***E-CE UTH***

***CE232***

***Computer Organisation***

***& Design***

***Lab 8***

***Iliadis Ilias:2523***

***Koen Leonando:2534***

***Άσκηση 1) Προσομοίωση λειτουργίας μνήμης cache***

Αρχικά , μετατρέπουμε την δοθείσα ακολουθία διευθύνσεων μνήμης σε binary ώστε να αποκομίσουμε τις απαραίτητες πληροφορίες για την μνήμη cache:

Αφού η cache είναι 64 Bytes και 2 way-set associative με cache block 4 bytes και στρατηγική απελευθέρωσης LRU :

**Block Offset** = log2(bytes/block) = log2(4) = 2 bits

Cache bytes = (sets )\* (blocks /set) \* (bytes/block) <=> Sets = (Cache bytes) / (blocks/set) \* (bytes/block) = 64 / 2 \* 4 = 64 / 8 = 8 <=> **Sets** = 8 sets

**Set Index** = log2(sets) = log2(8)= 3 bits

**Tag** = (address bits) - (block offset bits) - (set index bits) = 10 – 2 – 3 = 5 MSB’s

Κάθε διεύθυνση αναπαρίσταται με **10 bits** επειδή μας δίνεται ότι η κύρια μνήμη έχει μέγεθος 512 bytes => log2(2^10) = 10 bits.

|  |  |
| --- | --- |
| **Address in decimal** | **Address in binary** |
| 207 | 00110|011|11 |
| 212 | 00110|101|00 |
| 416 | 01101|000|00 |
| 289 | 01001|000|01 |
| 204 | 00110|011|00 |
| 222 | 00110|111|10 |
| 461 | 01110|011|01 |
| 109 | 00011|011|01 |
| 280 | 01000|110|00 |
| 303 | 01001|011|11 |
| 2 | 00000|000|10 |
| 480 | 01111|000|00 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0110 | 11 | 1 | Mem[204] | Mem[205] | Mem[206] | Mem[207] |
| N |  |  |  |  |  |  |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 110 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 111 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| Addr: 207 , 00110|011|11 , Set Index : 011, MISS | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0110 | 11 | 1 | Mem[204] | Mem[205] | Mem[206] | Mem[207] |
| N |  |  |  |  |  |  |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 111 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| Addr: 212 , 00110|101|00, Set Index : 101, MISS | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | Y | 1101 | 00 | 1 | Mem[416] | Mem[417] | Mem[418] | Mem[419] |
| N |  |  |  |  |  |  |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0110 | 11 | 1 | Mem[204] | Mem[205] | Mem[206] | Mem[207] |
| N |  |  |  |  |  |  |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 111 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| Addr: 416 , 01101|000|00, Set Index : 000, MISS | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | Y | 1101 | 00 | 0 | Mem[416] | Mem[417] | Mem[418] | Mem[419] |
| Y | 1001 | 01 | Mem[288] | Mem[289] | Mem[290] | Mem[291] |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0110 | 11 | 1 | Mem[204] | Mem[205] | Mem[206] | Mem[207] |
| N |  |  |  |  |  |  |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 111 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| Addr: 289 ,01001|000|01 , Set Index : 000, MISS | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | Y | 1101 | 00 | 0 | Mem[416] | Mem[417] | Mem[418] | Mem[419] |
| Y | 1001 | 01 | Mem[288] | Mem[289] | Mem[290] | Mem[291] |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0110 | 00 | 1 | Mem[204] | Mem[205] | Mem[206] | Mem[207] |
| N |  |  |  |  |  |  |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 111 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| Addr: 204 ,00110|011|00 , Set Index : 011, HIT | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | Y | 1101 | 00 | 0 | Mem[416] | Mem[417] | Mem[418] | Mem[419] |
| Y | 1001 | 01 | Mem[288] | Mem[289] | Mem[290] | Mem[291] |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0110 | 00 | 1 | Mem[204] | Mem[205] | Mem[206] | Mem[207] |
| N |  |  |  |  |  |  |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 111 | Y | 0110 | 10 | 1 | Mem[220] | Mem[221] | Mem[222] | Mem[223] |
| N |  |  |  |  |  |  |
| Addr: 222 ,00110|111|10 , Set Index : 111, MISS | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | Y | 1101 | 00 | 0 | Mem[416] | Mem[417] | Mem[418] | Mem[419] |
| Y | 1001 | 01 | Mem[288] | Mem[289] | Mem[290] | Mem[291] |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0110 | 00 | 0 | Mem[204] | Mem[205] | Mem[206] | Mem[207] |
| Y | 0011 | 01 | Mem[460] | Mem[461] | Mem[462] | Mem[463] |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 111 | Y | 0110 | 10 | 1 | Mem[220] | Mem[221] | Mem[222] | Mem[223] |
| N |  |  |  |  |  |  |
| Addr: 461 , 01110|011|01, Set Index : 011, MISS | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | Y | 1101 | 00 | 0 | Mem[416] | Mem[417] | Mem[418] | Mem[419] |
| Y | 1001 | 01 | Mem[288] | Mem[289] | Mem[290] | Mem[291] |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0011 | 01 | 1 | Mem[108] | Mem[109] | Mem[110] | Mem[111] |
| Y | 0011 | 01 | Mem[460] | Mem[461] | Mem[462] | Mem[463] |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 111 | Y | 0110 | 10 | 1 | Mem[220] | Mem[221] | Mem[222] | Mem[223] |
| N |  |  |  |  |  |  |
| Addr: 109 , 00011|011|01, Set Index : 011, MISS | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | Y | 1101 | 00 | 0 | Mem[416] | Mem[417] | Mem[418] | Mem[419] |
| Y | 1001 | 01 | Mem[288] | Mem[289] | Mem[290] | Mem[291] |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0011 | 01 | 1 | Mem[108] | Mem[109] | Mem[110] | Mem[111] |
| Y | 0011 | 01 | Mem[460] | Mem[461] | Mem[462] | Mem[463] |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | Y | 1000 | 00 | 1 | Mem[280] | Mem[281] | Mem[282] | Mem[283] |
| N |  |  |  |  |  |  |
| 111 | Y | 0110 | 10 | 1 | Mem[220] | Mem[221] | Mem[222] | Mem[223] |
| N |  |  |  |  |  |  |
| Addr: 280 , 01000|110|00 , Set Index : 110, MISS | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | Y | 1101 | 00 | 0 | Mem[416] | Mem[417] | Mem[418] | Mem[419] |
| Y | 1001 | 01 | Mem[288] | Mem[289] | Mem[290] | Mem[291] |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0011 | 01 | 0 | Mem[108] | Mem[109] | Mem[110] | Mem[111] |
| Y | 1001 | 11 | Mem[300] | Mem[301] | Mem[302] | Mem[303] |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | Y | 1000 | 00 | 1 | Mem[280] | Mem[281] | Mem[282] | Mem[283] |
| N |  |  |  |  |  |  |
| 111 | Y | 0110 | 10 | 1 | Mem[220] | Mem[221] | Mem[222] | Mem[223] |
| N |  |  |  |  |  |  |
| Addr: 303 , 01001|011|11, Set Index : 011, MISS | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | Y | 0000 | 10 | 1 | Mem[0] | Mem[1] | Mem[2] | Mem[3] |
| Y | 1001 | 01 | Mem[288] | Mem[289] | Mem[290] | Mem[291] |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0011 | 01 | 0 | Mem[108] | Mem[109] | Mem[110] | Mem[111] |
| Y | 1001 | 11 | Mem[300] | Mem[301] | Mem[302] | Mem[303] |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | Y | 1000 | 00 | 1 | Mem[280] | Mem[281] | Mem[282] | Mem[283] |
| N |  |  |  |  |  |  |
| 111 | Y | 0110 | 10 | 1 | Mem[220] | Mem[221] | Mem[222] | Mem[223] |
| N |  |  |  |  |  |  |
| Addr: 2 ,00000|000|10, Set Index : 000, MISS | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set  Index | Valid  Bit | Tag | Block  Offset | RLU bit | BYTE:  00 | BYTE:  01 | BYTE:  10 | BYTE:  11 |
| 000 | Y | 0000 | 10 | 0 | Mem[0] | Mem[1] | Mem[2] | Mem[3] |
| Y | 0000 | 00 | Mem[480] | Mem[481] | Mem[482] | Mem[483] |
| 001 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 010 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 011 | Y | 0011 | 01 | 0 | Mem[108] | Mem[109] | Mem[110] | Mem[111] |
| Y | 1001 | 11 | Mem[300] | Mem[301] | Mem[302] | Mem[303] |
| 100 | N |  |  | 0 |  |  |  |  |
| N |  |  |  |  |  |  |
| 101 | Y | 0110 | 00 | 1 | Mem[212] | Mem[213] | Mem[214] | Mem[215] |
| N |  |  |  |  |  |  |
| 110 | Y | 1000 | 00 | 1 | Mem[280] | Mem[281] | Mem[282] | Mem[283] |
| N |  |  |  |  |  |  |
| 111 | Y | 0110 | 10 | 1 | Mem[220] | Mem[221] | Mem[222] | Mem[223] |
| N |  |  |  |  |  |  |
| Addr: 480 , 01111|000|00, Set Index : 000, MISS | | | | | | | | |

***Άσκηση 2) Χαρακτηριστικά της μνήμης Cache***

|  |  |  |
| --- | --- | --- |
| ***Ακολουθία #*** | ***Διευθύνσεις*** | ***Cache Hit Ratio*** |
| ***1*** | 0, 2, 4, 7, 32, 40 | ***33%*** |
| ***2*** | 0, 512, 1024, 1536, 2048, 1536, 1024, 512, 0, 1024 | ***10%*** |
| ***3*** | 0, 64, 128, 256, 512, 256, 128, 64, 0, 512 | ***30%*** |
| ***4*** | 0, 512, 1024, 0, 1536, 0, 2048, 0 | ***25%*** |

***Οι διευθύνσεις των ακολουθίων σε binary:***

***#1)***

|  |  |
| --- | --- |
| **0** | 000000 |
| ***2*** | 000010 |
| ***4*** | 000100 |
| ***7*** | 000111 |
| ***32*** | 100000 |
| ***40*** | 101000 |

***#2)***

|  |  |
| --- | --- |
| **0** | 000000000000 |
| ***512*** | 001000000000 |
| ***1024*** | 010000000000 |
| ***1536*** | 011000000000 |
| ***2048*** | 100000000000 |
| ***1536*** | 011000000000 |

***#3)***

|  |  |
| --- | --- |
| **0** | 0000000000 |
| ***64*** | 0001000000 |
| ***128*** | 0010000000 |
| ***256*** | 0100000000 |
| ***512*** | 1000000000 |

***#4)***

|  |  |
| --- | --- |
| **0** | 000000000000 |
| ***512*** | 001000000000 |
| ***1024*** | 010000000000 |
| ***1536*** | 011000000000 |
| ***2048*** | 100000000000 |

Αρχικά, για την πρώτη ακολουθία παρατηρούμε ότι το cache hit είναι ανεξάρτητο του Associativity διότι δεν εμφανίζεται καμία διεύθυνση παραπάνω από μία φορά (αν γινόταν αυτό θα μπορούσαμε να μειώσουμε τα misses με Associativity..) συνεπώς οποιαδήποτε υπόθεση κάνουμε για το Associativity δεν πρόκειται να επιφέρει αλλαγή στο cache ratio της εν λόγω ακολουθίας. Επομένως το μόνο που μπορούμε να ξεκαθαρήσουμε είναι το **μέγεθος του κάθε cache block.**

1) Αν το μέγεθος του cache block είναι 1 byte , τότε θα είχαμε *100% Miss Rate* διότι όλες οι διευθύνσεις είναι διαφορετικές.

2)Αν το μέγεθος του cache block είναι 2 byte, τότε πάλι θα είχαμε 100% *Miss Rate,* διότι μάζι με την εκάστοτε διεύθυνση θα φορτώναμε και την διπλανή του διεύθυνση (την πάνω η την κάτω αν το block offset είναι 1 ή 0 αντίστοιχα) η οπόια δεν θα είναι καμία από τις υπόλοιπες διευθύνσεις της ακολουθίας επειδή απέχουν περισσότερο από μια διεύθυνση μεταξύ τους.

3)Αν το μέγεθος του cache block είναι 4 byte, τότε το block offset ειναι 2 που σημάνει ότι κοιτάμε κάθε φορα τα 2 LSB της κάθε διεύθυνσης για να δούμε που ακριβώς θα τοποθετηθεί το byte που περιέχει η διεύθυνση για να προσθέσουμε και τα αλλα 3 byte από τις γειτονικές διευθύνσεις.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Διεύθυνση | Block Offset | Block byte: 00 | Block byte: 01 | Block byte: 10 | Block byte: 11 | HIT/MISS |
| 0 | 00 | Mem[0] | Mem[1] | Mem[2] | Mem[3] | MISS |
| 2 | 10 |  |  |  |  | HIT/στο block του 0 |
| 4 | 00 | Mem[4] | Mem[5] | Mem[6] | Mem[7] | MISS |
| 7 | 11 |  |  |  |  | HIT/στο block του 4 |
| 32 | 00 | Mem[32] | Mem[33] | Mem[34] | Mem[35] | MISS |
| 40 | 00 | Mem[40] | Mem[41] | Mem[42] | Mem[43] | MISS |

Συνολικά έχουμε 2/6 = 1/3 = 0.33% cache hit ratio το οποίο ισχύει γι αυτή την ακολουθία!

Ωστόσο πρέπει να εξετάσουμε και τις υπόλοιπες περιπτώσεις. Αν το μέγεθος του cache block είναι 8 byte, τότε το block offset θα είναι 3. Σε αυτή την περίπτωση όταν φορτώσουμε την διεύθυνση 0 θα έχουμε αρχικά MISS αλλά θα φορτώσουμε στο ίδιο block και τις διευθύνσεις : 1, 2, 3, 4, 5, 6, 7.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Addr | Block Offset | Block Byte: 000 | Block Byte: 001 | Block Byte: 010 | Block Byte: 011 | Block Byte: 100 | Block Byte: 101 | Block Byte: 110 | Block Byte: 111 |
| 000|000 | 000 | Mem[0] | Mem[1] | Mem[2] | Mem[3] | Mem[4] | Mem[5] | Mem[6] | Mem[7] |

Παρατηρούμε ότι όταν πάμε να φορτώσουμε τις διευθύνσεις 2, 4 και 7 θα έχουμε 3 διαδοχικά HIT το οποίο ξεπερνά το δεδομένο rate 0.33% (3/6 (μέχρι στιγμής) = 0.5% cache hit ratio) επομένως η περίπτωση αυτή απορρίπτεται όπως και οι περιπτώσεις με τα 16 και 32 bytes( θα φορτώσουμε περισσότερα απ’ ότι φορτώσαμε στο προηγούμενο παράδειγμα με αποτέλεσμα να έχουμε ακόμα περισσότερα HITS) Άρα ισχύει σίγουρα μέχρι στιγμής ότι **Cache block size = 4 bytes.**

Προχωράμε στην ανάλυση της ακολουθίας #2. Στην συγκεκριμένη ακολουθία το block size που υπολογίσαμε προηγουμένως μας είναι αδιάφορο διότι οι διευθύνσεις δεν πρόκειται να εχουν φορτωθεί από κάποια προηγούμενη διεύθηνση (μεγάλη απόσταση μεταξύ των διευθύνσεων >> 4 bytes). Συνεπώς είναι βολικό να αναφέρουμε απλα σε ποιο block θα μπεί η κάθε διεύθυνση. Παρατηρούμε ότι συναντάμε επανειλημμένες εμφανήσεις ορισμένων διευθύνσεων το οποίο μας παροτρύνει στον καθορισμό του Associativity της cache.

1) **’Εστω ότι η cache είναι 1 way associative με μέγεθος 256 bytes και block size = 4 bytes** (η πολιτικη αντικατάστασης στο 1 way associative δεν μας ενδιαφέρει)

Τότε τα sets θα είναι: 256/4 = 64 = 2^6, το οποίο ισοδυναμεί με μια direct – mapped cache.

Για να βρούμε σε ποιο set ανήκει η κάθε διεύθυνση κοιτάμε τα 6 bits μετα από τα 2 LSB του block offset.

|  |  |  |  |
| --- | --- | --- | --- |
| Διεύθυνση | Set | Data | MISS/HIT |
| 0 | 000000 | Mem[0] | MISS |
| 512 | 000000 | Mem[512] | MISS |
| 1024 | 000000 | Mem[1024] | MISS |
| 1536 | 000000 | Mem[1536] | MISS |
| 2048 | 000000 | Mem[2048] | MISS |
| 1536 | 000000 | Mem[1536] | MISS |
| 1024 | 000000 | Mem[1024] | MISS |
| 512 | 000000 | Mem[512] | MISS |
| 0 | 000000 | Mem[0] | MISS |
| 1024 | 000000 | Mem[1024] | MISS |

Έχουμε Cash Hit Rate = 0% το οποίο είναι λάθος , συνεπώς η περίπτωση αυτή απορρίπτεται!

2)**’Εστω ότι η cache είναι 2 way associative με μέγεθος 256 bytes και block size = 4 bytes και πολιτική αντικατάστασης LRU**

Στην προκειμένη περίπτωση τα sets είναι: 256/4(bytes per block)\*2(blocks per set) = **32 = 2^5**

Για να βρούμε σε ποιο set ανήκει η κάθε διεύθυνση κοιτάμε τα **5 bits** μετα από τα 2 LSB του block offset.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Διεύθυνση | Set | Set Block 0 | Set Block 1 | LRU bit | MISS/HIT |
| 0 | 00000 | Mem[0] | - | 1 | MISS |
| 512 | 00000 | Mem[0] | Mem[512] | 0 | MISS |
| 1024 | 00000 | Mem[1024] | Mem[512] | 1 | MISS |
| 1536 | 00000 | Mem[1024] | Mem[1536] | 0 | MISS |
| 2048 | 00000 | Mem[2048] | Mem[1536] | 1 | MISS |
| 1536 | 00000 | Mem[2048] | Mem[1536] | 0 | HIT |
| 1024 | 00000 | Mem[1024] | Mem[1536] | 1 | MISS |
| 512 | 00000 | Mem[1024] | Mem[512] | 0 | MISS |
| 0 | 00000 | Mem[0] | Mem[512] | 1 | MISS |
| 1024 | 00000 | Mem[0] | Mem[1024] | 0 | MISS |

Έχουμε Cash hit ratio 1/10 = **10%** το οποίο είναι σωστό! Ας εξετάσουμε την ίδια περίπτωση απλώς μ ε πολιτική αντικατάστασης FIFO(First In First Out).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Διεύθυνση | Set | Set Block 0 | Set Block 1 | FIFO bit | MISS/HIT |
| 0 | 00000 | Mem[0] | - | 0 | MISS |
| 512 | 00000 | Mem[0] | Mem[512] | 0 | MISS |
| 1024 | 00000 | Mem[1024] | Mem[512] | 1 | MISS |
| 1536 | 00000 | Mem[1024] | Mem[1536] | 0 | MISS |
| 2048 | 00000 | Mem[2048] | Mem[1536] | 1 | MISS |
| 1536 | 00000 | Mem[2048] | Mem[1536] | 1 | HIT |
| 1024 | 00000 | Mem[2048] | Mem[1024] | 0 | MISS |
| 512 | 00000 | Mem[512] | Mem[1024] | 1 | MISS |
| 0 | 00000 | Mem[512] | Mem[0] | 0 | MISS |
| 1024 | 00000 | Mem[1024] | Mem[0] | 1 | MISS |

Παρατηρούμε ότι έχουμε σωστό cache hit rate και με τις δυο πολιτικές αντικατάστασης!

3)**’Εστω ότι η cache είναι 4 way associative με μέγεθος 256 bytes και block size = 4 bytes και πολιτική αντικατάστασης LRU**

Στην προκειμένη περίπτωση τα sets είναι: 256/4(bytes per block)\*4(blocks per set) = **16 = 2^4**

Για να βρούμε σε ποιο set ανήκει η κάθε διεύθυνση κοιτάμε τα **4 bits** μετα από τα 2 LSB του block offset.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Διεύθυνση | Set | Set Block 00 | Set Block 01 | Set block  10 | Set block  11 | RLU bits | MISS/HIT |
| 0 | 0000 | Mem[0] | - | - | - | 00 | MISS |
| 512 | 0000 | Mem[0] | Mem[512] | - | - | 00 | MISS |
| 1024 | 0000 | Mem[0] | Mem[512] | Mem[1024] | - | 00 | MISS |
| 1536 | 0000 | Mem[0] | Mem[512] | Mem[1024] | Mem[1536] | 00 | MISS |
| 2048 | 0000 | Mem[2048] | Mem[512] | Mem[1024] | Mem[1536] | 01 | MISS |
| 1536 | 0000 | Mem[2048] | Mem[512] | Mem[1024] | Mem[1536] | 01 | HIT |
| 1024 | 0000 | Mem[2048] | Mem[512] | Mem[1024] | Mem[1536] | 01 | HIT |
| 512 | 0000 | Mem[2048] | Mem[512] | Mem[1024] | Mem[1536] | 00 | HIT |
| 0 | 0000 | Mem[0] | Mem[512] | Mem[1024] | Mem[1536] | 11 | MISS |
| 1024 | 0000 | Mem[0] | Mem[512] | Mem[1024] | Mem[1536] | 11 | HIT |

Έχουμε Cash hit ratio 4/10 = **40%** το οποίο είναι σωστό! Ας εξετάσουμε την ίδια περίπτωση απλώς μ ε πολιτική αντικατάστασης FIFO(First In First Out).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Διεύθυνση | Set | Set Block 00 | Set Block 01 | Set block  10 | Set block  11 | FIFO bits | MISS/HIT |
| 0 | 0000 | Mem[0] | - | - | - | 00 | MISS |
| 512 | 0000 | Mem[0] | Mem[512] | - | - | 00 | MISS |
| 1024 | 0000 | Mem[0] | Mem[512] | Mem[1024] | - | 00 | MISS |
| 1536 | 0000 | Mem[0] | Mem[512] | Mem[1024] | Mem[1536] | 00 | MISS |
| 2048 | 0000 | Mem[2048] | Mem[512] | Mem[1024] | Mem[1536] | 01 | MISS |
| 1536 | 0000 | Mem[2048] | Mem[512] | Mem[1024] | Mem[1536] | 01 | HIT |
| 1024 | 0000 | Mem[2048] | Mem[512] | Mem[1024] | Mem[1536] | 01 | HIT |
| 512 | 0000 | Mem[2048] | Mem[512] | Mem[1024] | Mem[1536] | 01 | HIT |
| 0 | 0000 | Mem[2048] | Mem[0] | Mem[1024] | Mem[1536] | 10 | MISS |
| 1024 | 0000 | Mem[0] | Mem[512] | Mem[1024] | Mem[1536] | 10 | HIT |

Παρατηρούμε ότι έχουμε λάθος cache hit rate και με τις δυο πολιτικές αντικατάστασης!

**ΣΗΜΕΙΩΣΗ:** παρόμοια αποτελέσματα βγάζουμε και για μέγεθος cache: 512 bytes απλώς στην περίπτωση 1 θα ελέγχαμε 6 bits σε κάθε διεύθυνση και στην 2η περίπτωση 5 bits αντίστοιχα:

(1) sets: 512/4(bytes per block)\*2(blocks per set) = **64 = 2^6**

(2) sets: 512/4(bytes per block)\*4(blocks per set) = **32** = **2^5**

Όμως τα bits αυτά δείχνουν πάλι ότι όλες οι διευθύνσεις αντιστοιχούν στο set index 0 επομένως θα παράγουμε το ακριβώς ιδιο πινακάκι συμπεριφοράς της cache.

Καταληκτικά από την ακολουθία #2 συμπεραίνουμε ότι **η cache είναι 2-way-associative** με **4 bytes/block** και μέχρι στιγμής με άγνωστο μέγεθος και πολιτική αντικατάστασης.

Η αναπαράσταση συμπεριφοράς της cache με την ακολουθία #3 θα γίνει με τον προηγούμενο τρόπο , από την στιγμή που έχουμε παλι διευθύνσεις που διαφέρουν κατά πολύ μεταξύ τους (θα αναφερόμαστε μόνο στα block του κάθε set).Με την συγκεκριμένη ακολουθία θα προσπαθήσουμε να βρούμε την πολιτική αντικατάστασης η το μέγεθος της cache η και τα δυο ανάλογα με τα αποτελέσματα που θα μας παρέχει η συγκεκριμένη ανάλυση.

1)**’Εστω ότι η cache είναι 2-way-associative με 4 bytes/block , μέγεθος 256 bytes και πολιτική αντικατάστασης LRU**

Στην προκειμένη περίπτωση τα sets είναι: 256/4(bytes per block)\*2(blocks per set) = **32 = 2^5**

Για να βρούμε σε ποιο set ανήκει η κάθε διεύθυνση κοιτάμε τα **5 bits** μετα από τα 2 LSB του block offset.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Διεύθυνση | Set Index | Set Block 0 | Set Block 1 | LRU Bit | MISS/HIT |
| 0 | 00000 | Mem[0] | - | 1 | MISS |
| 64 | 10000 | Mem[64] | - | 1 | MISS |
| 128 | 00000 | Mem[0] | Mem[128] | 0 | MISS |
| 256 | 00000 | Mem[256] | Mem[128] | 1 | MISS |
| 512 | 00000 | Mem[256] | Mem[512] | 0 | MISS |
| 256 | 00000 | Mem[256] | Mem[512] | 1 | HIT |
| 128 | 00000 | Mem[256] | Mem[128] | 0 | MISS |
| 64 | 10000 | Mem[64] | - | 1 | HIT |
| 0 | 00000 | Mem[0] | Mem[128] | 1 | MISS |
| 512 | 00000 | Mem[0] | Mem[512] | 0 | MISS |

Έχουμε Cash hit ratio 2/10 = **20%** το οποίο είναι λάθος. Ας εξετάσουμε την ίδια περίπτωση απλώς μ ε πολιτική αντικατάστασης FIFO(First In First Out).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Διεύθυνση | Set Index | Set Block 0 | Set Block 1 | FIFO bit | MISS/HIT |
| 0 | 00000 | Mem[0] | - | 1 | MISS |
| 64 | 10000 | Mem[64] | - | 1 | MISS |
| 128 | 00000 | Mem[0] | Mem[128] | 0 | MISS |
| 256 | 00000 | Mem[256] | Mem[128] | 1 | MISS |
| 512 | 00000 | Mem[256] | Mem[512] | 0 | MISS |
| 256 | 00000 | Mem[256] | Mem[512] | 0 | HIT |
| 128 | 00000 | Mem[128] | Mem[512] | 1 | MISS |
| 64 | 10000 | Mem[64] | - | 1 | HIT |
| 0 | 00000 | Mem[128] | Mem[0] | 0 | MISS |
| 512 | 00000 | Mem[512] | Mem[0] | 1 | MISS |

Παρατηρούμε ότι έχουμε λάθος cache hit rate (**20%**) και με τις δυο πολιτικές αντικατάστασης!

Σε αυτό το σημείο εύκολα συνάγουμε το συμπέρασμα ότι έχουμε πρόβλημα με το μέγεθος της μνήμης. Θα ξαναεξετάσουμε την παραπάνω περίπτωση αλλά αυτή την φορά με μέγεθος μνήμης **512 byte.**

2) **’Εστω ότι η cache είναι 2-way-associative με 4 bytes/block , μέγεθος 512 bytes και πολιτική αντικατάστασης LRU**

Στην προκειμένη περίπτωση τα sets είναι: 512/4(bytes per block)\*2(blocks per set) = **64 = 2^6**

Για να βρούμε σε ποιο set ανήκει η κάθε διεύθυνση κοιτάμε τα **6 bits** μετα από τα 2 LSB του block offset.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Διεύθυνση | Set Index | Set Block 0 | Set Block 1 | LRU Bit | MISS/HIT |
| 0 | 000000 | Mem[0] | - | 1 | MISS |
| 64 | 010000 | Mem[64] | - | 1 | MISS |
| 128 | 100000 | Mem[128] | - | 1 | MISS |
| 256 | 000000 | Mem[0] | Mem[256] | 0 | MISS |
| 512 | 000000 | Mem[512] | Mem[256] | 1 | MISS |
| 256 | 000000 | Mem[512] | Mem[256] | 0 | HIT |
| 128 | 100000 | Mem[128] | - | 1 | HIT |
| 64 | 010000 | Mem[64] | - | 1 | HIT |
| 0 | 000000 | Mem[0] | Mem[256] | 1 | MISS |
| 512 | 000000 | Mem[0] | Mem[512] | 0 | MISS |

Έχουμε Cash hit ratio 3/10 = **30%** το οποίο είναι σωστό! Ας εξετάσουμε την ίδια περίπτωση απλώς μ ε πολιτική αντικατάστασης FIFO(First In First Out).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Διεύθυνση | Set Index | Set Block 0 | Set Block 1 | FIFO Bit | MISS/HIT |
| 0 | 000000 | Mem[0] | - | 0 | MISS |
| 64 | 010000 | Mem[64] | - | 0 | MISS |
| 128 | 100000 | Mem[128] | - | 0 | MISS |
| 256 | 000000 | Mem[0] | Mem[256] | 0 | MISS |
| 512 | 000000 | Mem[512] | Mem[256] | 1 | MISS |
| 256 | 000000 | Mem[512] | Mem[256] | 1 | HIT |
| 128 | 100000 | Mem[128] | - | 0 | HIT |
| 64 | 010000 | Mem[64] | - | 0 | HIT |
| 0 | 000000 | Mem[512] | Mem[0] | 0 | MISS |
| 512 | 000000 | Mem[512] | Mem[0] | 0 | HIT |

Τελίκα με πολιτική αντικατάστασης FIFO βγάζουμε λάθος cache hit rate (40%), άρα επικρατεί η πολιτική LRU!

Με την τελευταία ανάλυση έχουμε σχηματίσει μια ολοκληρωμένη εικόνα για την μνημη cache, συγκεκριμένα:

1) Assiocivity: **2-way**

2) Μέγεθος του cache block: **4 bytes**

3) Μέγεθος της Cache: **512 bytes**

4) Πολιτική αντικατάστασης ενός cache block: **LRU**

**Για να επαληθεύσουμε αυτά τα αποτελέσματα θα προβούμε στην ανάλυση της ακολουθίας #4..**

Για την ακολουθία #4 με τα παραπάνω χαρακτηριστικά της μνήμης cache προκύπτει ότι έχει sets:

512/4(bytes per block)\*2(blocks per set) = 64 = **2^6**

Για να βρούμε σε ποιο set ανήκει η κάθε διεύθυνση κοιτάμε τα **6 bits** μετα από τα 2 LSB του block offset.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Διεύθυνση | Set Index | Set block: 0 | Set Block: 1 | LRU bit | MISS/HIT |
| 0 | 000000 | Mem[0] | **-** | 1 | MISS |
| 512 | 000000 | Mem[0] | Mem[512] | 0 | MISS |
| 1024 | 000000 | Mem[1024] | Mem[512] | 1 | MISS |
| 0 | 000000 | Mem[1024] | Mem[0] | 0 | MISS |
| 1536 | 000000 | Mem[1536] | Mem[0] | 1 | MISS |
| 0 | 000000 | Mem[1536] | Mem[0] | 0 | HIT |
| 2048 | 000000 | Mem[2048] | Mem[0] | 1 | MISS |
| 0 | 000000 | Mem[2048] | Mem[0] | 0 | HIT |

To cache hit rate για την ακολουθία #4 είναι : 2/8 = **25% το οποιό ειναι σωστό! ‘Αρα οι τελικοί παράγοντες που υπολογίσαμε είναι σωστοί!**

***Άσκηση 3)***

Έχουμε :

Κώδικας C:

1)double a[3][100], b[101][3];

2)for (i = 0; i < 3; i++)

3) for (j = 0; j < 100; j++)

4) a[i][j] = b[j][0] \* b[j+1][0];

Η σειρά 4 θα εκτελεστεί 300 φορές επίσης διακρίνουμε ότι **σε κάθε εκτέλεση της εντολής 4 πραγματοποιούνται 2 loads(b[j][0], b[j+1][0]) και 1 store(a[i][j]).** Επειδή η C εκτελεί εντολές από δεξιά στα αριστερά θα εκτελείτε πρώτα η load(b[j+1][0]) και έπειτα η load(b[j][0]) ακολουθούμενη από την store(a[i][j] = b[j][0] \* b[j+1][0]).

H μνήμη cache είναι direct-mapped με 32-byte-blocks(4 doubles(8 bytes) ανα block) συνολικής μνήμης 16ΚΒ > 8(double bytes) \* (3\*100 (a)+ 3\*101(b)) οπότε οι πίνακες χωράνε ολόκληροι στην L1 cache. :

Αφού είναι 32 bytes / block αυτό σημαίνει ότι το **Block offset** = log2(32) = **5 bits**

Και η συνολική μνήμη 16ΚΒ = 16 \* 1024 = 16364 / 32 = 512 συνολικά blocks άρα το **index** = log2(512) = **9 bits**

Γενικά γνωρίζουμε ότι οι πίνακες στην C αποθηκεύονται στην μνήμη σε συνεχόμενα κελiα μνήμης και ότι η διεύθυνση του κάθε double ξεκινάει από πολ/σιο του 8.

**Υποθέτουμε επίσης ότι ο (α) πίνακας αποθηκεύεται κοντά στην μνήμη με τον (β) πίνακα ώστε να μην προκύπτουν μεταξύ τους διευθύνσεις με το ίδιο index.**

Οπότε τρέχοντας τον αλγόριθμο(δίνοντας τιμές στο i και j ) και τοποθετώντας τους doubles προκύπτει το ακόλουθο pattern (miss / hit) για τα loads:

**Θεωρούμε ότι το κάθε κελί είναι 8 bytes( 1 double) και όπου έχουμε [j, k] εννοούμε τον double b[j][k]**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Index** | **0-7 Bytes** | **8–15Bytes** | **16-23Bytes** | **24-32Bytes** | **Load([j,0])** | **Miss/Hit** | **Loop** |
| index | [0,0] | [0,1] | [0,2] | [1,0] | Load[1,0] | miss | 1 |
|  |  |  |  |  | Load[0,0] | hit |  |
| Index + 1 | [1,1] | [1,2] | [2,0] | [2,1] | Load[2,0] | miss | 2 |
|  |  |  |  |  | Load[1,0] | hit |  |
| Index + 2 | [2,2] | [3,0] | [3,1] | [3,2] | Load[3,0] | miss | 3 |
|  |  |  |  |  | Load[2,0] | hit |  |
| Index + 3 | [4,0] | [4,1] | [4,2] | [5,0] | Load[4,0] | miss | 4 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Load([3,0]) | hit |  |
|  |  |  |  |  | Load([5,0]) | hit | 5 |
|  |  |  |  |  | Load([4,0]) | hit |  |
| Index + 4 | [5,1] | [5,2] | [6,0] | [6,1] | Load([6,0]) | miss | 6 |
|  |  |  |  |  | Load([5,0]) | hit |  |
| Index + 5 | [6,2] | [7,0] | [7,1] | [7,2] | Load([7,0]) | miss | 7 |
|  |  |  |  |  | Load([6,0]) | hit |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Index + 6 | [8,0] | [8,1] | [8,2] | [9,0] | Load([8,0]) | miss | 8 |
|  |  |  |  |  | Load([7,0]) | hit |  |
|  |  |  |  |  | Load([9,0]) | hit | 9 |
|  |  |  |  |  | Load([8,0]) | hit |  |
| .  .  . | .  .  . | .  .  . | .  .  . | .  .  . | .  .  . | .  .  . | .  .  . |
|  |  |  |  |  |  |  |  |

Παρατηρούμε ότι εκτός από την 1η επανάληψη υπάρχει ανά 4εις επαναλήψεις το μοτίβο :

**Miss hit miss hit miss hit hit hit**

Για i = 1 γίνονται 100 επαναλήψεις για το j οπότε εκτός από την 1η έχουμε 99 επαναλήψεις που ακολουθούν το μοτίβο άρα : 99 = 24 \* 4 + 3 οπότε το μοτίβο εκτελείτai 24εις φορες και τις 3εις πρωτες επαναλήψεις το μοτίβου οπότε γενικά εχουμε :

**Miss hit |24 \*(miss hit miss hit miss hit hit hit) | miss hit miss hit miss hit**

Οπότε συνολικά **76 misses** και **124 hits**

Στις επόμενες επαναλήψεις οπίνακας b είναι φορτωμένος στην cache οπότε έχουμε 100% hit ratio άρα 0 misses οπότε από τις 600 συνολικά loads που εκτελούνται οι **524 είναι hit** και οι **76 είναι misses**(\* 8 bytes)

Παρόμοια βλέπουμε για τις stores ότι ακολουθούν το μοτίβο :

**Miss hit hit hit**

100(Επαναλήψεις 1 store ανά επανάληψη ) = 4 \* 25 + 0 οπότε το μοτίβο εκτελείτε 25 φορές αρα

25 misses στις 100 επαναλήψεις -> **75 misses στα 300 stores**

**Συνολικές(loads + stores)** = 600 + 300 = **900**

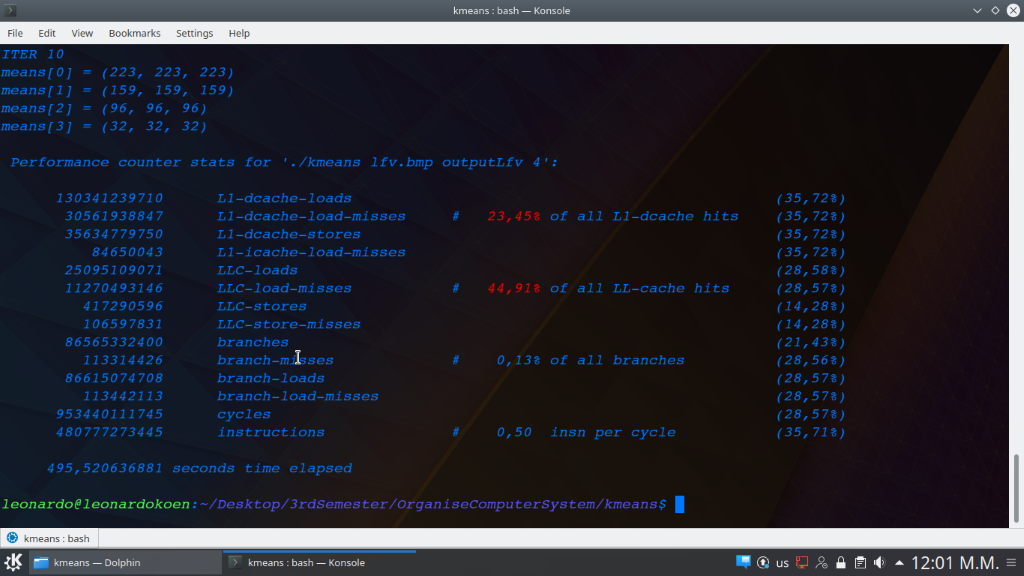
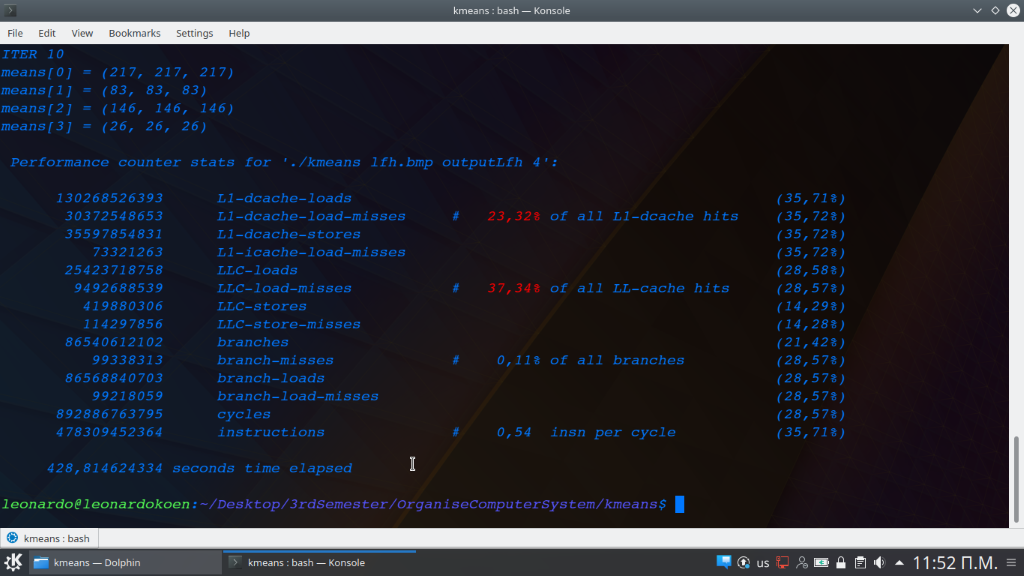
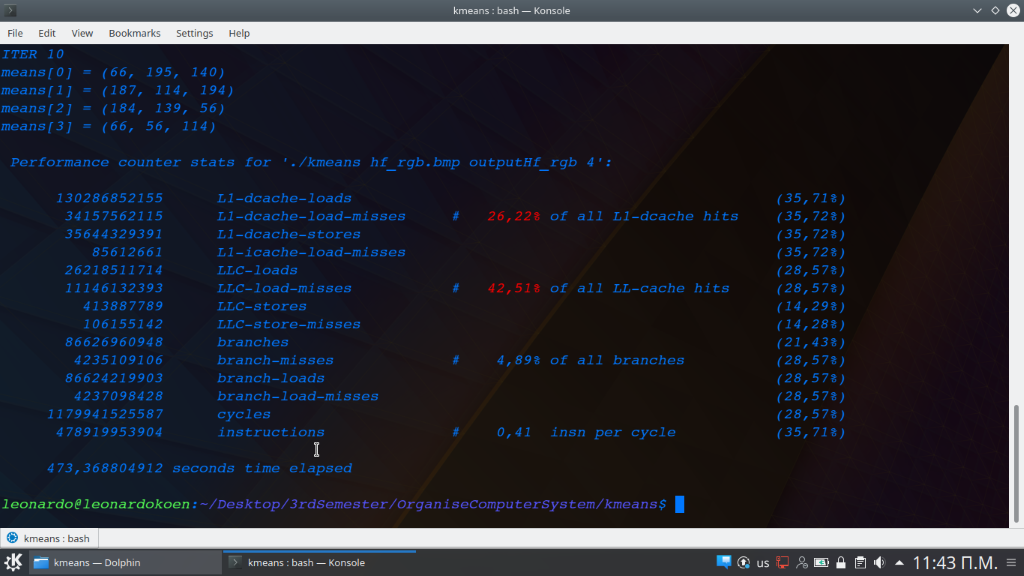
**Συνολικά misses** = 76 + 75 = **151**

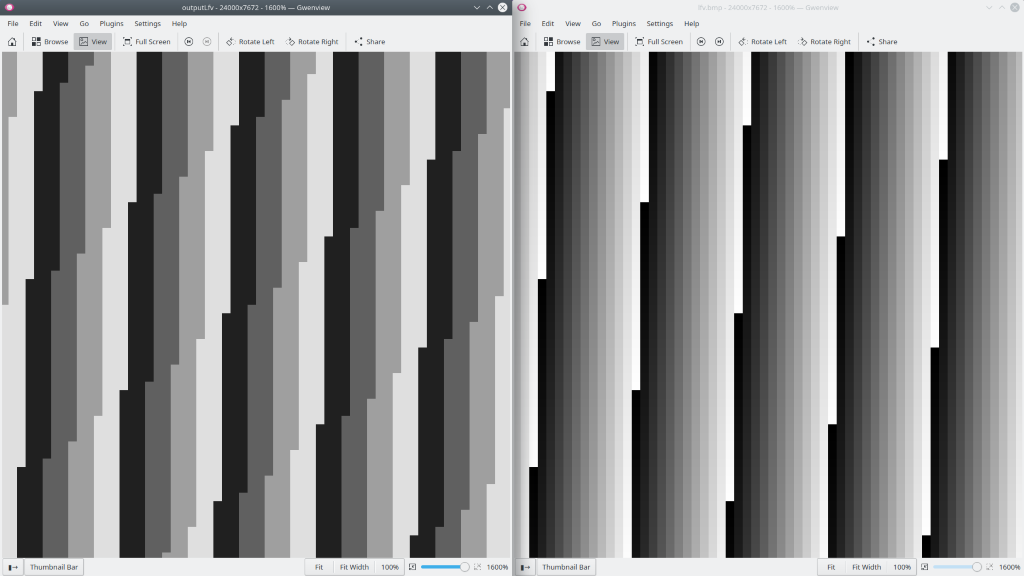
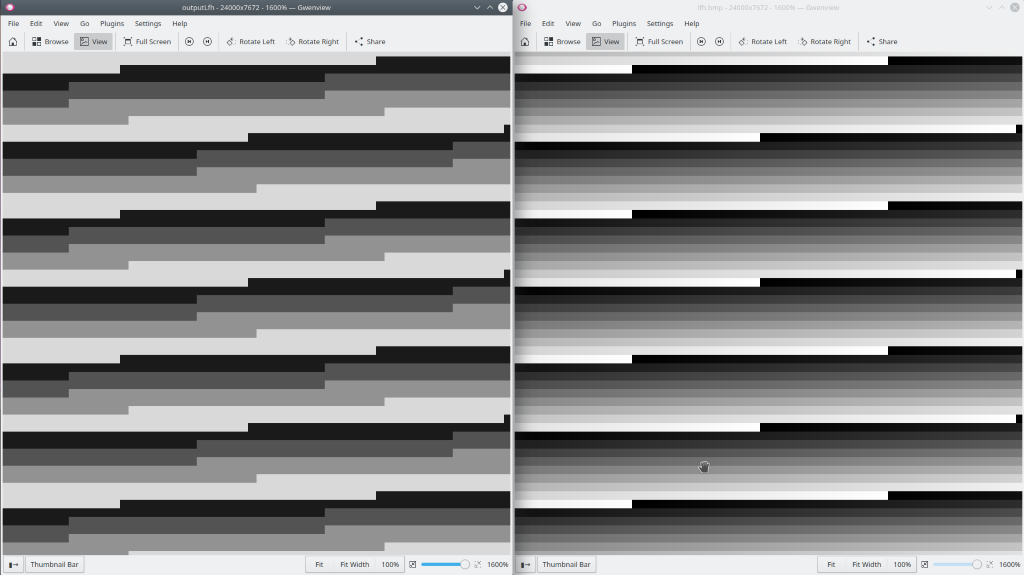
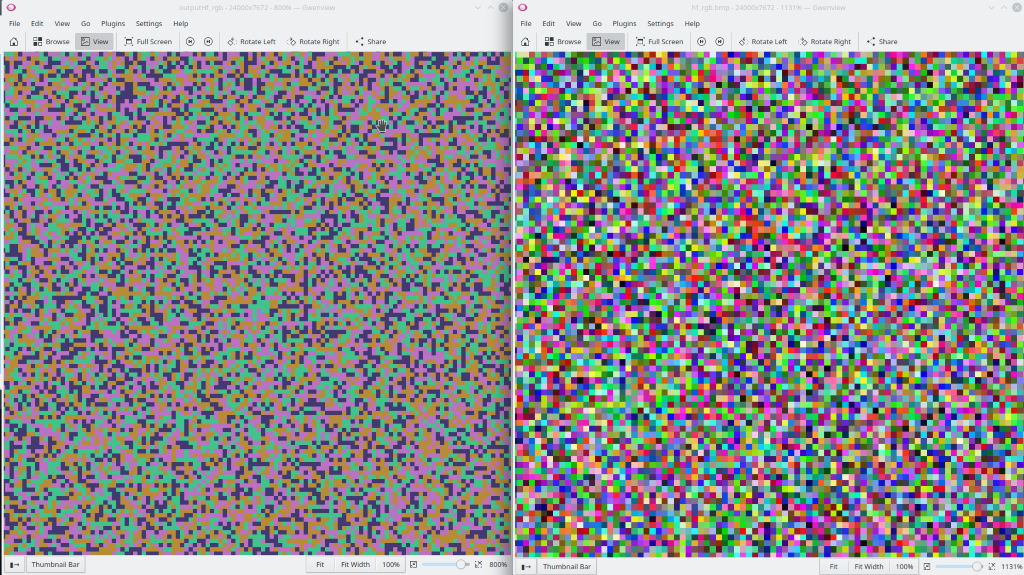
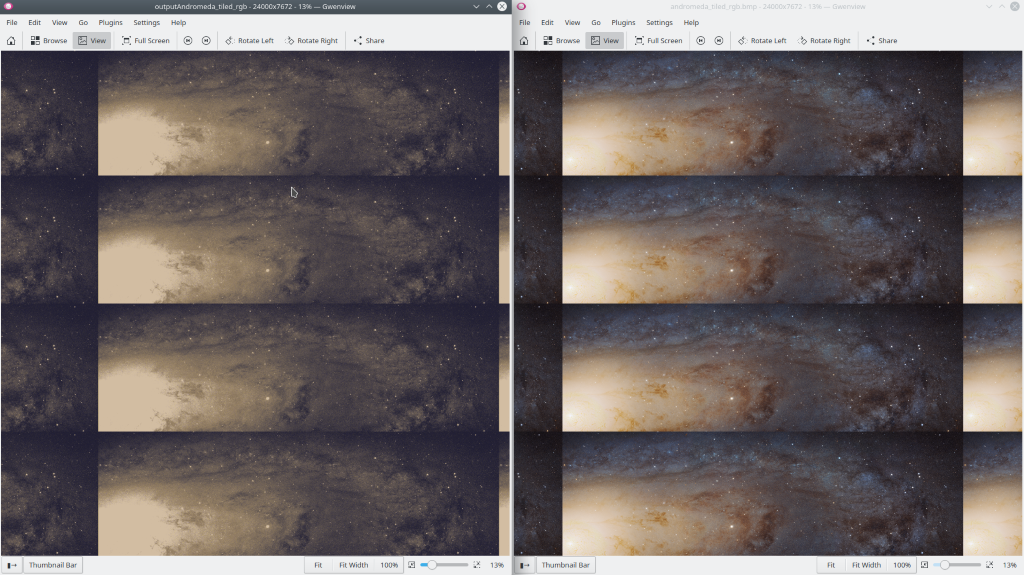
Οπότε **miss ratio** = 151 / 900 = **0.16777778** περιπου **16.8%**

***ΑΣΚΗΣΗ 4)***

**Α)**

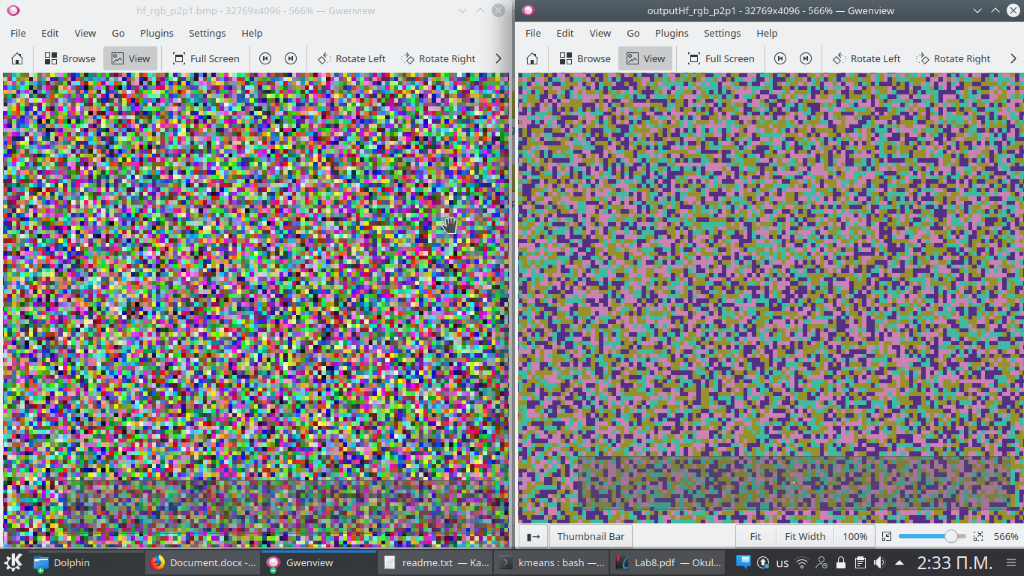
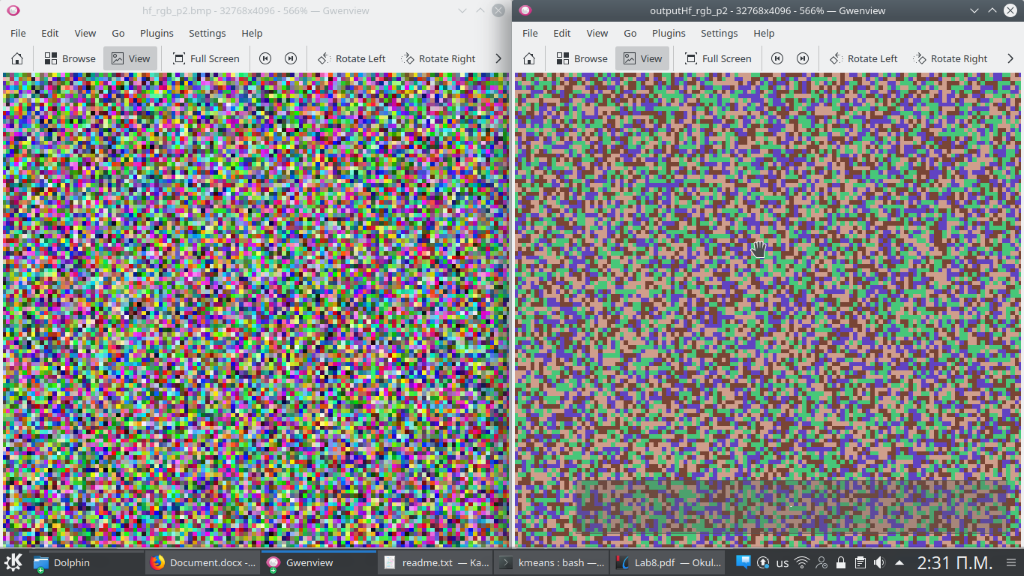
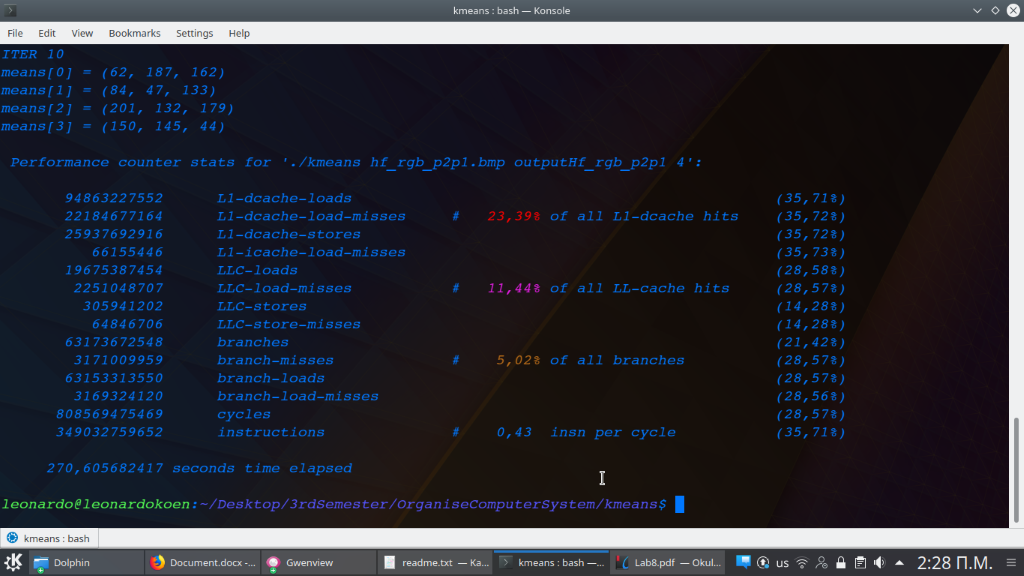
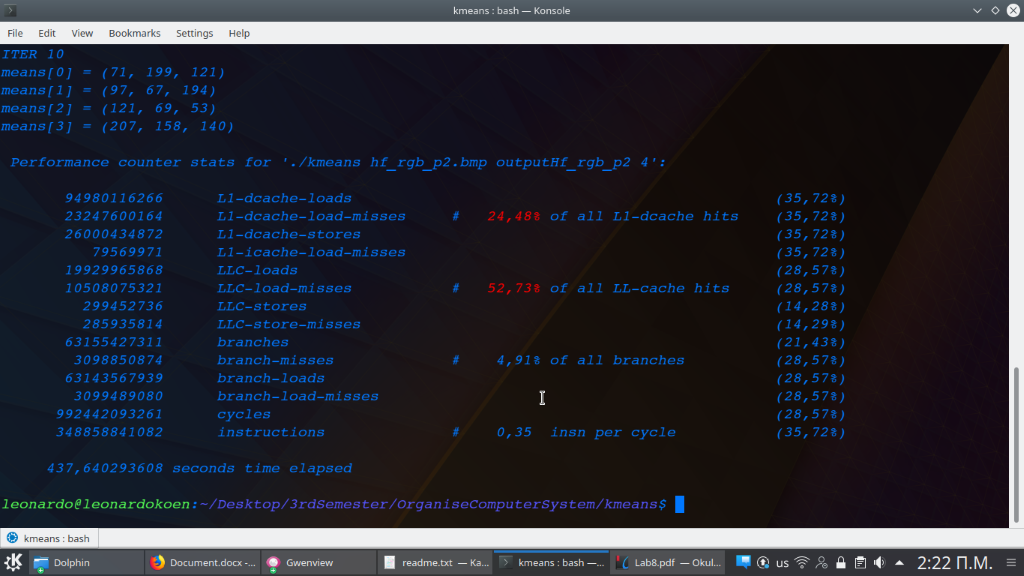






Όπως διακρίνουμε στα στατιστικά τα miss ratio είναι πάνω κάτω τα ίδια και στις 4 εκτελέσεις του κώδικα. Tα branch misprediction ratio διαφέρουν στις 2 πρώτες εκτελέσεις (andromeda\_rgb/hf\_rgb)σε σχέση με τις 2 τελευταίες(lfh/lfv) , διότι στις 2 τελευταίες διακρίνεται ένα συγκεκριμένο μοτίβο το οποίο είναι φιλικό προς τον branch predictor ενώ η τυχαιότητα των χρωμάτων των πρώτων 2 εικόνων τον δυσκολεύουν με αποτέλεσμα περισσότερα wrong predictions.

**B)**



**c)**

