4

Question 1: Assuming a 32-byte memory organized as 2 banks of 16 bytes each.

(a) Show the bank major then row major ordering of the 16 bytes

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Bank 0

16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

Bank 1

Example:
Where is byte 25?

25 = 11001
Colum 01 (25 mod 4 = 1)
Row 10 (25/4 mod 4 = 6 mod 4 = 2)
Bank 1 (6/4 = 1)

(b) Show the bank major then column major ordering of the 16 bytes

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Bank 0

16	20	24	28
17	21	25	29
18	22	26	30
19	23	27	31

Bank 1

5

Question 2: Assume a 2MByte memory chip with 4 banks:

a) Determine the number of bits, n, needed to address a byte?
   n = 21 bits

b) How many address lines are needed to interface with this memory assuming that the bank+row address will be sent to memory first, followed by the column address. Note that the cell array in each bank should be as square as possible.
   512KB per bank → 2^{11} rows x 2^{8} columns → need 11+2 = 13 address bits

c) For "Bank major then row major ordering", If a byte address is b_{n-1}, b_n, … , b_1, b_0, where n is determined in part a, which bits are used for
   - bank address: b_{20}, b_{19}
   - row address: b_{18}, …. , b_8
   - column address: b_7, ….. , b_0

d) Repeat part c assuming "Row major then column major ordering":
   - row address: b_{20}, … , b_{10}
   - column address: b_9, …. , b_2
   - bank address: b_1, b_0

Question 3: Assume a 512KByte memory chip with 4 banks, each with 1024 rows and 128 columns of bytes.

a) For "Bank major then row major ordering", find (in decimal) the bank, row and column addresses for the byte whose address is 1054 (decimal)

#of bytes per bank:	128 * 1024
bank address:	1054 / (128*1024) = 0
address within bank:	1054 mod (128*1024) = 1054
#of bytes per row	128
row address:	1054 / 128 = 8
address within row:	1054 mod 128 = 30
column address:	30

Question 4: This question compares the *open row* and *closed row* DRAM access policy. Assume the following:

- it takes 10 cycles to activate (open) a row,
- it takes 3 cycles to access a column, and
- it takes 8 cycles to close a row,
- *r* is the probability that a memory reference is to a row that is already open,
- *1-r* is the probability that the reference is to a row that is not open.

a) What is the average memory access times (in terms of *r*) corresponding to the two policies (open row and closed row) assuming that accesses are separated by a relatively large number of cycles?

Definition: memory access time is the number of cycles between the issue of the memory request until the byte is returned from the row buffer.

Question 4:

- it takes 10 cycles to activate (open) a row,
- it takes 3 cycles to access a column, and
- it takes 8 cycles to close a row,
- *r* is the probability that a memory reference is to a row that is already open,
- *1-r* is the probability that the reference is to a row that is not open.

a) What is the average memory access times (in terms of *r*) corresponding to the two policies (open row and closed row) assuming that accesses are separated by a relatively large number of cycles?

Closed row policy: Average access time (in number of cycles) = 10(activate row) + 3(read column)
= 13 (in cycles)

Open row policy:

Average access time = { 3, if access is to an open row
8{close row} + 10{activate} + 3{read column} = 21, if access is to a closed row

Average access time = 3 * r + 21 * (1-r) = 21 - 18 * r (in cycles)

This is only true when a new access is separated from the previous access by at least 8 cycles (not back to back accesses) to avoid the penalty of closing the row in the closed row policy.

Question 4:

- it takes 10 cycles to activate (open) a row,
- it takes 3 cycles to access a column, and
- it takes 8 cycles to close a row,
- *r* is the probability that a memory reference is to a row that is already open,
- *1-r* is the probability that the reference is to a row that is not open.

a) What is the average memory access times (in terms of *r*) corresponding to the two policies (open row and closed row) assuming that accesses are separated by a relatively large number of cycles?

b) For what value of the row buffer hit rate (*r*) will you chose one policy over the other to get the best average access time?

Open row policy: Average access time = 21 - 18 * r

Closed row policy: Average access time = 13

The open row policy is more efficient when 21-18*r < 13. That is when r > 8/18 = 44.4%.

Question 4:

- it takes 10 cycles to activate (open) a row,
- it takes 3 cycles to access a column, and
- it takes 8 cycles to close a row,
- *r* is the probability that a memory reference is to a row that is already open,
- *1-r* is the probability that the reference is to a row that is not open.

c) What will be your answer to the above questions if accesses are consecutive in the sense that a memory access is issued as soon as the previous access is satisfied

If accesses are back to back, then the "open row policy" will not be affected, but the closed row policy will be. Specifically, the access time for the "closed row policy" will be 21 cycles since the new access will always have to wait while the row for the previous access is being closed. Hence

Open row policy: Average access time = 21 - 18 * r

Closed row policy: Average access time = 21

So, the "open row policy" is more efficient when 21-18r < 21, which is always true.