Q1)

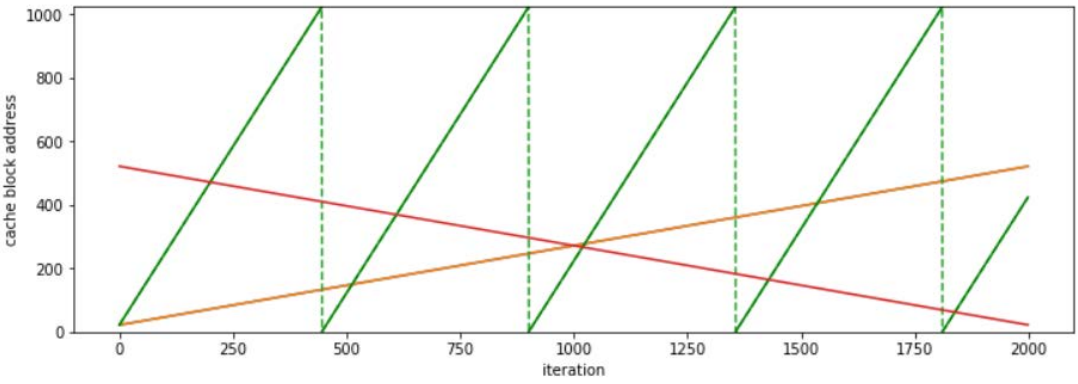
① Since my student ID is 2018142059, N=9 and the base address of array a is $0000010000000011000000101000010000000101100100 0_2$, and the base address of array b is $0010010000000011000000101000010000000101100100 0_2$. With those address, I got the memory addresses to access for each load and save instructions, by adding 8*(array number #) to their base address, since the system is byte addressable and one element of an array is 64bit=8byte. To get the cache block address, I get the memory address field configuration with 3bit byte offset, 2bit word offset, 10bit index bits and 31bit tag bits. Since cache block address is (memory block address) modulo (Number of cache blocks in the cache), it is same as 'index bit' part of memory address.

Since there are 4 load and save instruction in one iteration, 40 memory accesses are made for the code execution for the first 10 iterations. Also, I checked number of cache misses during the iteration, which was 17.

To get the cache miss rate of whole 2000 iterations, I found patterns how the cache block addresses change. Address for array b is same for an iteration, and it increases about 8*1 for an iteration. Address for first save instruction increases about 8*9 for an iteration, ant for the second save instruction it decreases about 8*1. Then, since the cache block is [14:5] ot the memory address, by dividing the memory address change by 2^5, we can get abstract slope of each cache block address change. The patterns can be written as the functions as below. B1, B2 is at the memory access at the load instructions which is for b array, A1 is at the first memory access at the save instruction, and A2 is the second and last memory access by save instruction in the iteration.

a1	1	(index)=22.25+2.25*x
	2	(index)=1.75*2.25*(x-446)
	3	(index)=1.5+2.25*(x-901)
	4	(index)=1.25+2.25*(x-1356)
	5	(index)=1+2.25*(x-1811)
a2	(index)=522.25-0.25*x	
b1, b2	(index)=22.25+0.25*x	

Then those graphs can be plotted like the figure below. Green plots are for a1, red plots are for a2, and orange plots are for b1 and b2. Firstly, since A1 has 5 cycles which means that each cycle has different tag value, therefore we can say that there is about 2000 misses. Then, subtract the number of misses that is by preceding a2 plot, which is from 0th to 200th iteration, then since the first index value of a2 is 522 and 200th index value of a2 is 472, we can get 2000-((522-472)*4/9≅1978. Then at the plot of B1, since the value range is 22 to 522, it misses approximately 500. Also, in the first iteration it gets another miss since A1 also have same index but different tag value, and there are 3 misses where the plot meets A2, near the 1000th iteration. Therefore, B1 have approximately 504 misses. Since B2 has same index value with B1 in an iteration, there is one miss at the first iteration, and three misses at the 1000th iteration. Where A2 plot and the second cycle of A1 plot meet has index value 370. Then there is about (522-370)-(522-370*4/9=84 misses in the range. The other part of plot A2 has 370-22=348 misses. Also, where the A2 plot meets A1 plot, there is about 12 misses. Where the plot meets B1/B2 plot, which makes 3 more misses. Therefore, there is about ((2000-22) + (500+1+3) + (1+3) + (84+348+12+3) = 2935. Therefore, the miss rate of this case is about 2935 /8000 * 100 ≅36.7%.



i	Instruction type	Instruction #	Memory Address	cache block address	Hit /Miss
0	load b	0	0900C0A102C8	0000010110	Miss
	save a	0	0100C0A102C8	0000010110	Miss
	load b	0	0900C0A102C8	0000010110	Miss
	save a	2000	0100C0A14148	1000001010	Miss
1	load b	1	0900C0A102D0	0000010110	Hit
	save a	9	0100C0A10310	0000011000	Miss
	load b	1	0900C0A102D0	0000010110	Hit
	save a	1999	0100C0A14140	1000001010	Hit
2	load b	2	0900C0A102D8	0000010110	Hit
	save a	18	0100C0A10358	0000011010	Miss
	load b	2	0900C0A102D8	0000010110	Hit
	save a	1998	0100C0A14138	1000001001	Miss
3	load b	3	0900C0A102E0	0000010111	Miss
	save a	27	0100C0A103A0	0000011101	Miss
	load b	3	0900C0A102E0	0000010111	Hit
	save a	1997	0100C0A14130	1000001001	Hit
4	load b	4	0900C0A102E8	0000010111	Hit
	save a	36	0100C0A103E8	0000011111	Miss
	load b	4	0900C0A102E8	0000010111	Hit
	save a	1996	0100C0A14128	1000001001	Hit
5	load b	5	0900C0A102F0	0000010111	Hit
	save a	45	0100C0A10430	0000100001	Miss
	load b	5	0900C0A102F0	0000010111	Hit
	save a	1995	0100C0A14120	1000001001	Hit
6	load b	6	0900C0A102F8	0000010111	Hit
	save a	54	0100C0A10478	0000100011	Miss
	load b	6	0900C0A102F8	0000010111	Hit
	save a	1994	0100C0A14118	1000001000	Miss
7	load b	7	0900C0A10300	0000011000	Miss
	save a	63	0100C0A104C0	0000100110	Miss
	load b	7	0900C0A10300	0000011000	Hit
	save a	1993	0100C0A14110	1000001000	Hit
8	load b	8	0900C0A10308	0000011000	Hit
	save a	72	0100C0A10508	0000101000	Miss
	load b	8	0900C0A10308	0000011000	Hit
	save a	1992	0100C0A14108	1000001000	Hit
9	load b	9	0900C0A10310	0000011000	Hit
	save a	81	0100C0A10550	0000101010	Miss
	load b	9	0900C0A10310	0000011000	Hit
	save a	1991	0100C0A14100	1000001000	Hit

② The base addresses of array a and array b are same as the previous one, The size of array element did not change so the memory address is also same. However, the field of memory address is different, since it is 4-way associative cache, it has 256 set, therefore index bits become 8 bits, and tag bits become 33 bits. There are 4 memory access instructions – 2 lh and 2 sh – in one iteration, therefore 40 memory accesses are made for the code execution for the first 10 iteration, and there occur 16 cache misses.

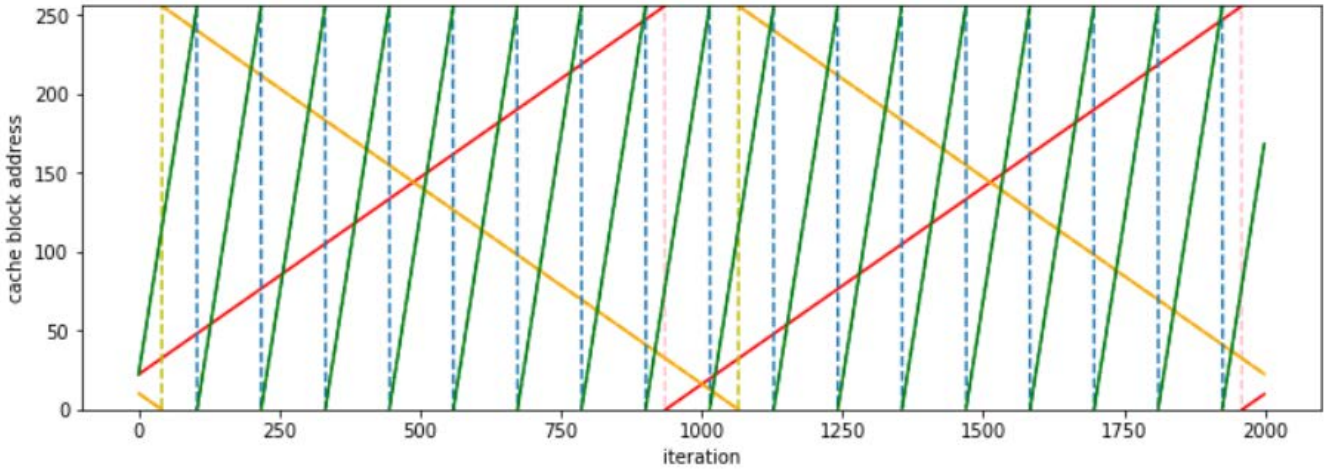
To get the cache miss rate of the 2000 iterations, patterns how the cache block addresses change can be used. Since cache block address is [12:5] of the memory address and the cache block addresses change almost linearly according to the array number, the pattern can be written in linear equations. The real index values are integer form of the linear equation value.

A1	1	(index)=22.25+2.25*x
	2	(index)=0.25+2.25*(x-104)
	3	(index)=0.75+2.25*(x-218)
	4	(index)=1.25+2.25*(x-332)
	5	(index)=1.75+2.25*(x-446)
	6	(index)=2.25+2.25*(x-560)
	7	(index)=0.5+2.25*(x-673)
	8	(index)=1+2.25*(x-787)
	9	(index)=1.5+2.25*(x-901)
	10	(index)=2+2.25*(x-1015)
	11	(index)=0.25+2.25*(x-1128)
	12	(index)=0.75+2.25*(x-1242)
	13	(index)=1.25+2.25*(x-1356)
	14	(index)=1.75+2.25*(x-1470)
	15	(index)=2.25+2.25*(x-1584)
	16	(index)=0.5+2.25*(x-1697)
	17	(index)=1+2.25*(x-1811)
	18	(index)=2+2.25*(x-1925)
A2	1	(index)=10.25-0.25*x
	2	(index)=255.75-0.25*(x-42)
	3	(index)=255.75+0.25*(x-1066)
B1, b2	1	(index)=22.25+0.25*x
	2	(index)=0.25*(x-936)
	3	(index)=0.25*(x-1960)

These can be plotted as the figure at the following page. Green plots are for A1, yellow plots are for A2, and red plots are for B1 and B2, which have always the same value in an iteration. Tag changes for each cycle, for A1 and B1/B2 plots, tag increases since the slope of the equation is positive value. For A2, tag decreases if the cycle changes as the iteration done.

Since index for first sh instruction changes for every iteration cycle, we can assume that there are 2000 misses that is related to the A1 index value. Also, since A2 and B1 get 500 different index or tag values each, and B2 does not have cache misses since it has same value to B1, we can assume that there are 2000+500+500=3000 misses, then we should subtract the number of misses that has been double counted. First cycle of A2 graph has same tag value of the third cycle of A1 graph, and the second cycle of A2 graph has same tag value of second cycle of A1, and the third cycle of A2 graph has the same tag value of first cycle of A1 graph. Therefore, there will be some double counted cache misses. Since slope of A1 graph is 2.25, the index increases often 2 or 3 once in 4 cycle. Therefore, the data assigned by the A1, first sh instruction in an iteration would rarely deleted until the A2, second sh instruction needed the data. Therefore, we can predict that the number of double counted cache misses would be little bit smaller but almost the number of memory accesses at the second cycle A1, first sh instruction. The value is 256*4/9≅113. Therefore, the number of cache misses in the whole 2000 times iteration would be similar to 2000+500+500-113=2887. Then the miss rate would be similar with 2887/8000*100≅36.1%.

i	Instruction type	Instruction #	Memory Address	Cache block address	Hit /Miss
0	lh b	0	0900C0A102C8	00010110	miss
	sh a	0	0100C0A102C8	00010110	miss
	lh b	0	0900C0A102C8	00010110	hit
	sh a	2000	0100C0A14148	00001010	miss
1	lh b	1	0900C0A102D0	00010110	hit
	sh a	9	0100C0A10310	00011000	miss
	lh b	1	0900C0A102D0	00010110	hit
	sh a	1999	0100C0A14140	00001010	hit
2	lh b	2	0900C0A102D8	00010110	hit
	sh a	18	0100C0A10358	00011010	miss
	lh b	2	0900C0A102D8	00010110	hit
	sh a	1998	0100C0A14138	00001001	miss
3	lh b	3	0900C0A102E0	00010111	miss
	sh a	27	0100C0A103A0	00011101	miss
	lh b	3	0900C0A102E0	00010111	hit
	sh a	1997	0100C0A14130	00001001	hit
4	lh b	4	0900C0A102E8	00010111	hit
	sh a	36	0100C0A103E8	00011111	miss
	lh b	4	0900C0A102E8	00010111	hit
	sh a	1996	0100C0A14128	00001001	hit
5	lh b	5	0900C0A102F0	00010111	hit
	sh a	45	0100C0A10430	00100001	miss
	lh b	5	0900C0A102F0	00010111	hit
	sh a	1995	0100C0A14120	00001001	hit
6	lh b	6	0900C0A102F8	00010111	hit
	sh a	54	0100C0A10478	00100011	miss
	lh b	6	0900C0A102F8	00010111	hit
	sh a	1994	0100C0A14118	00001000	miss
7	lh b	7	0900C0A10300	00011000	miss
	sh a	63	0100C0A104C0	00100110	miss
	lh b	7	0900C0A10300	00011000	hit
	sh a	1993	0100C0A14110	00001000	hit
8	lh b	8	0900C0A10308	00011000	hit
	sh a	72	0100C0A10508	00101000	miss
	lh b	8	0900C0A10308	00011000	hit
	sh a	1992	0100C0A14108	00001000	hit
9	lh b	9	0900C0A10310	00011000	hit
	sh a	81	0100C0A10550	00101010	miss
	lh b	9	0900C0A10310	00011000	hit
	sh a	1991	0100C0A14100	00001000	hit



③ Since the one element of the array is 16bit=2byte, the memory address to access is base address+2*(array index #), and from the memory address. The memory address field did not change from the case ①, which is 31 tag bits, 10 index bits, and 5 offset value. Therefore, I got cache block address of each instruction, [14:5], which is index bit.

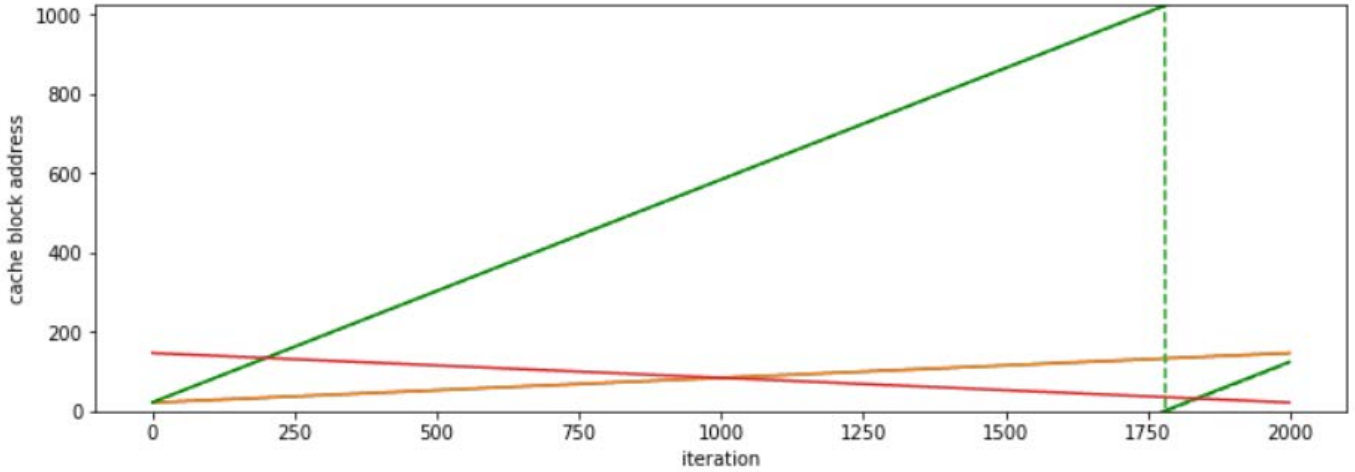
There is 4 memory access in an iteration, there is 40 memory accesses are made for the code execution for the first 10 iteration. The number of cache misses during this iteration is 12.

Since cache block address is [14:5] of memory address, I could get the patterns how cache block addresses of each instruction in an iteration, and those can be written as follows. The real index values are the integer value of them. B1, B2 is at the memory access at the load instructions which is for b array, A1 is at the first memory access at the save instruction, and A2 is the second and last memory access by save instruction in the iteration.

A1	1	(index)=22.25+9/16*x
	2	(index)=0.0625+9/16*(x-1781)
A2	(index)=147.25-x/16	
B1, b2	(index)=22.25+x/16	

Those can be plotted like the figure at the following page. Green plots are for A1, red plot is for A2, and orange plot is for B1 and B2, which has same value in same iteration. A1 has 2 cycles which have different tag value each. Since the slope of index value of A1 is 9/16, the index value changes about 9 during 16 cycles. Therefore, there is about 2000*9/16=1125 misses at the first save instruction, A1. For the index value of A2, before the plot meets B1/B2 plot, we only have to count the case which A1 not effected A2. Since the slope of A2 is 9.16 and A2 is 1/16, the number of misses during the range is (147-84)/16+(84-22)+6=72, 84 is value where plot A2 and B1/B2 plot meets, 147 and 22 are the first and last value in A2(index value at i=1999 iteration), and 6 is the miss made where the plot A2 and B1/B2 meet. Miss at B1 plot is (147-22)+1+6 = 132, 147 and 22 are the first and last value in B1, 1 ist miss at the first iteration, and 6 is number of misses at where the plot B1 and A2 meets. Misses at B2 plot is 2, which is in the first and second iteration. Therefore, the number of misses during the whole iteration is about 1331, then the miss rate is about 16.64%.

i	Instruction		Memory Address	cache block address	Hit /Miss
	type	#			
0	load b	0	0900C0A102C8	0000010110	miss
	save a	0	0100C0A102C8	0000010110	miss
	load b	0	0900C0A102C8	0000010110	miss
	save a	2000	0100C0A11268	0010010011	miss
1	load b	1	0900C0A102CA	0000010110	hit
	save a	9	0100C0A102DA	0000010110	miss
	load b	1	0900C0A102CA	0000010110	miss
	save a	1999	0100C0A11266	0010010011	hit
2	load b	2	0900C0A102CC	0000010110	hit
	save a	18	0100C0A102EC	0000010111	miss
	load b	2	0900C0A102CC	0000010110	hit
	save a	1998	0100C0A11264	0010010011	hit
3	load b	3	0900C0A102CE	0000010110	hit
	save a	27	0100C0A102FE	0000010111	hit
	load b	3	0900C0A102CE	0000010110	hit
	save a	1997	0100C0A11262	0010010011	hit
4	load b	4	0900C0A102D0	0000010110	hit
	save a	36	0100C0A10310	0000011000	miss
	load b	4	0900C0A102D0	0000010110	hit
	save a	1996	0100C0A11260	0010010011	hit
5	load b	5	0900C0A102D2	0000010110	hit
	save a	45	0100C0A10322	0000011001	miss
	load b	5	0900C0A102D2	0000010110	hit
	save a	1995	0100C0A1125E	0010010010	miss
6	load b	6	0900C0A102D4	0000010110	hit
	save a	54	0100C0A10334	0000011001	hit
	load b	6	0900C0A102D4	0000010110	hit
	save a	1994	0100C0A1125C	0010010010	hit
7	load b	7	0900C0A102D6	0000010110	hit
	save a	63	0100C0A10346	0000011010	miss
	load b	7	0900C0A102D6	0000010110	hit
	save a	1993	0100C0A1125A	0010010010	hit
8	load b	8	0900C0A102D8	0000010110	hit
	save a	72	0100C0A10358	0000011010	hit
	load b	8	0900C0A102D8	0000010110	hit
	save a	1992	0100C0A11258	0010010010	hit
9	load b	9	0900C0A102DA	0000010110	hit
	save a	81	0100C0A1036A	0000011011	miss
	load b	9	0900C0A102DA	0000010110	hit
	save a	1991	0100C0A11256	0010010010	hit



④ The base addresses for the address and memory addresses for whole memory access instruction are same as the previous case ③. However, the memory address field configuration is same as case ②, which has 33bit tag bits, 8bit index bits, and 5bit offset, since the cache is 4-way associated. Since cache block address is the index part of memory address, it has also 8 bits. Therefore, I got the cache block address.

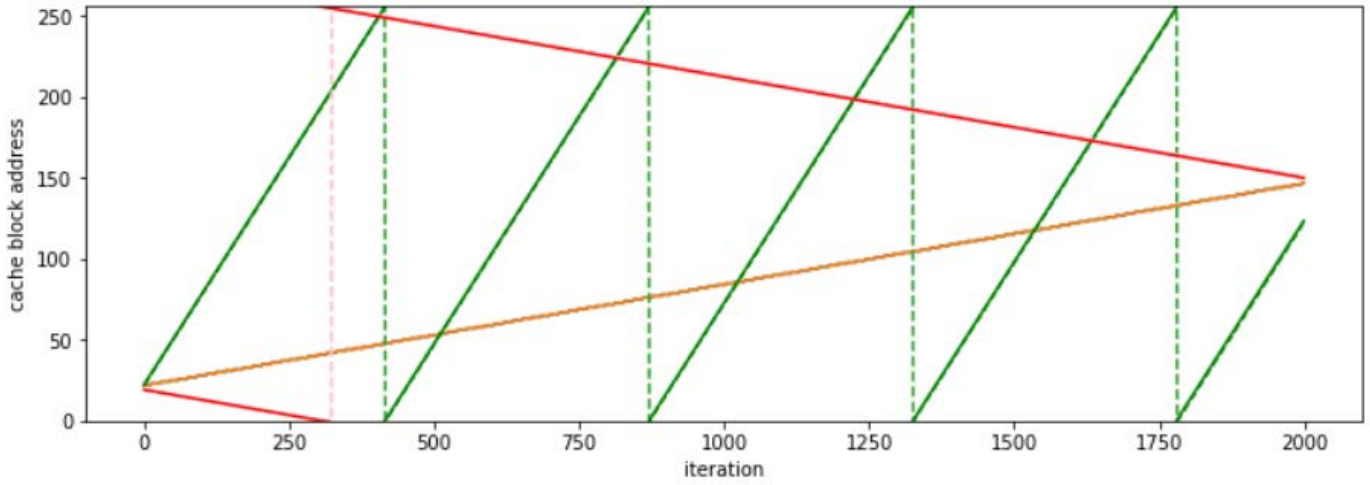
There are 4 memory access instruction in one iteration, there are all 40 memory accesses made during the first 10 iteration. During this iteration, 16 cache misses occurred.

For the cache miss rate for whole 2000 times of iteration, I also found patterns how the cache block address changes. Cache block address is [12:5] of memory address, and the memory address increases regularly according to the given code. Therefore, its approximation can be easily expressed in a equation like follows.

A1	1	(index)=22.5+9/16*x
	2	(index)=0.3125+9/16*(x-416)
	3	(index)=0.3125+9/16*(x-871)
	4	(index)=0.3125+9/16*(x-1326)
	5	(index)=0.3125+9/16*(x-1780)
A2	1	(index)=19.25-x/16
	2	(index)=255.75-(x-309)/16
B1, B2	(index)=19.25-x/16	

I also plotted them in the figure on the following page. Green plot, A1 is index equation for the first sh instruction which access to the memory of array a, and the red plot A2 is index equation for the second sh instruction which also access to the array a, and orange plot, which is B1 and B2 are for lh instructions which access to array b. Like the previous case, we can add all the cache misses that is related to each plot and subtract the number of double counted cache misses. For A1, the sum of index value change at each cycle is 2000*9/16=1125. For A2, there is 19+(256-151)=124 cache misses, which 19 is the starting value in the first cycle of A2, and 256 and 151 are first and last value in the second cycle of A2. For B1, 147-22=125, which is the range of B1 which has only one cycle. Since B2 has same value with B1 in a cycle, B2 has no cache misses. However, since A1 and A2 shares the same tag for the whole A2 range, and the plot shows that the data in the cache would not be deleted and could be used again, there are 125 of doubled counted cache misses. Therefore, the number of cache misses occurred in the whole iteration is about 1150+124+125-124=1250. Then the miss rate would be about 1250/8000*100=15.65%.

i	Instruction		Memory Address	Cache block address	Hit /Miss
	type	#			
0	lh b	0	0900C0A102C8	00010110	miss
	sh a	0	0100C0A11268	00010110	miss
	lh b	0	0900C0A102CA	00010110	hit
	sh a	2000	0100C0A102DA	00001010	miss
1	lh b	1	0900C0A102CA	00010110	hit
	sh a	9	0100C0A11266	00011000	miss
	lh b	1	0900C0A102CC	00010110	hit
	sh a	1999	0100C0A102EC	00001010	hit
2	lh b	2	0900C0A102CC	00010110	hit
	sh a	18	0100C0A11264	00011010	miss
	lh b	2	0900C0A102CE	00010110	hit
	sh a	1998	0100C0A102FE	00001001	miss
3	lh b	3	0900C0A102CE	00010111	miss
	sh a	27	0100C0A11262	00011101	miss
	lh b	3	0900C0A102D0	00010111	hit
	sh a	1997	0100C0A10310	00001001	hit
4	lh b	4	0900C0A102D0	00010111	hit
	sh a	36	0100C0A11260	00011111	miss
	lh b	4	0900C0A102D2	00010111	hit
	sh a	1996	0100C0A10322	00001001	hit
5	lh b	5	0900C0A102D2	00010111	hit
	sh a	45	0100C0A1125E	00100001	miss
	lh b	5	0900C0A102D4	00010111	hit
	sh a	1995	0100C0A10334	00001001	hit
6	lh b	6	0900C0A102D4	00010111	hit
	sh a	54	0100C0A1125C	00100011	miss
	lh b	6	0900C0A102D6	00010111	hit
	sh a	1994	0100C0A10346	00001000	miss
7	lh b	7	0900C0A102D6	00011000	miss
	sh a	63	0100C0A1125A	00100110	miss
	lh b	7	0900C0A102D8	00011000	hit
	sh a	1993	0100C0A10358	00001000	hit
8	lh b	8	0900C0A102D8	00011000	hit
	sh a	72	0100C0A11258	00101000	miss
	lh b	8	0900C0A102DA	00011000	hit
	sh a	1992	0100C0A1036A	00001000	hit
9	lh b	9	0900C0A102DA	00011000	hit
	sh a	81	0100C0A11256	00101010	miss
	lh b	9	000000000000	00011000	hit
	sh a	1991	000000000000	00001000	hit



Q2) Intel's 8th Generation Coffee Lake processors

To get the whole cache size for a single core, length of tag should be calculated from the memory address field.

A block has 16 words, which is 2^4, so the word offset is 4 bits. Also, a word is 32bit length, which is 4=2^2 byte. Therefore, the byte offset is 2bit long. Since memory address has tag, index, and offset part. Therefore, we can learn tag part is 64-6-2-4=52bits. Then the size of a cache set is (valid bit+tag bits+data)*(# of blocks in a set) = (1+52+(16 words * 32bits))* 8 blocks=4520 bits. Then the whole cache size is (size of a cache set) * (number of sets)=4520*64=289280bits=37170 byte=35.3125KB(KiB), which is not 35KB. The reason that the whole size of the cache is different from 32KB is that cache size usually means capacity of data that the cache can save.

Q3) Performance of Multilevel Cache

To get the total CPI of the processor, expected value of miss penalty should be added to the base CPI. In the case there is only L1 cache, miss penalty to the main memory would be (memory access time)/(1/CPU clock rate)=100ns/(1/4GHz)=100*10^(-9)*4*10^9=400 clock cycles. Therefore, the total CPI with only L1 would be 1.0+0.02*400=9.0. If there is a secondary cache L2, miss penalty to L2 would be 5ns/(1/4GHz)=20 clock cycles. Therefore, the total CPI with both L1 and L2 would be 1+0.02*20+0.005*400=1.0+0.4+2.0=3.4. Then the processor with a secondary cache would be faster about 9.0-3.4=5.6 total CPI.