CISC 203 Problem Set 4

Amy Brons

April 5, 2022

1. To start this problem, first we need to process and list the necessary given information.

   Given the listed information that the file is 2MB with 512-byte logical blocks, we can calculate the sectors:

   $T_{sectors} = \frac{(2(1024 \times 1024))}{512}$

   $= 4096$

   Now lets find the total read time as $T_{avgseek} + T_{rotationaldelay} + T_{transfer}$
   The rotational delay can be found by dividing the rotational rate over seconds, and then halving to find the average:

   $T_{rotationaldelay} = \frac{18000RPM}{60sec} = \frac{\frac{1}{300sec}}{2} = \frac{1}{600s}$

   $= 0.00166667s$

   $= 1.67ms$

   Transfer time is 1 rotation in 2000 sectors, for each of the 512 bytes, so therefore:

   $\frac{1}{300s} = 2000 \times 512$

   $1s = 300(2000 \times 512)$

   So therefore to transfer for each of the 512 bytes, this would equal:

   $\frac{512}{300 \times 2000 \times 512} = \frac{1}{300 \times 2000}s$

   A. To find the best case for this file to be read, we find the total time to do a single seek.

   This is found through the equation:

   $T_{total} = T_{avgseek} + T_{rotational} + T_{transfer}$

   $= 8ms + 1.67ms + (4096 \times \frac{1}{2000 \times 300}s)$

   $= 9.67ms + 0.00682666s$

   $= 9.67ms + 6.8267ms$

   $= 16.4967ms$

Therefore the best case for this file to be read given these specifications is 16.5 ms.

B. For a random case, we need to multiply the values of the average seek, transfer sector and the rotational with the total number of sectors.:
$T_{total} = 4096 \times (T_{avgseek} + T_{transfer} + T_{rotational}$

$= 4096 \times (8ms + 1.67ms + (\frac{1}{2000 \times 300}s)$

$= 4096 \times (9.67ms + 0.00000166667s)$

$= 4096 \times (9.67ms + 0.0016667ms)$

$= 4096 \times 9.671667ms$

$= 39615.146803ms$

Therefore a total time for processing on a random case, where blocks are mapped in a random order is 39615.15 ms.

2. Here are the processes to find the each values of the table:

To find the value of S, we need to first find the Block size x associativity, and then divide by the cache size. Basically the formula will be:

$S = \frac{C}{B \times E}$

These are the calculations:

$1.S = \frac{1024}{4 \times 4} = \frac{1024}{16} = 64$

$2.S = \frac{1024}{4 \times 256} = \frac{1024}{1024} = 1$

$3.S = \frac{1024}{8 \times 1} = \frac{1024}{8} = 128$

$4.S = \frac{1024}{8 \times 128} = \frac{1024}{1024} = 1$

$5.S = \frac{1024}{32 \times 1} = \frac{1024}{32} = 32$

$6.S = \frac{1024}{32 \times 4} = \frac{1024}{128} = 8$

To calculate the value of s, we take the log_2 of our number of cache sets. Here are the calculations:

$1.log_2 64 = 6$
$2.log_2 1 = 0$
$3.log_2 128 = 7$
$4.log_2 1 = 0$
$5.log_2 32 = 5$
$6.log_2 8 = 3$

To find the value of b, we need to base this on the number of bytes in a block, and applying this to a log. For example for 1.:

$log_2 4 = 2$

Therefore the number of block offset bits is 2.
Here are the full calculations:

1.$log_2 4 = 2$
2.$log_2 4 = 2$
3.$log_2 8 = 3$
4.$log_2 8 = 3$
5.$log_2 32 = 5$
6.$log_2 32 = 5$

Finally, to calculate the tag bits, we need to subtract from our full physical address bits. We will take out the offset bits and set bits. This is summarized with the formula:

$t = m - (s + b)$

Here are the calculations:

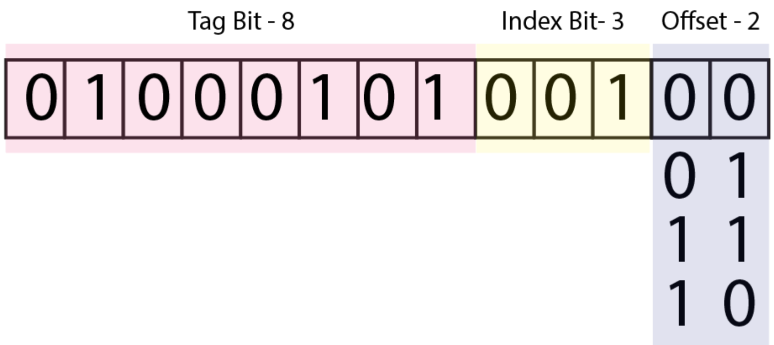
1.$t = 32 - (6 + 2) = 24$
2.$t = 32 - (0 + 2) = 30$
3.$t = 32 - (7 + 3) = 22$
4.$t = 32 - (0 + 3) = 29$
5.$t = 32 - (5 + 5) = 22$
6.$t = 32 - (3 + 5) = 24$

This is the completed table:

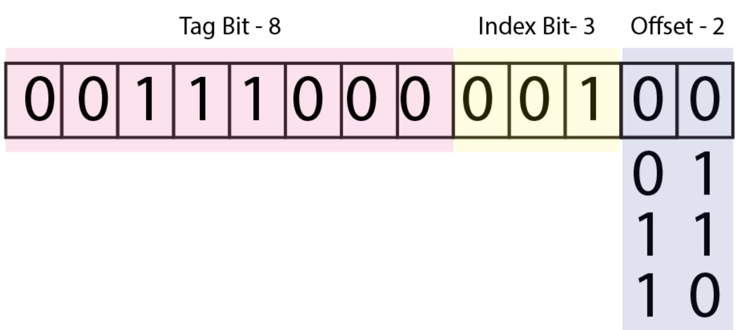
Cache	*m*	*C*	*B*	*E*	*S*	*t*	*s*	*b*
1.	32	1,024	4	4	64	24	6	2
2.	32	1,024	4	256	1	30	0	2
3.	32	1,024	8	1	128	22	7	3
4.	32	1,024	8	128	1	29	0	3
5.	32	1,024	32	1	32	22	5	5
6.	32	1,024	32	4	8	24	3	5

3. A. To list all of the hex memory addresses that the cache will hit in the chart found in 6.12, I created this chart to break down the different values:

   For set 1, tag 45:



For set 1, tag 38:



These are the values that will be hit in set one:

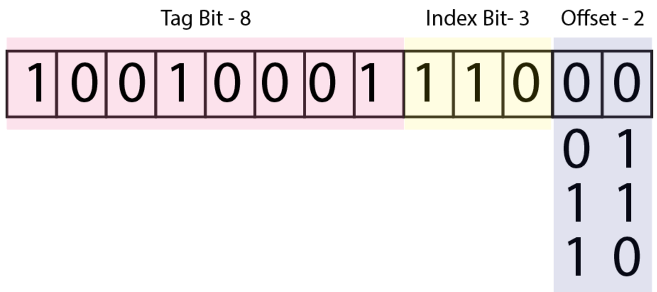
For tag 45:
0100010100100 = 0x8A4
0100010100101 = 0x8A5
0100010100110 = 0x8A6
0100010100111 = 0x8A7

For tag 38:
0011100000100 = 0x704
0011100000101 = 0x705
0011100000110 = 0x706
0011100000111 = 0x707

Therefore in set 1, these will be the hex addresses hit:
0x8A4, 0x8A5, 0x8A6, 0x8A7, 0x704, 0x705, 0x706, 0x707

B.To list all of the hex memory addresses that the cache will hit in the chart found in 6.12, I created this chart to break down the different values:

For set 6, tag 91:



These will be the values hit in set six:
1001000111000 = 0x1238
1001000111001 = 0x1239
1001000111010 = 0x123A
1001000111011 = 0x123B

Therefore in set 6, these will be the hex addresses hit:
0x1238, 0x1239, 0x123A, 0x123B