----------------------------------CS2610

Assignment #4

-Cs20b072

Cs20b040

**Problem Statement**: Reverse Engineering of L1 Cache Memory.

**Objective:** To identify the cache block size and the associativity of the L1 cache.

**Procedure**

functions used are:

* rdtsc() [read time-stamp counter]:- It is used to determine how many ticks took place since processor was reset.
* \_mm\_mfence() : Its used to inform the compiler of the requirement of pending reads or writes not to be moved before or after the specified fence statement.
* \_mm\_clflush(data): It flushes the cache line.

**1. Finding block size:**

First we declared a latency array ‘lat[N]’, to store latency for each load instruction and a character array ‘arr[N]’ in order to generate load instructions with the help of a temporary character variable ‘temp’. Two unsigned long long integers t1 and t2 are declared which are used for calculating latency.

After flushing the cache line we access memory for each element in the array and record the access latency for each memory access. To calculate this latency we read the time stamp before the load function (which is temp = arr[i]) and store it in t1 and then read time stamp after load function and store it in t2. Then we subtract t1 from t2 in order to get latency. Once we have calculated the latency we add its value in the lat[N] array.

Then we carry out the same process 50 times to remove any noise, after which we can print the latency values to check for the cache miss.

Upon printing we get the following first 100 values;

0 515 // cold miss

1 75

2 76

3 79

4 80

5 77

6 92

7 79

8 75

9 82

10 75

11 94

12 76

13 76

14 76

15 75

16 236 // miss : 16

17 89

18 77

19 82

20 76

21 76

22 75

23 75

24 75

25 75

26 75

27 78

28 76

29 76

30 76

31 75

32 76

33 76

34 76

35 75

36 75

37 76

38 75

39 75

40 75

41 75

42 75

43 76

44 75

45 75

46 75

47 75

48 75

49 75

50 75

51 75

52 75

53 75

54 75

55 75

56 75

57 75

58 76

59 75

60 75

61 75

62 75

63 75

64 75

65 75

66 82

67 78

68 75

69 75

70 75

71 75

72 75

73 76

74 75

75 75

76 75

77 75

78 75

79 75

80 326 // miss : 80 – 16 = 64

81 78

82 81

83 75

84 76

85 75

86 75

87 85

88 76

89 76

90 77

91 75

92 76

93 75

94 75

95 75

96 75

97 75

98 76

99 76

100 75

Hence we get the difference between two cache miss = 64.

Thus block size = 64.

**2. Finding Associativity:**

Now since we have block size we can find associativity.

We can assume our associativity to be less than 17.

Since L1 cache size is 64KB we access array values with separated by 64KB. We expect our latency behaviour to be like less, less, less, less, high,high…….,high. The number of lows tell us associativity which is because we keep replacing blocks of same set. Then we access it in reverse order , that way we use already existing data in the cache to create lows and then creating the highs.

We get values as:

1. 130
2. 74
3. 73
4. 75
5. 74
6. 75
7. 76
8. 78
9. 79
10. 82 //from here we get all equal values
11. 83 //hence associativity is 9-1 = 8.
12. 82
13. 83
14. 83
15. 83
16. 83
17. 83
18. 83

Hence block size = 64B and associativity = 8.

