Brandon Cortez

CprE 381

HW 4

11/13/2020

1. Principle of Locality
    a. A program with little temporal or spatial locality with regard to data access would not often access a memory address more than once or access nearby addresses.
        i. Store $3 into memory address 12

            Load from memory address 48 into $10

            Store $25 into memory address 112
    b. A program with high temporal and low spatial locality with regard to data access would often access a memory address more than once but not access nearby addresses often.
        i. Store $3 into memory address 36

            Load from memory address 36 into $10

            Store $25 into memory address 36
    c. A program with low temporal and high spatial locality with regard to data access would not often access a memory address more than once but access nearby addresses often.
        i. Store $3 into memory address 24

            Load from memory address 28 into $10

            Store $25 into memory address 20
    d. A program with little temporal or spatial locality with regard to instruction access would not often reuse instructions and would operate on many different registers as well as jump to far away addresses.

i. Store $3 into memory address 12

Add $11, $20, $31

JR $2

e. A program with high temporal and low spatial locality with regard to instruction access would often reuse instructions but would operate on many different registers as well as jump to far away addresses.

i. Store $14 into memory address 36

Store $9 into memory address 20

Store $25 into memory address 8

JR $2

f. A program with low temporal and high spatial locality with regard to instruction access would not often reuse instructions and would not operate on many different registers as well as not jump to far away addresses.

i. Store $3 into memory address 12

Add $4, $3, $3

JR $4

2. Cache Configuration and Performance

a.

2	3	11	16	21	13	64	48	19	11	3	22	4	27	6	11
M	M	M	M	M	M	M	M	M	H	M	M	M	M	M	M

b.

0 - 0000	48
1 - 0001	
2 - 0010	2
3 - 0011	3

4 - 0100	4
5 - 0101	21
6 - 0110	6
7 - 0111	
8 - 1000	
9 - 1001	
10 - 1010	
11 - 1011	11
12 - 1100	
13 - 1101	13
14 - 1110	
15 - 1111	

c. The reference string

   i. 0x00000000

      0x00000020

      0x00000040

      0x00000041

   Gives 4 misses for C1 and 3 misses for C2

   C1 miss cycles = 4*8 = 32

   C2 miss cycles = 3*11 = 33

d. Processor comparison

   i = number of instructions

   CPI = 2

   Cache 1: Miss penalty = 6 + 1 = 7

   Inst Miss Cycles = .04 * 7 * i = .28 * i

   Data Miss Cycles = .06 * 7 * .5 * i = .49 * i

Total cycles on cache miss = 2.49 cycles

Cache 2: Miss penalty = 6 + 4 = 10

Inst Miss Cycles = .02 * 10 * i = .2 * i

Data Miss Cycles = .04 * 10 * .5 * i = .2 * i

Total cycles on cache miss = 2.40 cycles

Cache 3: Miss penalty = 6 + 4 = 10

Inst Miss Cycles = .02 * 10 * i = .2 * i

Data Miss Cycles = .03 * 10 * .5 * i = .15 * i

Total cycles on cache miss = 2.35 cycles

Cache 1 spends the most cycles on cache misses.

3. Breaking Locality
    a. Row-major ordering time: 4.456053s
    b. Column-major ordering time: 5.858528s

    Row major is faster as the addresses being read are closer together in the same row so there is high spacial locality while column major jumps addresses to the next address in the same column giving it low spatial locality.