (second stage) and the branch condition is determined in the MEM stage (fourth stage). Assume that the instruction mix executing on the processor contains 20% branch instructions, 70% of which are taken. Assume also that the processor’s CPI is 1.9 without accounting for the overhead of dealing with control hazards (but accounting for other hazards). Compute the CPI when each of the following techniques is used to avoid control hazards.

(a) Always predict “branch not taken” and take a corrective action if the prediction is wrong
\[
\text{CPI} = 1.9 + 0.2 \times 0.7 \times 3 = 1.9 + 0.42 = 2.32
\]

(b) Always predict “branch taken” and take a corrective action if the prediction is wrong (see slide 70)
\[
\text{CPI} = 1.9 + 0.2 \times (0.7 \times 1 + 0.3 \times 2) = 1.9 + 0.2 \times 1.3 = 1.9 + 0.26 = 2.16
\]

(c) Assume a branch predictor with an 80% accuracy. That is, the predictor makes the right prediction 80% of the time and the wrong prediction 20% of the time. For this part assume that a predictor always makes a prediction and ignore the cases where the predictor cannot make a prediction because of a BTB collision or invalid entry (branch seen for the first time).

Assuming Branch predictor also predicts the target address
\[
\text{CPI} = 1.9 + 0.2 \times (0.2 \times 3) = 1.9 + 0.12 = 2.02
\]

Question 2: Consider the following loop which computes \(a[i]=b[i]+a[i]\), \(i=1,2,…\):

L: lw $t0, 1000($s4) /* load \(a[i]\) */
    lw $t1, 4000($s4) /* load \(b[i]\) */
    addi $s4, $s4, 4
    add $t0, $t0, $t1 /* compute temp = \(a[i] + b[i]\) */
    sw $t0, 996($s4) /* store temp in \(a[i]\) */
    beq $s4, $s6, L /* repeat until \(i\) reaches a maximum value */

(a) Using a table similar to the one below, show the scheduling of one iteration of the loop on a superscalar (static) 2-issue architecture with two pipelines, one for ALU/branch instructions and the other for lw/sw instructions (see Figure 4.69). Assume that the architecture has forwarding and data hazard hardware.

<table>
<thead>
<tr>
<th>ALU/branch pipeline</th>
<th>Load/store pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>lw</td>
</tr>
<tr>
<td>addi</td>
<td>lw</td>
</tr>
<tr>
<td>add</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) How many cycles will it take to execute the above loop 1000 times on the 2-issue architecture assuming perfect branch prediction (always correct prediction)?

One iteration → 6 instructions in 5 cycles
1000 iterations → 6000 instructions → 5000 cycles → CPI = 5/6
(c) If the branch is resolved when the instruction is in the ID stage (second stage), how many cycles will it take to execute the above loop 1000 times on the 2-issue architecture assuming no branch prediction (always predict “branch is not taken”)?
Due to wrong predictions, there will be one bubble after each branch (except the last)
One iteration → 6 instructions in 6 cycles
1000 iterations → 6000 instructions → 6000 cycles → CPI = 6/6 = 1

(d) How many cycles will it take to execute the same loop 1000 times on a single issue architecture (the usual 5-stage pipeline) assuming perfect branch prediction?
There will be no data hazard if the loop executes on a single pipeline:
One iteration → 6 instructions in 6 cycles
1000 iterations → 6000 instructions → 6000 cycles → CPI = 1

(e) If the branch is resolved when the instruction is in the ID stage (second stage), how many cycles will it take to execute the above loop 1000 times on the single issue architecture assuming no branch prediction?
Due to wrong predictions, there will be one bubble after each branch (except the last)
One iteration → 6 instructions in 7 cycles
1000 iterations → 6000 instructions → 7000 cycles → CPI = 7/6

Question 3:
(a) Assuming that the loop in Question 3 executes an even number of times, show that loop after unrolling it twice (reducing the number of iterations by a factor of 2).

L: lw $t0, 1000($s4)
lw $t1, 4000($s4)
lw $t2, 1004($s4)
lw $t3, 4004($s4)
addi $s4, $s4, 8
add $t0, $t0, $t1
add $t2, $t2, $t3
sw $t0, 992($s4)
sw $t2, 996($s4)
beq $s4, $s6, L

(b) Show the scheduling of one iteration of the unrolled loop on a superscalar (static) 2-issue architecture with two pipelines, one for ALU/branch instructions and the other for lw/sw instructions (see Figure 4.69). Assume that the architecture has forwarding and data hazard hardware.
(c) How many cycles will it take to execute the unrolled loop 1000 times on the 2-issue architecture assuming perfect branch prediction (always correct prediction)?

Two iterations of original loop → one iteration of the unrolled loop → 7 cycles
1000 iterations of the original loop → 3500 cycles

Question 4:
In this question, consider the following series of address references (given as word addresses):
20, 33, 21, 4, 21, 39, 21, 17, 28, 48, 45, 33, 9, 4, 22, 44

For each of the following cache organizations, show the content of the cache after each memory reference and indicate whether the reference is a hit or a miss. Use [tag, M(address), ...] to describe the content of each entry (see the examples on the class progress page). For example [4,M(46)] indicates that the entry contains tag=4 and the data from memory location 46. Similarly, [4,M(46),M(47)] indicates that the entry contains a block of two words from memory locations 46 and 47. As discussed in class, avoid drawing the cache after each reference by drawing only one cache and indicating that an entry E1 is replaced by E2 by crossing E1 and writing E2 next to it. Assume Least Recently Used replacement and assume that the cache is initially empty (invalid entries).

(a) a direct mapped cache with 16 one-word blocks.
20(M), 33(M), 21(M), 4(M), 21(H), 39(M), 21(H), 17(M), 28(M), 48(M), 45(M), 33(M), 9(M), 4(H), 22(M), 44(M)

0 [3, M(48)]
1 [2, M(33)] [1, M(17)] [2, M(33)]
2
3
4 [1, M(20)] [0, M(4)]
5 [1, M(21)]
6 [1, M(22)]
7 [2, M(39)]
8
9 [0, M(9)]
10
11
12 [1, M(28)] [2, M(44)]
13 [2, M(45)]
14
15

(b) a direct mapped cache with two-word blocks and total size of 16 words.
20(M), 33(M), 21(H), 4(M), 21(M), 39(M), 21(H), 17(M), 28(M), 48(M), 45(M), 33(M), 9(M), 4(M), 22(H), 44(H)

0, 1 [2, M(32), M(33)] [1, M(16), M(17)] [3, M(48), M(49)] [2, M(32), M(33)]
2, 3
4, 5 [1, M(20), M(21)] [0, M(4), M(5)] [1, M(20), M(21)] [0, M(4), M(5)]
6, 7 [2, M(38), M(39)] [1, M(22), M(23)]
(c) a 2-way associative cache with two-word blocks and total size of 16 words
20(M), 33(M), 21(H), 4(M), 21(H), 39(M), 21(H), 17(M), 28(M), 48(M), 45(M), 33(M), 9(M), 4(M), 22(M), 44(H)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8, 9</td>
<td>[0, M(8), M(9)]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10, 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12, 13</td>
<td>[1, M(28), M(29)] [2, M(44), M(45)]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14, 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question 5:
Compute the total number of bits required to implement each of the caches specified below.
Assume that each memory word is 32-bit long and the memory is byte addressable with 32-bits addresses. Note that the number of bits needed to implement the cache represents the total amount of memory needed for storing all of the data, tags and valid bits (ignore any other bits).

<table>
<thead>
<tr>
<th>Configuration</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache size</td>
<td>512KB</td>
<td>512KB</td>
<td>512KB</td>
<td>512KB</td>
</tr>
<tr>
<td>Block size</td>
<td>one word</td>
<td>4 words</td>
<td>one word</td>
<td>8 words</td>
</tr>
<tr>
<td>Associativity</td>
<td>direct mapped</td>
<td>direct mapped</td>
<td>4-way</td>
<td>2-way</td>
</tr>
<tr>
<td># of bits for data array</td>
<td>4096 Kbits</td>
<td>4096 Kbits</td>
<td>4096 Kbits</td>
<td>4096 Kbits</td>
</tr>
<tr>
<td># of bits for tag array</td>
<td>1664 Kbits</td>
<td>416 Kbits</td>
<td>1920 Kbits</td>
<td>224 Kbits</td>
</tr>
<tr>
<td># of bits for valid bits</td>
<td>128 Kbits</td>
<td>32 Kbits</td>
<td>128 Kbits</td>
<td>16 Kbits</td>
</tr>
<tr>
<td>Total # of bits</td>
<td>5888 Kbits</td>
<td>4544 Kbits</td>
<td>6144 Kbits</td>
<td>4336 Kbits</td>
</tr>
</tbody>
</table>

(a)
Cache Size: 512KB = 2^19 bytes
Block Size: 2^0 word = 2^2 bytes
# of blocks: (2^19)/(2^2) = 2^17
# of sets: 2^17 for direct mapped cache
Tag field: (32-2-17) = 13 bits
# of bits for data array = 4096Kbits
# of bits for tag array = (2^17) * 13 = 1664Kbits
# of bits for valid bits = (2^17) * 1 = 128Kbits
Total # of bits = 5888Kbits

(b)
Cache Size: 512KB = 2^19 bytes
Block Size: 2^2 word = 2^4 bytes
# of blocks: (2^19)/(2^4) = 2^15
# of sets: $2^{15}$ for direct mapped cache
Tag field: $(32-4-15) = 13$ bits
# of bits for data array = 4096Kbits
# of bits for tag array = $(2^{15}) * 13 = 416Kbits$
# of bits for valid bits = $(2^{15}) * 1 = 32Kbits$
Total # of bits = 4544Kbits
(c)
Cache Size: 512KB = $2^{19}$ bytes
Block Size: $2^0$ word = $2^2$ bytes
# of blocks: $(2^{19})/(2^2) = 2^{17}$
# of sets: $2^{15}$ for 4-way associativity
Tag field: $(32-2-15) = 15$ bits
# of bits for data array = 4096Kbits
# of bits for tag array = $4*(2^{15}) * 15 = 1920$Kbits
# of bits for valid bits = $4*(2^{15}) * 1 = 128$Kbits
Total # of bits = 6144Kbits
(d)
Cache Size: 512KB = $2^{19}$ bytes
Block Size: $2^3$ word = $2^5$ bytes
# of blocks: $(2^{19})/(2^5) = 2^{14}$
# of sets: $2^{13}$ for 2-way associativity
Tag field: $(32-5-13) = 14$ bits
# of bits for data array = 4096Kbits
# of bits for tag array = $2*(2^{13}) * 14 = 224$Kbits
# of bits for valid bits = $2*(2^{13}) * 1 = 16$Kbits
Total # of bits = 4336

Questions 6:
Consider a program in which 20% of the instructions are memory load or store instructions, and assume that the CPI for this machine is 2.5 when the data and instructions are always found in the cache.

1. How many cycles does it take to access the cache if the CPU operates at 2GHz and the memory access latency is 80 n. sec.
   Given a 2GHz CPU, each cycle lasts $1/(2*10^9)$ sec = 0.5 n sec
   It takes 160 cycles to access the cache if the memory access latency is 80 n sec.

2. Assume that the cache miss penalty is equal to the memory access time computed above, what is the average memory access time if the instruction cache miss rate is 2.5% and the data cache miss rate is 5%?
   Instruction miss rate: 2.5%, data miss rate: 5%
   Effective miss rate = 0.025 + 0.05 * 0.2 = 0.035
   $AMAT = 1 + 0.035 * 160 = 1 + 5.6 = 6.6$

3. Assume that an on-chip L2 cache is added to the system, and that the hit time for the L2 cache is 6 cycles. What would be the effective CPI if 30% of the references to the L2 cache (the misses from L1) are L2 misses?
   $CPI = 2.5 + 0.035 * 6 + 0.035 * 0.3 * 160 = 4.39$