ISTANBUL TECHNICAL UNIVERSITY

Faculty of Computer and Informatics Engineering

Department of Computer Engineering

BLG 322E

Computer Architecture

Homework #4

İBRAHİM KARATEKE

150210705

Question 1:

a)

256 KB = 2^{18}

1 KB = 2^{10}

16 Bytes = 2^{4}

$2^{10} / 2^{4} = 2^{6}$ frames

$2^{6} / 2 = 2^{5}$ sets

18 – (4+5) = 9

Hence,

- Physical Address : 18-bit
- Tag : 9-bit
- Set Num: : 5-bit
- Word Num : 4-bit

b)

The tag memory consists of 64 rows, each dedicated to a specific frame and adorned with a 9-bit tag. In addition to these tags, each row contains extra details: a validity indicator, a dirty flag, and a flag for FIFO, collectively occupying a space of 12 bits per row. It's worth noting that there's no need for an aging counter, because the system doesn't employ the LRU (Least Recently Used) replacement method.

Size of the tag memory: **64x12** bits.

c)

- A[0][0] – A[0][9]: $00210 - $00219

  00 0000 0010 0001 xxxx

  A block including 10 bytes of array A[0] is placed into 1st frame of the set 0 0001 of the cache memory. Tag value is 00 0000 001.

- A[1][0] – A[1][9]: $00410 - $00419

  00 0000 0100 0001 xxxx

  A block including 10 bytes of array A[1] is placed into 2nd frame of the set 0 0001 of the cache memory. Tag value is 00 0000 010.

- A[2][0] – A[2][9]: $00820 - $00829

  00 0000 1000 0010 xxxx

  A block including 10 bytes of array A[2] is placed into 1st frame of the set 0 0010 of the cache memory. Tag value is 00 0000 100.

- A[3][0] – A[3][9]: $01020 - $01029

  00 0001 0000 0010 xxxx

  A block including 10 bytes of array A[3] is placed into 2nd frame of the set 0 0010 of the cache memory. Tag value is 00 0001 000.

- A[4][0] – A[4][9]: $02010 - $02019

  00 0010 0000 0001 xxxx

  Set 1: 0 0001 is full. According to FIFO method, 1st frame in set 1, including the elements A[0][0]-A[0][9] is replaced with A[4]. A block including 10 bytes of array A[4] is replaced into the 1st frame of set 0 0001 of the cache memory. Tag value is 00 0010 000.

- A[5][0] – A[5][9]: $04010 - $04019

  00 0100 0000 0001 xxxx

  Set 1: 0 0001 is full. According to FIFO method, 2nd frame in set 1, including the elements A[1][0]-A[1][9] is replaced with A[5]. A block including 10 bytes of array A[5] is replaced into the 2nd frame of set 0 0001 of the cache memory. Tag value is 00 0100 000.

- A[6][0] – A[6][9]: $08020 - $08029

  00 1000 0000 0010 xxxx

  Set 2: 0 0010 is full. According to FIFO method, 1st frame in set 2, including the elements A[2][0]-A[2][9] is replaced with A[6]. A block including 10 bytes of array A[6] is replaced into the 1st frame of set 0 0010 of the cache memory. Tag value is 00 1000 000.

- A[7][0] – A[7][9]: $10020 - $10029

  01 0000 0000 0010 xxxx

  Set 2: 0 0010 is full. According to FIFO method, 2nd frame in set 2, including the elements A[3][0]-A[3][9] is replaced with A[7]. A block including 10 bytes of array A[7] is replaced into the 2nd frame of set 0 0010 of the cache memory. Tag value is 01 0000 000.

In total, **4 replacements occur.**

d) In order to understand the iteration lets demonstrate reading the first column of the array.

- A[0][0] $00210
  00 0000 0010 0001 xxxx

  A block including 1 bytes of array A[0][0] is placed into 1st frame of the set 0 0001 of the cache memory. Tag value is 00 0000 001.

- A[1][0] $00410
  00 0000 0100 0001 xxxx
  A block including 1 bytes of array A[1][0] is placed into 2nd frame of the set 0 0001 of the cache memory. Tag value is 00 0000 010.

- A[2][0] $00820
  00 0000 1000 0010 xxxx
  A block including 1 bytes of array A[2][0] is placed into 1st frame of the set 0 0010 of the cache memory. Tag value is 00 0000 100

- A[3][0] $01020
  00 0001 0000 0010 xxxx
  A block including 1 bytes of array A[3][0] is placed into 2nd frame of the set 0 0010 of the cache memory. Tag value is 00 0001 000.

- A[4][0] $02010
  00 0010 0000 0001 xxxx
  Set 1: 0 0001 is full. According to FIFO method, 1st frame in set 1, including the element A[0][0] is replaced with A[4][0]. A block including 1 bytes of array A[4][0] is replaced into the 1st frame of set 0 0001 of the cache memory. Tag value is 00 0010 000

- A[5][0] $04010
  00 0100 0000 0001 xxxx
  Set 1: 0 0001 is full. According to FIFO method, 2nd frame in set 1, including the element A[1][0] is replaced with A[5][0]. A block including 1 bytes of array A[5][0] is replaced into the 2nd frame of set 0 0001 of the cache memory. Tag value is 00 0100 000

- A[6][0] $08020
  00 1000 0000 0010 xxxx
  Set 2: 0 0010 is full. According to FIFO method, 1st frame in set 2, including the element A[2][0] is replaced with A[6][0]. A block including 1 bytes of array A[6][0] is replaced into the 1st frame of set 0 0010 of the cache memory. Tag value is 00 1000 000.

- A[7][0] $10020
  01 0000 0000 0010 xxxx
  Set 2: 0 0010 is full. According to FIFO method, 2nd frame in set 2, including the elements A[3][0] is replaced with A[7][0]. A block including 1 bytes of array A[7][0] is replaced into the 2nd frame of set 0 0010 of the cache memory. Tag value is 01 0000 000.

Notice that in this loop, we are iterating on array column-wise. Therefore, in each iteration we need to access all the given starting addresses.

j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8	j=9
A[0][0]	A[0][1]	A[0][2]	A[0][3]	A[0][4]	A[0][5]	A[0][6]	A[0][7]	A[0][8]	A[0][9]
A[1][0]	A[1][1]	A[1][2]	A[1][3]	A[1][4]	A[1][5]	A[1][6]	A[1][7]	A[1][8]	A[1][9]
A[2][0]	A[2][1]	A[2][2]	A[2][3]	A[2][4]	A[2][5]	A[2][6]	A[2][7]	A[2][8]	A[2][9]
A[3][0]	A[3][1]	A[3][2]	A[3][3]	A[3][4]	A[3][5]	A[3][6]	A[3][7]	A[3][8]	A[3][9]
A[4][0]	A[4][1]	A[4][2]	A[4][3]	A[4][4]	A[4][5]	A[4][6]	A[4][7]	A[4][8]	A[4][9]
A[5][0]	A[5][1]	A[5][2]	A[5][3]	A[5][4]	A[5][5]	A[5][6]	A[5][7]	A[5][8]	A[5][9]
A[6][0]	A[6][1]	A[6][2]	A[6][3]	A[6][4]	A[6][5]	A[6][6]	A[6][7]	A[6][8]	A[6][9]
A[7][0]	A[7][1]	A[7][2]	A[7][3]	A[7][4]	A[7][5]	A[7][6]	A[7][7]	A[7][8]	A[7][9]

We have already shown the "j=0" case. Note that after this column, the only change occurs word. Hence, the set and the tag are the same for all 10 A[0][x] elements. Furthermore, after j = 0, the set 1 "0 0001" and set 2 "0 0010" are full. Consider the j=1 case:

A[0][1] Set: 0 0001 set 1 Tag: 00 0000 001
A[1][1] Set: 0 0001 set 1 Tag: 00 0000 010
A[2][1] Set: 0 0010 set 2 Tag: 00 0000 100
A[3][1] Set: 0 0010 set 2 Tag: 00 0001 000
A[4][1] Set: 0 0001 set 1 Tag: 00 0010 000
A[5][1] Set: 0 0001 set 1 Tag: 00 0100 000
A[6][1] Set: 0 0010 set 2 Tag: 00 1000 000
A[7][1] Set: 0 0010 set 2 Tag: 01 0000 000

Since we only work with set 1 and set 2 for all the elements, and both sets are already full due to the previous read operation in j=0, whenever we move another column for all elements a replacement occurs in that column.

- In j=0 **4** replacements occur
- For the rest of the columns 8 × (10 - 1) = **72** replacements occur.

In total, **4 + 72 = <u>76 replacements occur.</u>**

e)

To increase the hit ratio, we can change the starting addresses of the arrays as follows.

ID	Starting Address	Binary Representation	Set
A[0] (A[0][0])	$ 00210	00 0000 0010 0001 xxxx	0 0001 – Set 1
A[1] (A[1][0])	$ 00420	00 0000 0100 0010 xxxx	0 0010 – Set 2
A[2] (A[2][0])	$ 00830	00 0000 1000 0011 xxxx	0 0011 – Set 3
A[3] (A[3][0])	$ 01040	00 0001 0000 0100 xxxx	0 0100 – Set 4
A[4] (A[4][0])	$ 02050	00 0010 0000 0101 xxxx	0 0101 – Set 5
A[5] (A[5][0])	$ 04060	00 0100 0000 0110 xxxx	0 0110 – Set 6
A[6] (A[6][0])	$ 08070	00 1000 0000 0111 xxxx	0 0111 – Set 7
A[7] (A[7][0])	$ 10080	01 0000 0000 1000 xxxx	0 1000 – Set 8

By changing starting adresses, we have obtained 8 distinct sets. For row-wise reading (like we did in part-c) no replacement occurs. Because all blocks will moved to the separete sets. For column-wise reading there will not occur any replacement in first two columns for j=0 and j=1. However, after reading those columns, we will have moved two frames into those sets and they will be full. For the rest of all the elements there will occur replacements. Notice that since we have obtained distinct sets, the hit ratio is maximized.

Question 2:

64 KB = 2^{16}

1 KB = 2^{10}

8 Bytes = 2^{3}

$2^{10} / 2^{3} = 2^{7}$ frames

$2^{7} / 2 = 2^{6}$ sets

16 – (3+6) = 7

Hence,

- Physical Address : 16-bit
- Tag : 7-bit
- Set Num: : 6-bit
- Word Num : 3-bit

a) In the fastest case all variables are in the same block of the main memory and in the same cache frame. Same set number and same tag value. For example;

A: 0000 0000 0000 0000 = $0000

B: 0000 0000 0000 0001 = $0001

C: 0000 0000 0000 0010 = $0002

Generate address of A, miss, transfer block from main to cache memory at the same time get A. Read B, hit. Write C, hit, write to cache and main memory (because of WT).,

$\mathbf{t_a = t_B + t_c + t_m}$

b) In the slowest case all variables try to share the same set but they are in different blocks of the main memory. Set numbers are same but their tags are different. For example:

A: 0000 0000 0000 0000 = \$0000

B: 0000 0010 0000 0000 = \$0200

C: 0000 0100 0000 0000 = \$0400

Generate address of A, miss, transfer block from main to cache memory, at the same time get A (set:0, frame=0). Generate address of B, miss, transfer block from main to cache memory, at the same time get B (set:0, frame=1). Generate address of C, miss, transfer block from main to cache memory (because of WA) (set:0, frame=0), write to cache and main memory (because of WT).

$\mathbf{t_a = t_B + t_B + t_B + t_m}$