HOMEWORK-1

1.

$m = 16$

a.

$C_1 (m, C, B, E, S, t, s, b) = (16, 64, 4, 16, 1, 10, 4, 2)$

$C_2 (m, C, B, E, S, t, s, b) = (16, 64, 16, 1, 4, 10, 2, 4)$

b.

Cache1

			s	b	
BA00 - 1011	1010		0000	0000	→ Miss
BA04 - 1011	1010		0000	0100	→ Miss
AA08 - 1010	1010		0000	1000	→ Miss
BA05 - 1011	1010		0000	0101	→ Hit
AA14 - 1010	1010		0001	0100	→ Miss
AA11 - 1010	1010		0001	0001	→ Miss
AA13 - 1010	1010		0001	0011	→ Hit
AA38 - 1010	1010		0011	1000	→ Miss
AA09 - 1010	1010		0000	1001	→ Hit
AA0B - 1010	1010		0000	1011	→ Hit
BA04 - 1011	1010		0000	0100	→ Hit
AA2B - 1010	1010		0010	1011	→ Miss
BA05 - 1011	1010		0000	0101	→ Hit
BA06 - 1011	1010		0000	0110	→ Hit
AA09 - 1010	1010		0000	1001	→ Hit
AA11 - 1010	1010		0001	0001	→ Hit

Hits = 9
Miss = 7

S_0 – BA00 – BA03

S_1 – BA04 – BA07

S_2 – AA08 – AA0B

S_4 – AA10 – AA13

S_5 – AA14 – AA17

S_{10} – AA28 – AA2B

S_{14} – AA38 – AA3B

Sets available will be S_0 to S_{15}

Rest of the sets will be empty

Cache 2

		a	b	
BA00 – 1011	1010	0000	0000	→ Miss
BA04 – 1011	1010	0000	0100	→ Hit
AA08 – 1010	1010	0000	1000	→ Miss
BA05 – 1011	1010	0000	0101	→ Miss
AA14 – 1010	1010	0001	0100	→ Miss
AA11 – 1010	1010	0001	0001	→ Hit
AA13 – 1010	1010	0001	0011	→ Hit
AA38 – 1010	1010	0011	1000	→ Miss
AA09 – 1010	1010	0000	1001	→ Miss
AA0B – 1010	1010	0000	1011	→ Hit
BA04 – 1011	1010	0000	0100	→ Miss
AA2B – 1010	1010	0010	1011	→ Miss
BA05 – 1011	1010	0000	0101	→ Hit

BA06 - 1011 1010 0000 0110 → Hit

AA09 - 1010 1010 0000 1001 → Miss

AA11 - 1010 1010 0001 0001 → Hit

Miss - 9
Hit - 7

S_0 - AA00 - AA0F

S_1 - AA10 - AA1F

S_2 - AA20 - AA2F

S_3 - AA30 - AA3F

C. The above given address seq has more hits in C. Thus, no change required

BA00, BA04, AA08, BA05, AA14, AA11, AA13, AA38, AA09, AA0B, BA04, AA2B, BA05, BA06, AA09, AA11

2.

$t = 22$

$s = 8$

$b = 4$

$\Rightarrow S = 2^s$

$m = t + s + b$

$= 22 + 8 + 4$

$= 34.$

a. $m = 34.$

Address space = 31 bytes

= 2 GBytes $\equiv 2^{34}$ bits

b. $C = S \times E \times B$

$= 2^8 \times 1 \times 2^4$

$= 2^{12} = 4096$ bytes

c. # tag bits = 22

block bits = 4

set bits = 8

we have a valid bit too $\Rightarrow 22 + 4 + 8 + 1$

$= 35$ bits

d. $t = 22$

$\Rightarrow 2^{22}$ Ans

$\Rightarrow 2^{22}$ ~~caches~~ ~~blocks~~ Memory blocks can share one cache line

3.

Size (int) = 4

# of ints = 1,000,000

m = 64.

Cache hit rate desired = 87.5%

$= 7/8 \times 100$

⇒ if we read 8 ints in one go, we will have 7 hits for each Miss

for a direct mapped cache, E = 1

Lets say, we want to keep the cost of cache minimum

⇒ $C = E \times S \times B$ has to be min

1 (for direct mapped)

each int = 4 bytes

⇒ 8 x ints = 32 bytes

⇒ b = 5

⇒ B = 32

So, $C = S \times 1 \times 32$

if we have S = 1 ⇒ 1 set bit

We can read int, find Min & evict the line after that

cache parameters (m, C, B, E, S, t, s, b)

= (64, 64, 32, 1, 2, 58, 1, 5)

Let Min be a register variable

```
R1 = a[0]
for (i=1; i< 1000000; i++)
{
    if ( R1 > a[i])
        { R1 = a[i]; }
}
return R1;
```

4. C code

Row major by default

for a good locality of reference (I)

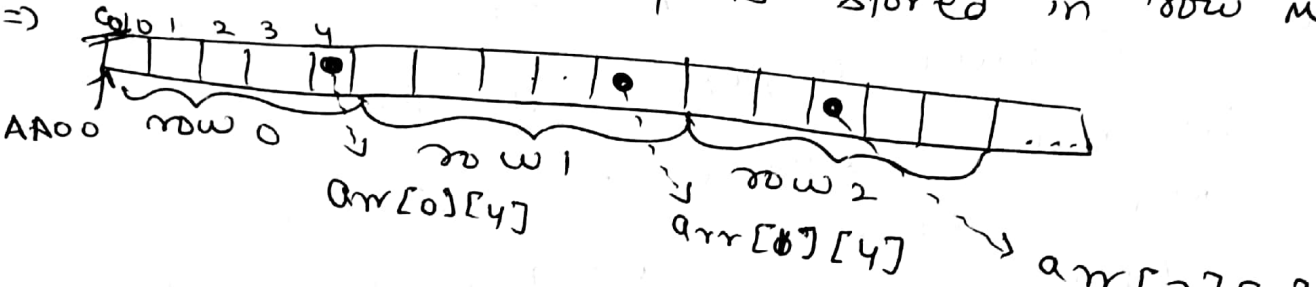
```
SUM = 0
for (int row = 0; row < n; row++)
{
    for ( int col = 0; col <m; col++)
        {
            SUM + = arr [row] [col]
        }
}
return SUM/(m*n);
```

for Poor locality of reference. (II)

```
SUM = 0
for (int col = 0; col < m; col++)
{
    for (int row = 0; row < n; row++)
    {
        SUM += arr[row][col];
    }
}
return SUM / (m*n);
```

In C, the 2D array is stored in row major

=>



So, in Pseudo-code (I)

Say, a cache ~~block~~ can hold 5 ints

=> we will get a hit rate of 4/5

or ≥ 80%

where as in pseudo-code (II)

we will always get a miss

because ~~arr[0][0]... one~~ access sequence is arr[0][0], arr[1][0] ... => we always get a miss

In case of Fortran, 2D array is stored in column major form.

=) pseudo code (II) will have hit rate of 80% where as code (I) will always have a miss if cache block can hold 5 ints and our array is say 3X5

In general, for C:

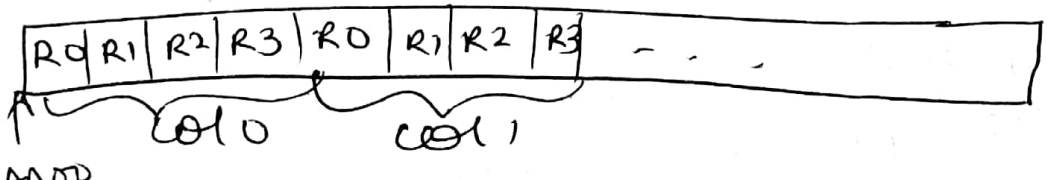
if cache block can hold elements < # of cols, we will have all Miss

Even if cache block can hold > # of cols, we will have a poor hit rate

on the other hand, we will be able to make effective use of cache using code (I)

The reverse holds for fortran

In fortran:



where R_i is Row i

m C B E S t s b

32 4096 16 1 256 20 8 4

Size of int = 4

data path = 16

⇒ 4 ints in each block.

$S_0 - S_{255}$ cache sets.

Starting address = 0x88002008

⇒ 1000 1000 0000 0000 0010 | 0000 0000 | 1000

Cache Sets

$S_0 = (a_0, a_1)$ $(a_{1022-1025})$ $(a_{2046-2048})$

$S_1 = (a_{2-5})$ $(a_{1026-1029})$

$S_{255} = (a_{1018-1021})$ $(a_{2042-45})$

```
for (i=0; i<2048; i++)
{
    sum += a[i];
}
return sum;
```

- Total RAM accesses
= 256 $(a_0 \ldots 1021)$
+ 256 $(a_{1022} \ldots 2045)$
+ 1 $(a_{2046} \ldots 2048)$
513 Ans