Name: Cicelia Siu

CS 219 – Assignment #7

Purpose: Become familiar with MIPS cache implementation

Points: 100

**Reading/References:**

Chapter 5

**Assignment:**

Answer the following questions:

1) Describe the key features of a program that would exhibit the following traits with regard to data accesses: [12 pts, 3 pts each]

1. low temporal locality and low spatial locality - very high miss ratio
2. high temporal locality - high cache size means miss ratio is lower
3. low spatial locality - higher miss ratio, brings less data around it
4. high spatial locality - lower miss ratio, brings more data around it

2) Describe the key features of a program that would exhibit the following traits with regard to

instruction fetches: [12 pts, 3 pts each]

1. low temporal locality - no loops, more jump statements
2. high temporal locality - programs with many uses of loops (reused data)
3. low spatial locality - many jumps
4. high spatial locality -instructions normally accessed sequentially (elements of arrays)

3) A new processor can use either a write-through or write-back cache, selectable through software. [8 pts, 4 pts each]

1. Assume the processor will run data-intensive applications with a large number of load and store operations. Explain which cache write policy should be used.

**Write-back. With a large number of load and store operations, write-through might be too slow.**

1. Consider the same question but this time for a safety-critical system in which data integrity is more important than memory performance. Explain which cache write policy should be used.

**Write-through. Data should be consistent in write-throughs, which is good for this application with data integrity.**

4) Here is a series of address references given as words addresses:

1, 2, 3, 4, 11, 16, 21, 13, 64, 48, 19, 11, 3, 22, 4, 27, 6, 4, 5, and 11.

1. Assuming a direct-mapped cache with 8 one-word blocks that is initially empty, label each reference in the list as a hit or a miss and show the contents of the cache (including previous, over-written values). You do not need to show the tag field. When done, include the hit ratio. [3 pts]

1 mod 8 = 1 1 = 000000012

2 mod 8 = 2 2 = 000000102

3 mod 8 = 3 3 = 000000112

4 mod 8 = 4 4 = 000001002

11 mod 8 = 3 11 = 000010112

16 mod 8 = 0 16 = 000100002

21 mod 8 = 5 21 = 000101012

13 mod 8 = 5 13 = 000011012

64 mod 8 = 0 64 = 010000002

48 mod 8 = 0 48 = 011000002

19 mod 8 = 4 19 = 000100112

11 mod 8 = 3 11 = 000010112

3 mod 8 = 3 3 = 000000112

22 mod 8 = 6 22 = 00010110

4 mod 8 = 4 4 = 000001002 (hit)

27 mod 8 = 3 27 = 000110112

6 mod 8 = 6 6 = 000001102

4 mod 8 = 4 4 = 000001002 (hit)

5 mod 8 = 5 5 = 000001012

11 mod 8 = 3 11 = 000010112

**hit ratio: 2 hits out of 20 = 2/20 = 1/10 = 10%**

|  |  |  |
| --- | --- | --- |
| Cache Set | v | Address |
| 000 | 1 | 16->64->48 |
| 001 | 1 | 1 |
| 010 | 1 | 2 |
| 011 | 1 | 3->11->19->11->3->27->11 |
| 100 | 1 | 4 |
| 101 | 1 | 21->13->5 |
| 110 | 1 | 22->6 |
| 111 | 0 |  |

1. Show the hits and misses and cache contents (including previous, overwritten values) for a direct-mapped cache with four-word blocks and a total size of 8 words. You do not need to show the tag field. When done, include the hit ratio. [3 pts]

1 = 000000012

2 = 000000102 (hit)

3 = 000000112 (hit)

4 = 000001002

11 = 000010112

16 = 000100002

21 = 000101012

13 = 000011012

64 = 010000002

48 = 011000002

19 = 000100112

11 = 000010112

3 = 000000112

22 = 00010110

4 = 000001002

27 = 000110112

6= 000001102 (hit)

4 = 000001002 (hit)

5 = 000001012 (hit)

11 = 000010112

hits: (2,3),(6,4,5)

**hit rate = 5/20 = ¼ = 25%**

|  |  |  |
| --- | --- | --- |
| Cache Set | v | Address |
| 00 | 1 | [0,1,2,3]->  [8,9,10,11]->  [16,17,18,19]->  [64,65,66,67]->  [48,49,50,51]->  [16,17,18,19]->  [8,9,10,11]->  [0,1,2,3]->  [24,25,26,27]->  [8,9,10,11] |
| 01 | 1 | [4,5,6,7]->  [20,21,22,23]->  [12,13,14,15]->  [20,21,22,23]->  [4,5,6,7] |

5) Here is a series of address references given as words addresses:

3, 7, 10, 13, 64, 48, 19, 2, 3, 11, 16, 21, 11, 3, 22, 4, 27, 6, 12, 3, 16, 17, 44, 21, and 12.

1. Assuming a direct-mapped cache with 16 one-word blocks that is initially empty, label each reference in the list as a hit or a miss and show the contents of the cache (including previous, overwritten values). You do not need to show the tag field. When done, include the hit ratio. [7 pts]

3 = 000000112

7 = 000001112

10 = 000010102

13 = 000011012

64 = 010000002

48 = 011000002

19 = 000100112

2 = 000000102

3 = 000000112

11 = 000010112

16 = 000100002

21 = 000101012

11 = 000010112 (hit)

3 = 000000112 (hit)

22 = 000101102

4 = 000001002

27 = 000110112

6 = 000001102

12 = 000011002

3 = 000000112 (hit)

16 = 000100002 (hit)

17 = 000100012

44 = 001011002

21 = 000101012 (hit)

12 = 000011002

**hit ratio: 5 hits out of 25 = 1/5**

|  |  |  |
| --- | --- | --- |
| Cache Set | v | Address |
| 0000 | 1 | 64->48->16 |
| 0001 | 1 | 17 |
| 0010 | 1 | 2 |
| 0011 | 1 | 3->19 ->3 |
| 0100 | 1 | 4 |
| 0101 | 1 | 21 |
| 0110 | 1 | 22->6 |
| 0111 | 1 | 7 |
| 1000 | 0 |  |
| 1001 | 0 |  |
| 1010 | 1 | 10 |
| 1011 | 1 | 11->27 |
| 1100 | 1 | 12->44->12 |
| 1101 | 1 | 13 |
| 1110 | 0 |  |
| 1111 | 0 |  |

1. Show the hits and misses and cache contents (including previous, overwritten values) for a direct-mapped cache with four-word blocks and a total size of 16 words. You do not need to show the tag field. When done, include the hit ratio. [7 pts]

3 = 000000112

7 = 000001112

10 = 000010102

13 = 000011012

64 = 010000002

48 = 011000002

19 = 000100112

2 = 000000102

3 = 000000112 (hit)

11 = 000010112 (hit)

16 = 000100002

21 = 000101012

11 = 000010112 (hit)

3 = 000000112

22 = 000101102 (hit)

4 = 000001002

27 = 000110112

6 = 000001102

12 = 000011002

3 = 000000112

16 = 000100002

17 = 000100012 (hit)

44 = 001011002

21 = 000101012 (hit)

12 = 000011002

**hit ratio = 6 hits out of 25 = 6/25**

|  |  |  |
| --- | --- | --- |
| Cache Set | v | Address |
| 00 | 1 | [64,65,66,67->]  [48,49,50,51]->  [12,13,14,15] ->  [16,17,18,19]->  [44,45,46,47]->  [12,13,14,15] |
| 01 |  | [12,13,14,15] ->  [20,21,22,23] |
| 10 |  | [8,9,10,11]->  [4,5,6,7] |
| 11 |  | [0,1,2,3,]->  [4,5,6,7]->  [16,17,18,19]->  [0,1,2,3,]->  [16,17,18,19]->  [0,1,2,3,]->  [4,5,6,7]->  [24,25,26,27]->  [0,1,2,3,]-> |

6) Here is a series of address references given as words addresses:

5, 18, 1, 2, 3, 11, 16, 21, 8, 9, 13, 19, 11, 3, 10, 22, 4, 27, 6, 24, 16, 11, 49, 13, and 21.

1. Assuming a direct-mapped cache with 16 one-word blocks that is initially empty, label each reference in the list as a hit or a miss and show the contents of the cache (including previous, overwritten values). You do not need to show the tag field. When done, include the hit ratio. [9 pts]

5 = 000001012

18 = 000100102

1 = 000000012

2 = 000000102

3 = 000000112,

11 = 000010112

16 = 000100002

21 = 000101012

8 = 000010002

9 = 000010012

13 = 000011012

19 = 000100112

11= 000010112 (hit)

3 = 000000112

10 = 000010102

22 = 000101102

4 = 000001002

27 = 000110112

6 = 000001102

24 = 000110002

16 = 000100002 (hit)

11 = 000010112

49 = 001100012

13 = 000011012 (hit)

21 = 000101012 (hit)

hit ratio : 4 hits of 25 = 4/25

|  |  |  |
| --- | --- | --- |
| Cache Set | v | Address |
| 0000 | 1 | 16 |
| 0001 | 1 | 1->39 |
| 0010 | 1 | 18->2 |
| 0011 | 1 | 3 ->19->3 |
| 0100 | 1 | 4 |
| 0101 | 1 | 5 ->21 |
| 0110 | 1 | 22 ->6 |
| 0111 | 0 |  |
| 1000 | 1 | 8->24 |
| 1001 | 1 | 9 |
| 1010 | 1 | 10 |
| 1011 | 1 | 11 ->27->11 |
| 1100 | 0 |  |
| 1101 | 1 | 13 |
| 1110 | 0 |  |
| 1111 | 0 |  |

1. Show the hits and misses and cache contents (including previous, overwritten values) for a direct-mapped cache with four-word blocks and a total size of 16 words. You do not need to show the tag field. When done, include the hit ratio. [9 pts]

5 = 000001012

18 = 000100102

1 = 000000012

2 = 000000102 (hit)

3 = 000000112, (hit)

11 = 000010112

16 = 000100002 (hit)

21 = 000101012

8 = 000010002 (hit)

9 = 000010012 (hit)

13 = 000011012

19 = 000100112 (hit)

11= 000010112 (hit)

3 = 000000112

10 = 000010102

22 = 000101102

4 = 000001002

27 = 000110112

6 = 000001102 (hit)

24 = 000110002 (hit)

16 = 000100002

11 = 000010112

49 = 001100012

13 = 000011012

21 = 000101012 (hit)

hit ratio: 10/25 = 40%

|  |  |  |
| --- | --- | --- |
| Cache Set | v | Address |
| 00 | 1 | [4,5,6,7] -> [16,17,18,19] |
| 01 | 1 | [4,5,6,7]->[0,1,2,3] ->[20,21,22,23] ->[12,13,14,15]->[48,49,50,51] ->[12,13,14,15] |
| 10 | 1 | [16,17,18,19]->[8,9,10,11]-> ->[20,21,22,23] |
| 11 | 1 | [8,9,10,11] ->[0,1,2,3]-> [24,25,26,27]-> [8,9,10,11] |

7) Here is a series of address references given as words addresses:

7, 8, 2, 3, 11, 16, 21, 13, 64, 48, 19, 11, 3, 22, 4, 27, 6, 7, 3, and 11.

Show the hits and misses and cache contents (including previous, over-written values) for a

two-way set-associative cache with one-word blocks and a total size of 16 words. You do not

need to show the tag field. Assume LRU replacement. When done, include the hit ratio. [9 pts]

**7, 8, 2, 3, 11, 16, 21, 13, 64, 48, 19, 11(hit), 3, 22, 4, 27, 6, 7(hit), 3(hit), and 11.**

**all the rest are miss**

**hit ratio: 3/20 = 15%**

Block 0 Block 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Set # | v | Address |  | Set # | v | Address |
| 000 | 1 | 8->64 |  | 000 | 1 | 16->48 |
| 001 | 0 |  |  | 001 | 0 |  |
| 010 | 1 | 2 |  | 010 | 0 |  |
| 011 | 1 | 3->19->3 |  | 011 | 1 | 11->27->11 |
| 100 | 1 | 4 |  | 100 | 0 |  |
| 101 | 1 | 21 |  | 101 | 1 | 13 |
| 110 | 1 | 22 |  | 110 | 1 | 6 |
| 111 | 1 | 7 |  | 111 | 0 |  |

8) Here is a series of address references given as words addresses:

13, 64, 48, 19, 2, 3, 11, 16, 21, 11, 3, 22, 4, 27, 6, 12, 4, 7,8, and 24.

Show the hits and misses and cache contents (including previous, over-written values) for a

two-way set-associative cache with one-word blocks and a total size of 16 words. You do not

need to show the tag field. Assume LRU replacement. When done, include the hit ratio. [9 pts]

**13, 64, 48, 19, 2, 3, 11, 16, 21, 11(hit), 3(hit), 22, 4, 27, 6, 12, 4(hit), 7,8, and 24**

**hit ratio: 3/20 = 15%**

Block 0 Block 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Set # | v | Address |  | Set # | v | Address |
| 000 | 1 | 64->16->24 |  | 000 | 1 | 48->8 |
| 001 | 0 |  |  | 001 | 0 |  |
| 010 | 1 | 2 |  | 010 | 0 |  |
| 011 | 1 | 19->11 |  | 011 | 1 | 3->27 |
| 100 | 1 | 4 |  | 100 | 1 | 12 |
| 101 | 1 | 13 |  | 101 | 1 | 21 |
| 110 | 1 | 22 |  | 110 | 1 | 6 |
| 111 | 1 | 7 |  | 111 | 0 |  |

9) As discussed in class, associativity typically improves the miss ratio. However, this is not

always true. Provide an explanation or an example of a short series of address references for

which a two-way set-associative cache with LRU replacement would experience more misses

that a direct-mapped cache of the same size. [6 pts]

caches must be same overall space.

32 entry direct mapped cache vs 16 entry 2-way set associative

1,17,33, 1,17, 33, 1,17,33,1,17,33, 1,17,33, 1,17,33,

Direct

|  |
| --- |
| 0 - |
| 1index - 1->33->1->33->1->33->1->33->1->33->1->33 |
| ... |
| 17 index - 17 |
| ... |
| 31 |

Direct gives about a 33% hit ratio (in general). In this case a 27%

2 way set associative

|  |  |
| --- | --- |
| 0- | 0- |
| 1 index- 1->33->17->1->33->17->1->33->17 | 1index - 17->1->33->17->1->33->17->1->33 |
| ... | ... |
| 15- | 15- |

2 way set associative gives about 0% hit ratio

10) Given a direct-mapped cache that will contain 32,768 bytes of data and four-word blocks

(where each word is 4 bytes) and assuming 32-bit addressing;

1. How many additional bits are required for the overhead associated with the 32,768 bytes of data?[4 pts]

4 words \*4 bytes = 16 bytes

32768 bytes / 16 bytes = 2048 blocks

32

|  |  |  |
| --- | --- | --- |
| tag | line | 0000 |

17 11 4

32 - 11-4 = 17 bits

18\*2048 = **36864 bits**

1. What is the overhead (additional space required) for the 32,768 bytes of data, expressed as a percentage? [2 pts]

Must show work for full credit.

32768 \* 8 =

262144

+36864

= 299008 bits (total)

36864 (overhead) / 299008 (total) = 0.12328 \*100 = **12.328%**