13) Compare and contrast the application of restoring and non restoring division method for an example $15/2$

Non-restoring algorithms

$Q \Rightarrow (15)_{10} \Rightarrow (1111)_2$

$M \Rightarrow (2)_{10} \Rightarrow (00010)_2$

$\bar{M} \Rightarrow 11101$
$\quad\quad\; + \;\; 1$
$\quad\quad\; 11110$

	Accumulator	Q	Count
	0 0 0 0 0	1 1 1 1	4
A>0)	0 0 0 0 1	1 1 1 $\square$	
	1 1 1 1 0		
A-M)	1 1 1 1 1	1 1 1 $\boxed{0}$	
A<0)			
			3
A<0) left shift	1 1 1 1 1	1 1 0 $\square$	
M	0 0 0 1 0		
A+M	0 0 0 0 1	1 1 0 $\boxed{1}$	
A>0)			
A>0) left shift	0 0 0 1 1	1 0 1 $\square$	2
	1 1 1 1 0		
A-M	0 0 0 0 1	1 0 1 $\boxed{1}$	
A>0			
A>0) Shift left	0 0 0 1 1	0 1 1 $\square$	1
	1 1 1 1 0		
A-M)	0 0 0 0 1	0 1 1 $\boxed{1}$	
A>0			
			0

Remainder $\Rightarrow (00001)_2 \Rightarrow (2)_{10}$

Quotient $\Rightarrow (0111)_2 \Rightarrow (7)_{10}$

Restoring division algorithm

Dividend (Q) ⇒ 15 ⇒ 1111

Divisor (B) ⇒ 2 ⇒ 00010

$\overline{B}$ ⇒ 11101

$$\begin{array}{r} 11101 \\ 1 \\ \hline 11110 \end{array}$$

	Accumulator	Q	n
	00000	1111	4
Shift left	00001	111$\boxed{\ }$	
	11110		
A-B	11111	111$\boxed{0}$	
A+B	00010		3
(restore)	00001	1110	
Shift left	00011	110$\boxed{\ }$	
	11110		2
A-B	00001	110$\boxed{1}$	
Shift left	00011	101$\boxed{\ }$	
	11110		1
A-B	00001	101$\boxed{1}$	
Shift left	00111		
	11110		
A-B	00001	011$\boxed{1}$	0

Quotient ⇒ $(0111)_2$ ⇒ 7

Remainder ⇒ $(00001)_2$ ⇒ 1

*) In restoring method, we add the divisor when the $Q_0 \Rightarrow 0$

*) In Non-restoring method, we do not add the divisor if we set $q_0 \Rightarrow$ as either 0 or 1. thereby reducing the count and proceed the algorithm.

3) Solve 100011110 / 10111 by using restoring and non-restoring method with the support of algorithm.

Restoring division algorithm:

$\bar{M} \Rightarrow 01000$
$\quad\quad\quad\quad\quad 1$

$Q \Rightarrow (100011110)_2$

1111101001) M 2's complement

$M \Rightarrow (10111)_2 \Rightarrow (0000010111)_2$

	Accumulator	Q	count
	0000000000	100011110	9
left shift	0000000001	00011110 ☐	
	1111101001		
A−M	1111101010	000111100	
A<0	0000010111		
A+M (restore)	0000000001	000111100	8
left shift	0000000010	00111100 ☐	
	1111101001		
A−M	1111101011	001111000	
A<0	0000010111		
A+M (restore)	0000000010	001111000	7

left shift	0 0 0 0 0 0 0 1 0 0	0 1 1 1 1 0 0 $\square$	
A-M	1 1 1 1 1 0 1 0 0 1		
	1 1 1 1 1 0 1 1 0 1	0 1 1 1 1 0 0 $\boxed{0}$	
A<0			
A+M	0 0 0 0 0 1 0 1 1 1		
(restore)	0 0 0 0 0 0 0 1 0 0	0 1 1 1 1 0 0 0	6
left shift	0 0 0 0 0 0 1 0 0 0	1 1 1 1 0 0 $\square$	
A-M	1 1 1 1 1 0 1 0 0 1		
	1 1 1 1 1 1 0 0 0 1	1 1 1 1 0 0 $\boxed{0}$	
A<0			
A+M	0 0 0 0 0 1 0 1 1 1		
(restore)	0 0 0 0 0 0 1 0 0 0	1 1 1 1 0 0 0	5
left shift	0 0 0 0 0 1 0 0 0 1	1 1 1 0 0 0 $\square$	
A-M	1 1 1 1 1 0 1 0 0 1		
	1 1 1 1 1 1 1 0 1 0	1 1 1 0 0 0 $\boxed{0}$	
A<0			
A+M	0 0 0 0 0 1 0 1 1 1		
(restore)	0 0 0 0 0 1 0 0 0 1	1 1 1 0 0 0 0	4
left shift	0 0 0 0 1 0 0 0 1 1	1 1 0 0 0 0 $\square$	
A-M	1 1 1 1 1 0 1 0 0 1		
	0 0 0 0 0 0 1 1 0 0	1 1 0 0 0 0 $\boxed{1}$	3
A>0			

	0 0 0 0 0 1 1 0 0 1	1 0 0 0 0 1 $\boxed{\ }$
left shift	1 1 1 1 1 0 1 0 0 1	2
A − M	0 0 0 0 0 0 0 0 1 0	1 0 0 0 0 0 1 $\boxed{1}$
A > 0		
left shift	0 0 0 0 0 0 0 1 0 1	0 0 0 0 0 1 1 $\boxed{\ }$
	1 1 1 1 1 0 1 0 0 1	0 0 0 0 0 1 1 $\boxed{0}$
A − M	1 1 1 1 1 0 1 1 1 0	
	0 0 0 0 0 1 0 1 1 1	0 0 0 0 1 1 0 1
A + M (restore)	0 0 0 0 0 0 0 1 0 1	
left shift	0 0 0 0 0 0 1 0 1 0	0 0 0 1 1 0 $\boxed{\ }$
	1 1 1 1 1 0 1 0 0 1	0 0 0 1 1 0 $\boxed{0}$
A − M	1 1 1 1 1 1 0 0 1 1	
	0 0 0 0 0 1 0 1 1 1	0 0 0 1 1 0 0
A < 0	0 0 0 0 0 0 1 0 1 0	

Quotient ⇒ 00001100 ⇒ 1100

Remainder ⇒ 00000 01010 ⇒ 1010

Non-restoring division algorithm

	Accumulator	Q	
	0 0000 00000	1 000 1111 0	9
left shift	0 0000 00001	0001 1110 $\square$	
	11 111 01 001		
A − M	11111 01 01 0	0001 1110 $\boxed{0}$	
A < 0			8
left shift	1111 01 010 0	00111 00 $\square$	
A + M	0000 0 1 0111		
	11111 010 11	00 1111 00 $\boxed{0}$	7
A < 0			
left shift	1111 0 1 0 110	01111 000 $\square$	
	00000 1 0 111		
A + M	11111 01 101	01111 000 $\boxed{0}$	6
A < 0			
left shift	1111 0 11 010	111 0000 $\square$	
A + M	00000 1 0111		
	11111 10001	1111 0000 $\boxed{0}$	5
A < 0			

④

left shift	11111 00011	11100000 $\boxed{\ }$
A + M	00000 10111	
	11111 11010	11100000 $\boxed{0}$
A < 0		
left shift	11111 10101	11000000 $\boxed{\ }$
A+M	00000 10111	
	00000 01100	11000000 $\boxed{1}$ ③
A > 0		
left shift	00000 11001	10000001 $\boxed{\ }$
	11111 01001	
A-M	00000 00010	10000001 $\boxed{1}$
A > 0		②
left shift	00000 00101	00000011 $\boxed{\ }$
A-M	11111 01001	
	11111 01110	00000011 $\boxed{0}$ ①
A < 0		
left shift	1111 011100	00000110 $\boxed{\ }$
A+M	0000 010111	
	1111 110011	00000110 $\boxed{0}$
A < 0		

0

A+M 1 1 1 1 1 1 0 0 1 1

0 0 0 0 0 1 0 1 1 1

0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 0

Quotient ⇒ 000001100 ⇒ 1100

Remainder ⇒ 0000001010 ⇒ 1010

5) Discuss the algorithm for multiplication of (-8 * -14) how the signed magnitude is handled.

*) The signed magnitude representation of the binary number must have either 0/1.

*) For +ve binary number, MSB ⇒ 0

*) For -ve binary number, MSB ⇒ 1

*) Otherwise, we can take 2's complement for -ve numbers with sign bit so that we can get signed binary number for -ve answers.

QR ⇒ multiplier ⇒ -14 ⇒ 10010

BR ⇒ multiplicand ⇒ -8 ⇒ 11000

$\overline{BR}$ ⇒ 00111
1
01000

	Accumulator	QR	Q_{n+1}	SC
	00000	10010	0	5
Shift right	00000	01001	0	4
AC ⇒ AC-BR	01000 01000	01001	0	
Shift right	00100	00100	1	3
AC ⇒ AC+BR	~~0~~1000 11100	00100	~~0~~	
Shift righ	11110	00010	0	2

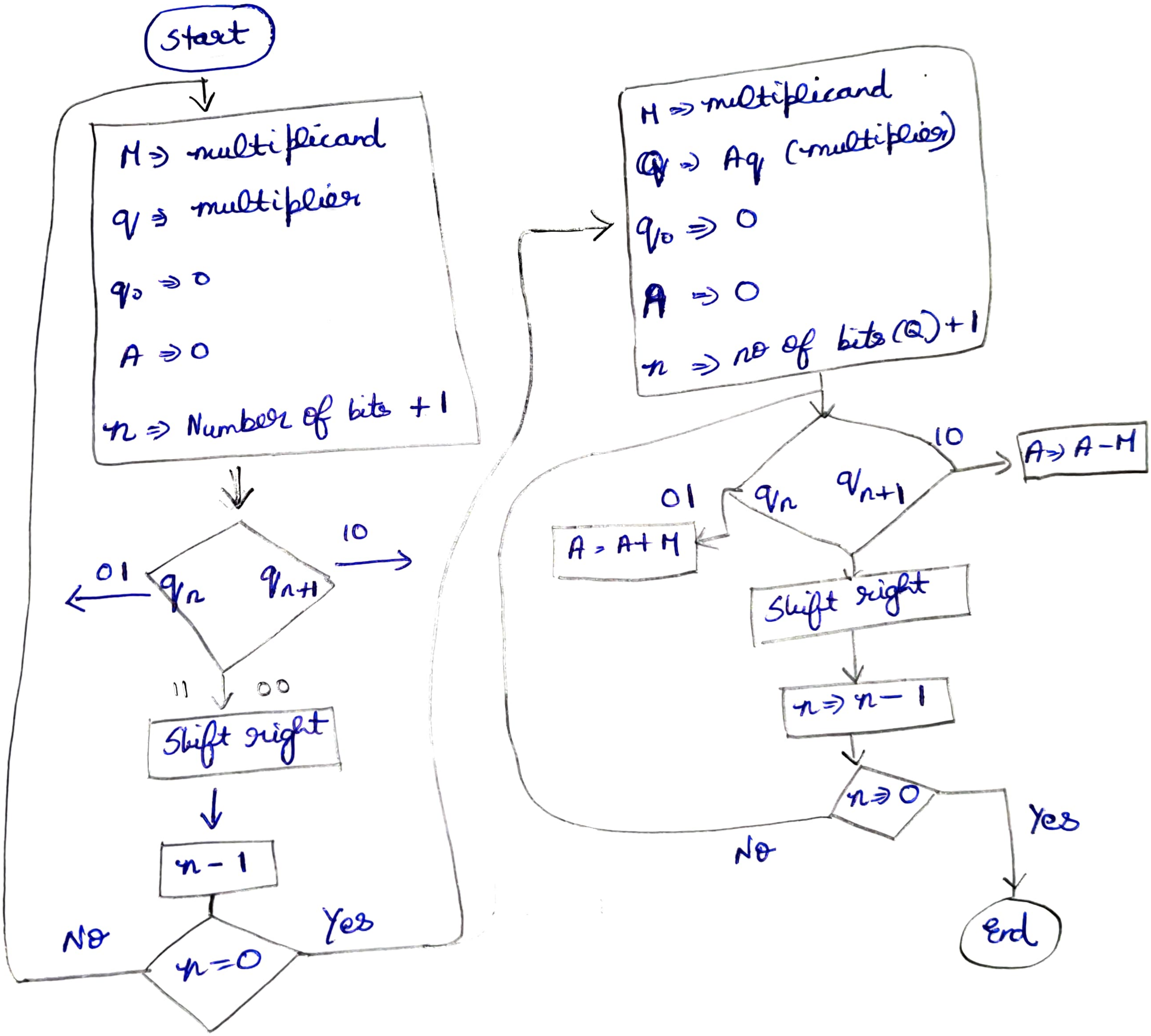
Shift right	11111	00001	0	1
AC → AC−BR	01000 00111	00001		
Shift right	00011	10000	1	0

10010 * 11000 ⇒ 0001110000

−8 * −14 → 112

10) What is the multiplication to be done on booth multiplication algorithm when 3 numbers given multiplication (1000 * 1001 * 0111)

Algorithm



11) Justify "Replacement algorithm not used in cache memory". Any other mechanism can be adopted?

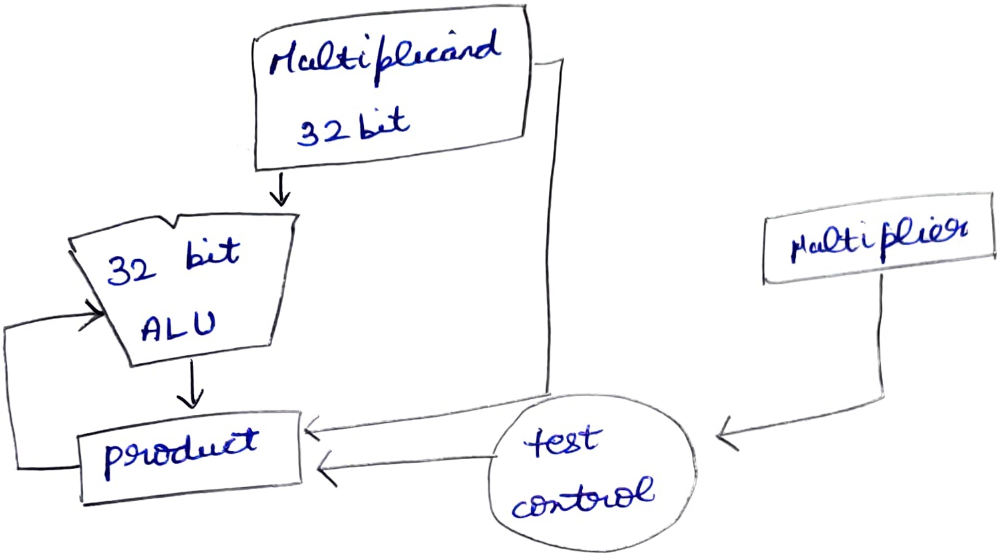
The replacement algorithm which is not used in cache memory is "~~optional~~ optimal algorithm". Because it is futuristic approach of page replacement which is just theoretical. The future pages which is approaching could not be determined and hence it is atmost impossible to implement.

<u>Other mechanisms :</u>

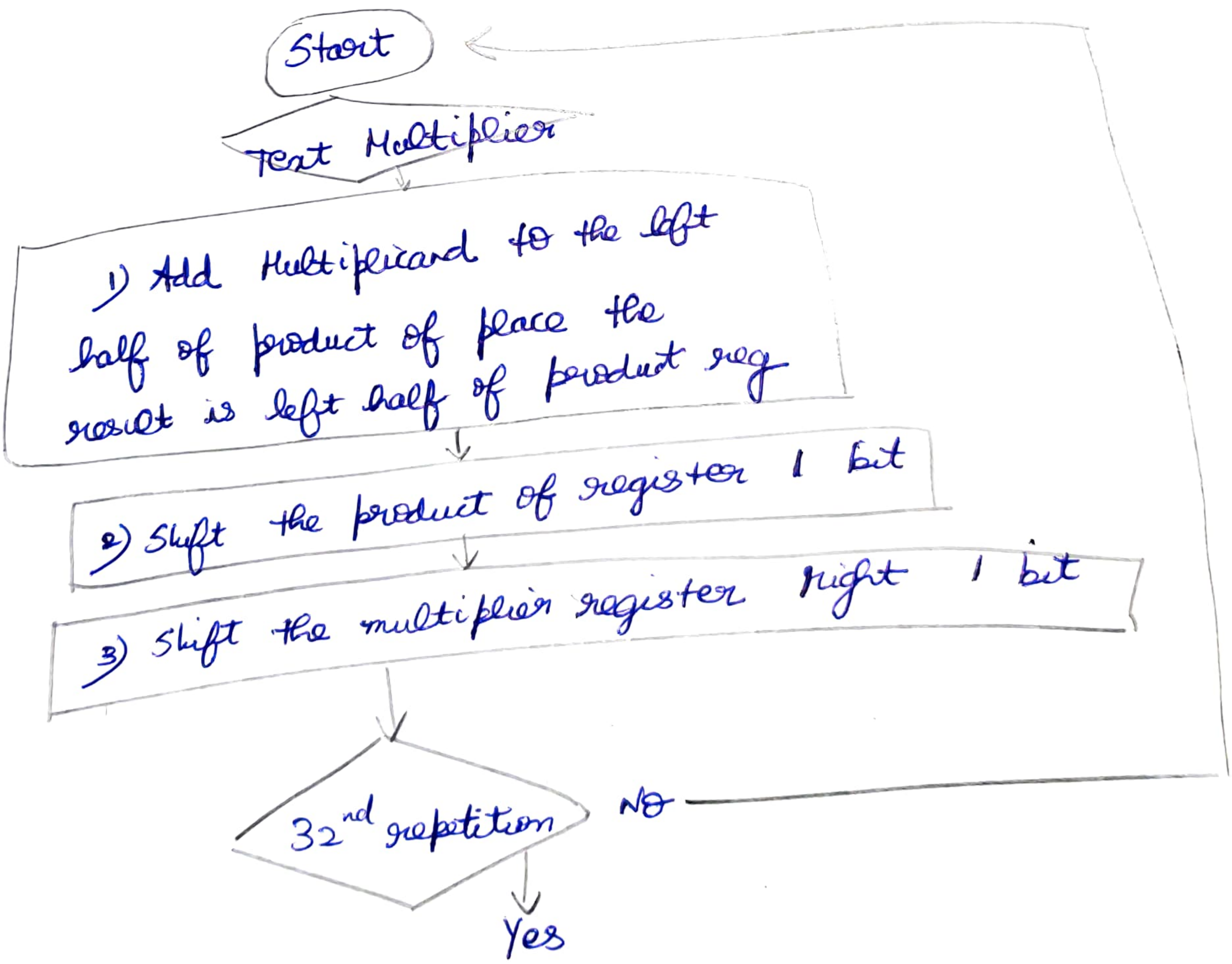
FIFO ⇒ This page replacement replaces the firstly entered page among the presently existing page.

LRU ⇒ This page replacement algorithm replaces the least recently used page if it is not in cache memory.

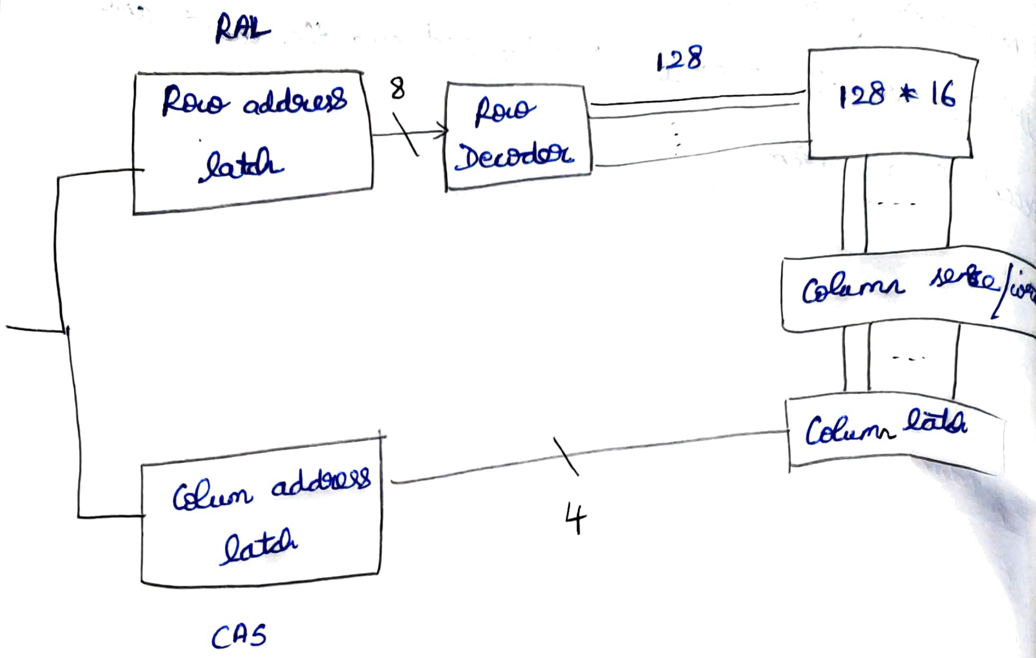
12) Design a hardware and algorithm for multiplying 32 bit * 32 bit



Algorithm :



18) Design a DRAM of 128 * 16



Row address:

↳ Set row address on address lines strobe RAS.

↳ Entire row read and stored in column latch.

↳ Contents of row of memory all destroyed.

Column address:

↳ Set colum address on address line and strob CAS

↳ Access selected bit

Read: transfers from selected colum latch to Dout.

Write: selected column latch to Din.

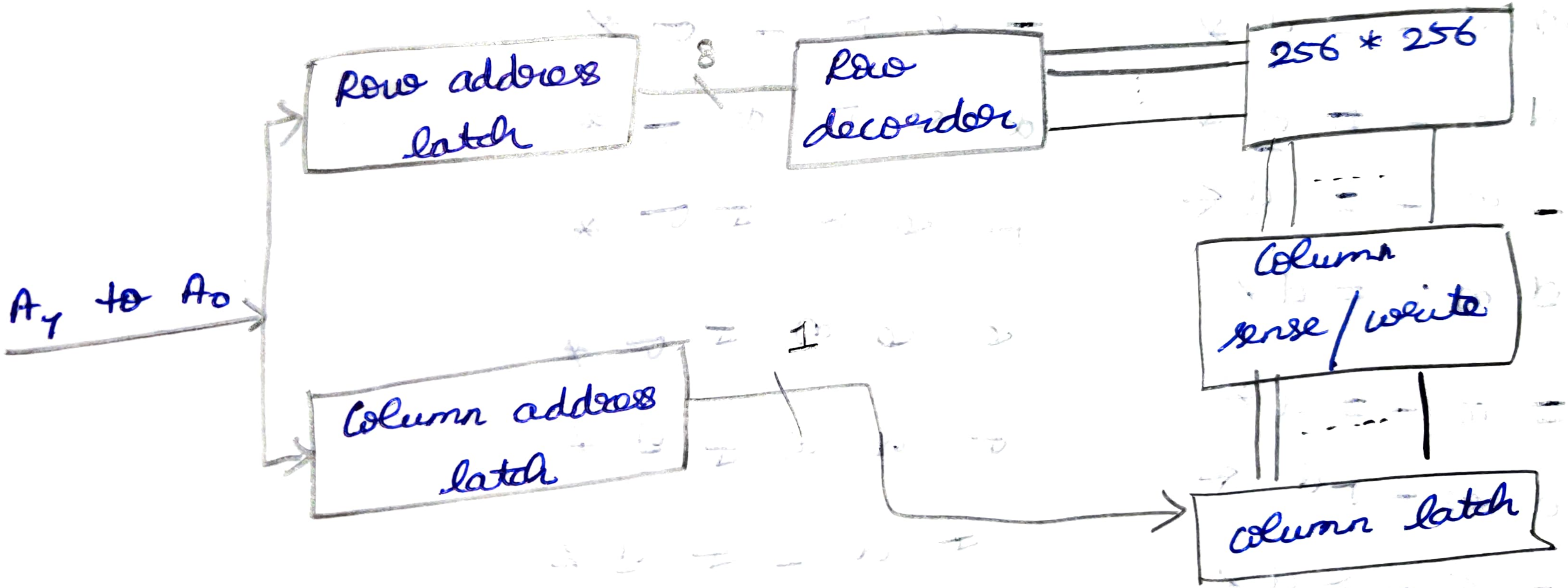
Rewrite: write back on entire row.

Conventional access: Row + Column
(RAS) + (CAS)

Page mode: Row + Series of columns
Gives successive bit

17). Design a hardware for storing a 6 bit number in the dynamic RAM

6 bit $\Rightarrow 2^6 \rightarrow 64k * 1$



16) Cache memory with the size of 4 KB (4 lines) and virtual memory with the size of 8 KB (lines) following reference string is needed for the execution.

Before the execution of all programs, it a taken into the main memory and then to cache memory to execute.

Virtual memory ⇒ 8 KB

Main memory ⇒ 4 KB

1	1				*
2	1	2			*
1	1	2			↑
2	1	2			↑
3	1	2	3		*
4	1	2	3	4	*
5	5	2	3	4	*
6	5	6	3	4	*
8	5	6	8	4	*
9	5	6	8	9	*
10	10	6	8	9	*
12	10	12	8	9	*
11	10	12	11	9	*
8	10	12	11	8	*
9	9	12	11	8	*
6	9	6	11	8	*
7	9	6	7	8	*
2	9	6	7	2	*
3	3	6	7	2	*
1	3	1	7	2	*
2	3	1	7	2	↑
11	3	1	11	2	*
12	3	1	11	2	*
1	3	1	11	12	↑
2	3	1	11	12	*
3	2	1	11	12	*
1	6	7	1	9	*
8	6	7	8	9	*
7	6	7	4	9	*
6	6	3	4	9	*
9	2	3	4	12	*
4	2	1	4	12	*

Algorithm used ⇒ First in First out

Hit ⇒ 4

Page fault ⇒ 28

Hit ratio ⇒ $4/32 \Rightarrow 0.125$

Page fault ratio ⇒ $28/32 \Rightarrow 0.875$