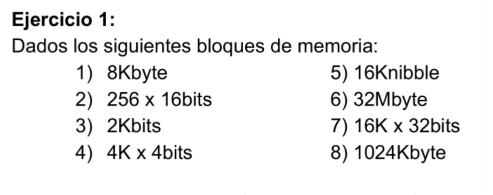
Contents

1)



Ejercicio 1:
Dados los siguientes bloques de memoria:

1) 8Kbyte	5) 16Knibble
2) 256 x 16bits	6) 32Mbyte
3) 2Kbits	7) 16K x 32bits
4) 4K x 4bits	8) 1024Kbyte

1) $8K \cdot 8 \text{bits} = 2^3 . 2^{10} \cdot 2^3 \text{ bits} = 2^6 . 2^{10} \text{bits}$

2) $256 \cdot 16 \text{bits} = 2^8 \cdot 2^4 \text{ bits} = 2^{10} . 2^2 \text{ bits}$

3) 2Kbits

4) $4K \cdot 4 \text{bits} = 2^2 . 2^{10} \cdot 2^2 \text{ bits} = 2^4 . 2^{10} \text{ bits}$

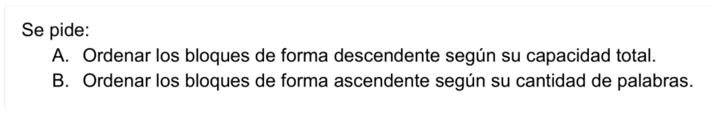
5) $16K . 4 \text{bits} = 2^4 . 2^{10} . 2^2 \text{bits} = 2^6 . 2^{10} \text{ bits}$

6) $32 \text{Mbyte} = 2^5 . 2^{20} . 2^3 \text{ bits} = 2^8 . 2^{20}$

7) $16K \cdot 32 \text{bits} = 2^4 . 2^{10} . 2^5 \text{bits} = 2^9 . 2^{10} \text{bits}$

8) $1024 \text{kbyte} = 2^{10} . 2^{10} . 2^3 \text{ bits} = 2^3 . 2^{20} \text{ bits}$

Cant.	Cant. Total	Cant. Palabras	Ord.	Ord.
8kbyte	64 Kbits	2^{13}	4	4
256 . 16bits	4 K bits	2^8	7	1
2k bits	2kbits	2^{11}	8	2
4k . 4bits	16 Kbits	2^{12}	6	3
16Knibble	64 kbits	2^{14}	5	5
32Mbyte	256 Mbits	2^{25}	1	8
16K · 32bits	512K bits	2^{14}	3	6
1024Kbyte	8 Mbits	2^{20}	2	7



Se pide:
A. Ordenar los bloques de forma descendente según su capacidad total.
B. Ordenar los bloques de forma ascendente según su cantidad de palabras.

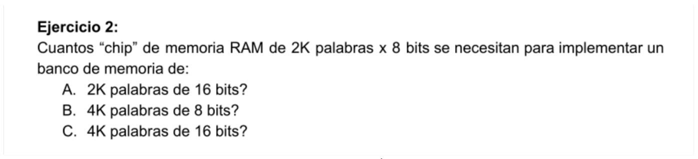
A)

1) 32Mbyte
2) 1024Kbyte
3) 16K * 32bits
4) 8Kbyte
5) 16Knibble
6) 4k * 4bits
7) 256 * 16bits
8) 2Kbits

B)

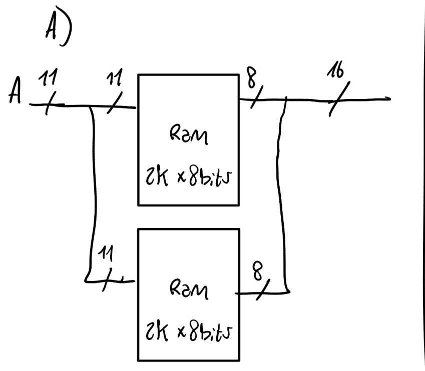
1) 256 * 16bits
2) 2Kbits
3) 4K * 4bits
4) 8Kbyte
5) 16Knibble
6) 16K * 32 bits
7) 1024K * byte
8) 32Mbyte

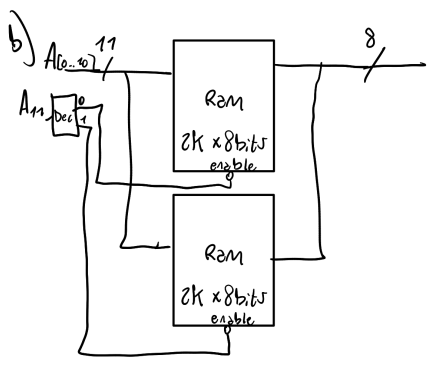
2)

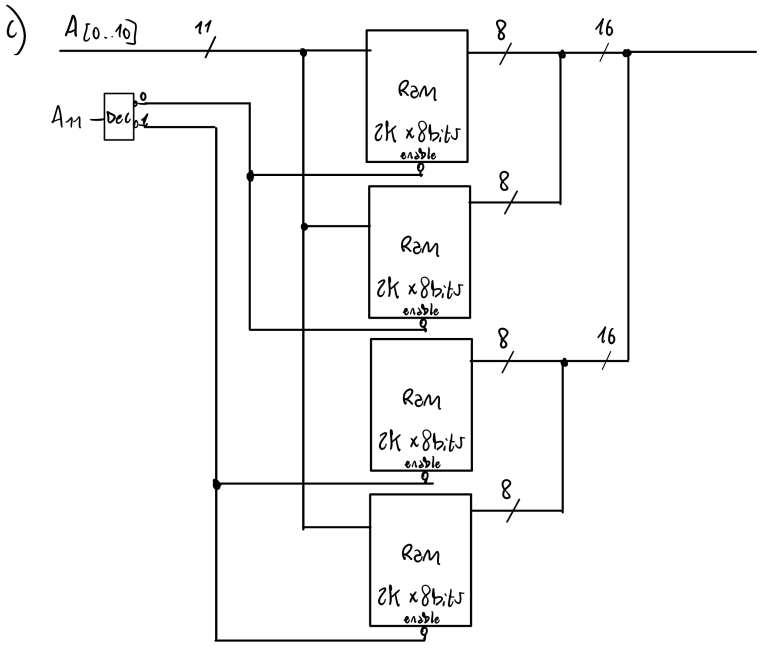


Ejercicio 2:
Cuantos "chip" de memoria RAM de 2K palabras x 8 bits se necesitan para implementar un banco de memoria de:

A. 2K palabras de 16 bits?
B. 4K palabras de 8 bits?
C. 4K palabras de 16 bits?







A)

2 en paralelo

B)

2 en serie

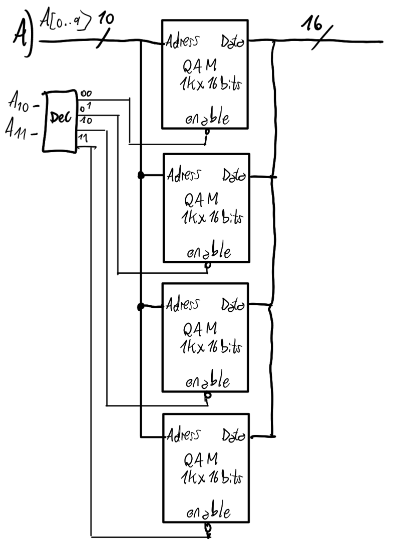
C)

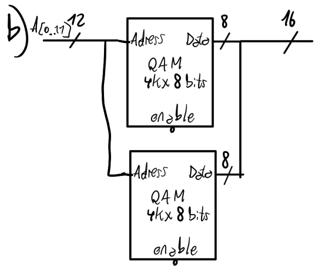
4, dos en paralelo y dos en serie

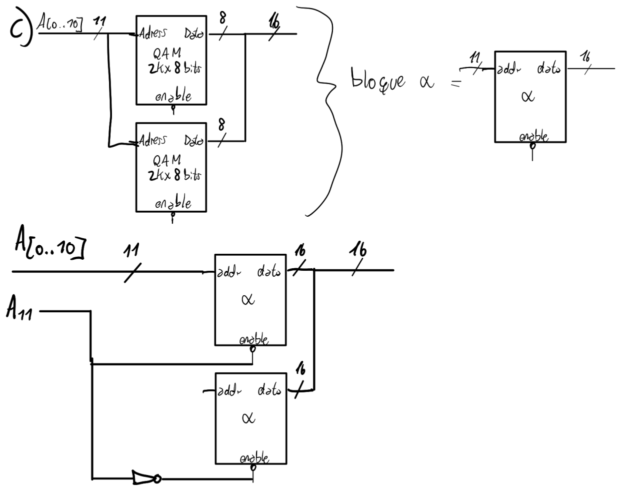
3

Ejercicio 3:
Construir un sistema de memoria RAM de 4K palabras de 16 bits mediante la utilización de “chips” de memoria de:

A. 1K palabras de 16 bits.
B. 4K palabras de 8 bits.
C. 2K palabras de 8 bits.

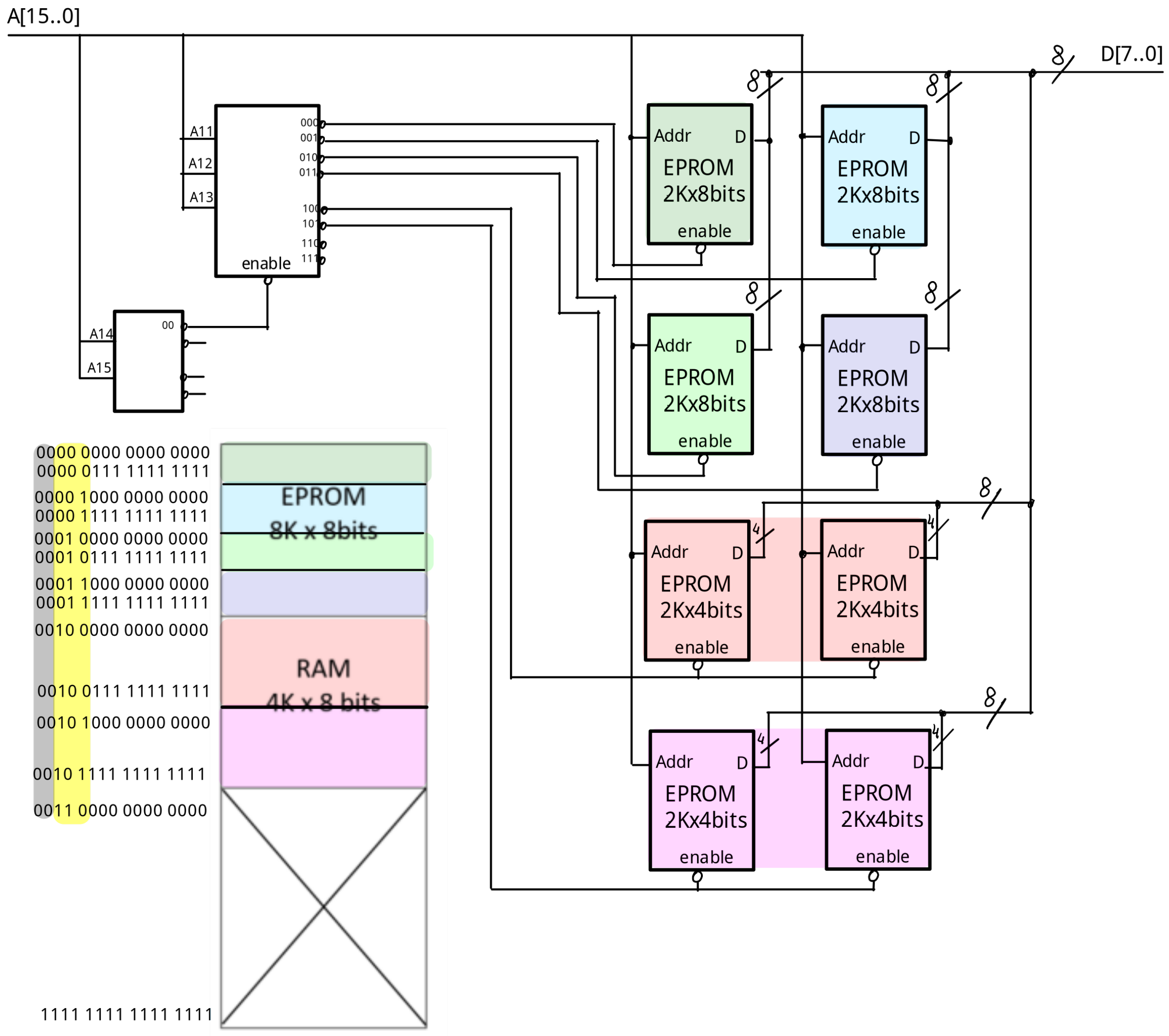




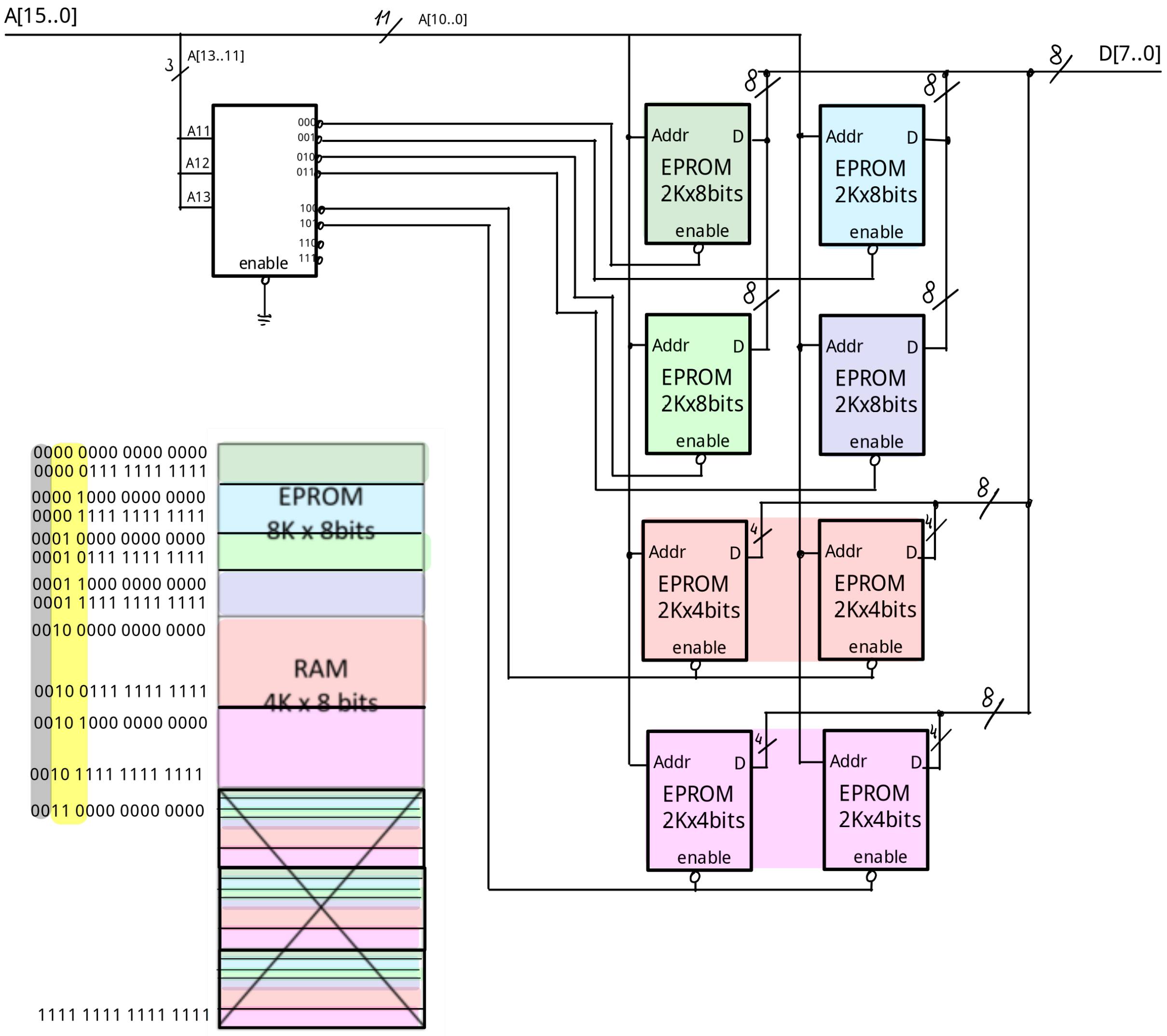


4)

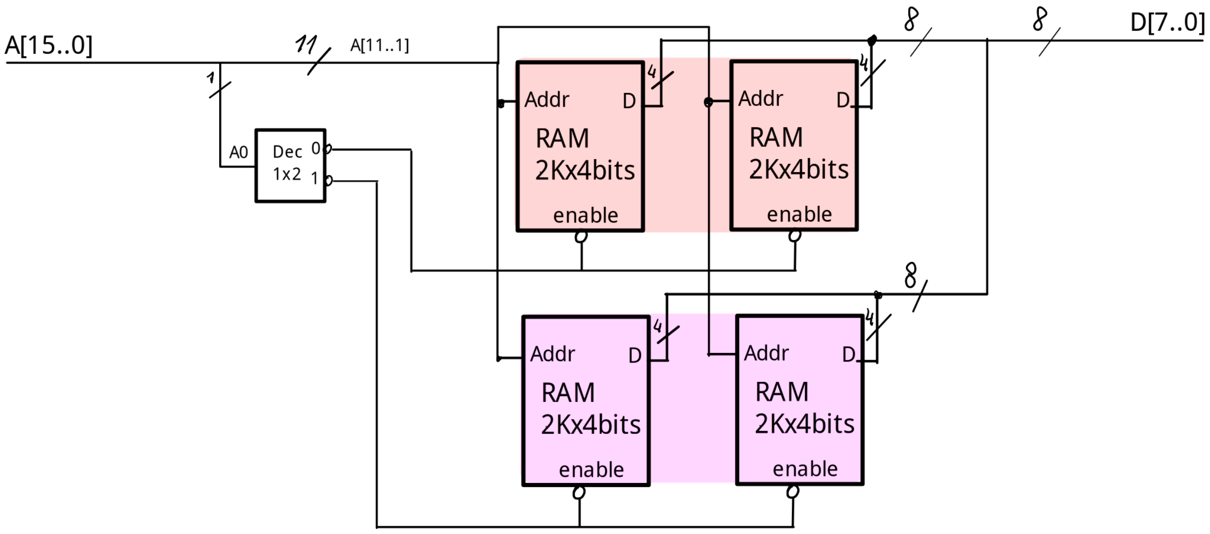
A)


A[15..0]
D[7..0]
8
A11
A12
A13
000
001
010
011
100
101
110
111
enable
A14
A15
00
Addr
D
EPROM
2Kx8bits
enable
4
EPROM
2Kx4bits
0000 0000 0000 0000
0000 0111 1111 1111
0000 1000 0000 0000
0000 1111 1111 1111
0001 0000 0000 0000
0001 0111 1111 1111
0001 1000 0000 0000
0001 1111 1111 1111
0010 0000 0000 0000
0010 0111 1111 1111
0010 1000 0000 0000
0010 1111 1111 1111
0011 0000 0000 0000
1111 1111 1111 1111
EPROM
8K x 8bits
RAM
4K x 8 bits

B)


A[15..0]
11
A[10..0]
3
A[13..11]
A11
A12
A13
000
001
010
011
100
101
110
111
enable
8
D[7..0]
Addr
D
EPROM
2Kx8bits
enable
Addr
D
EPROM
2Kx4bits
enable
4
0000 0000 0000 0000
0000 0111 1111 1111
0000 1000 0000 0000
0000 1111 1111 1111
0001 0000 0000 0000
0001 0111 1111 1111
0001 1000 0000 0000
0001 1111 1111 1111
0010 0000 0000 0000
0010 0111 1111 1111
0010 1000 0000 0000
0010 1111 1111 1111
0011 0000 0000 0000
1111 1111 1111 1111
EPROM
8K x 8bits
RAM
4K x 8 bits

5)

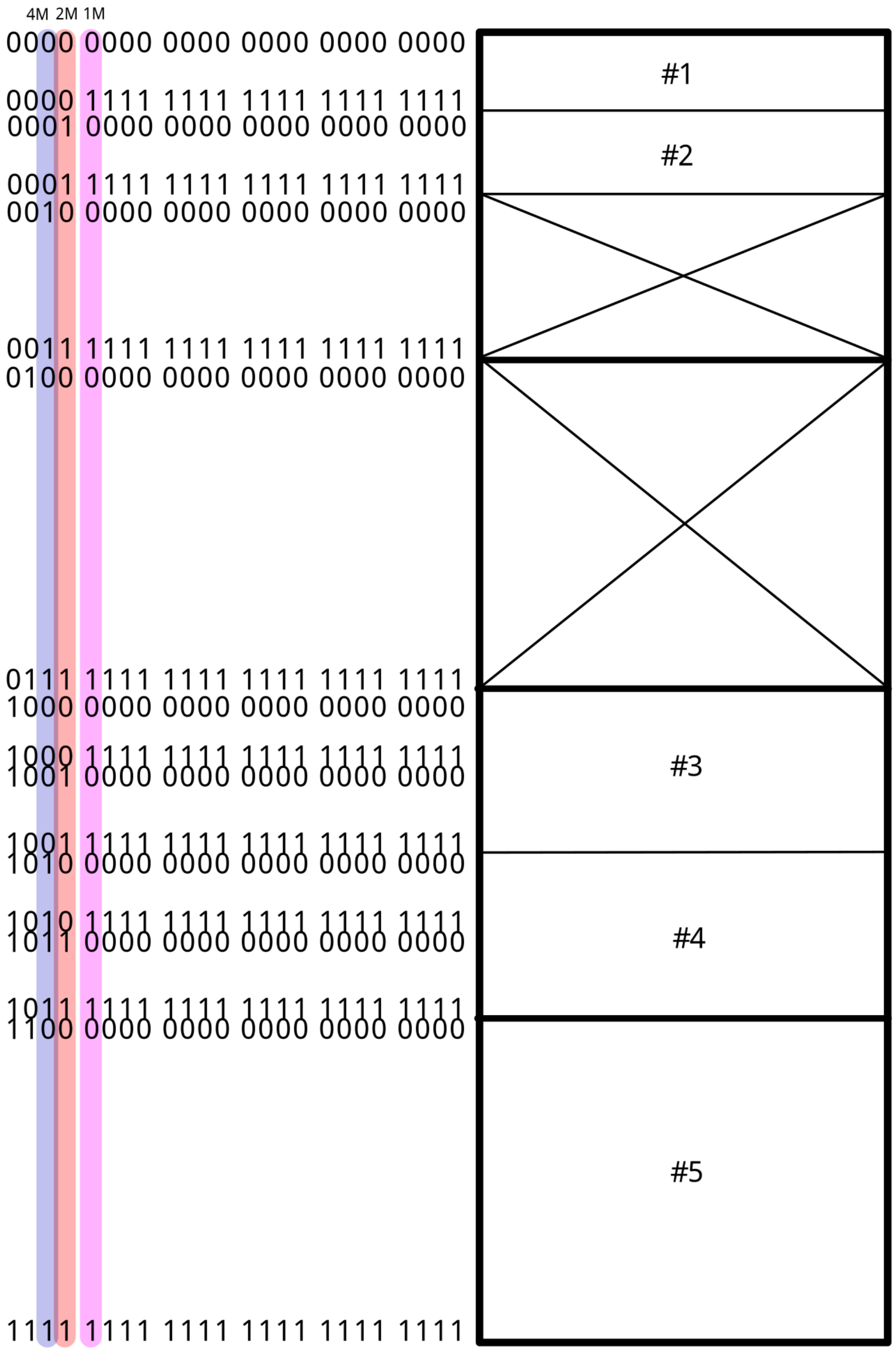


6)

A)

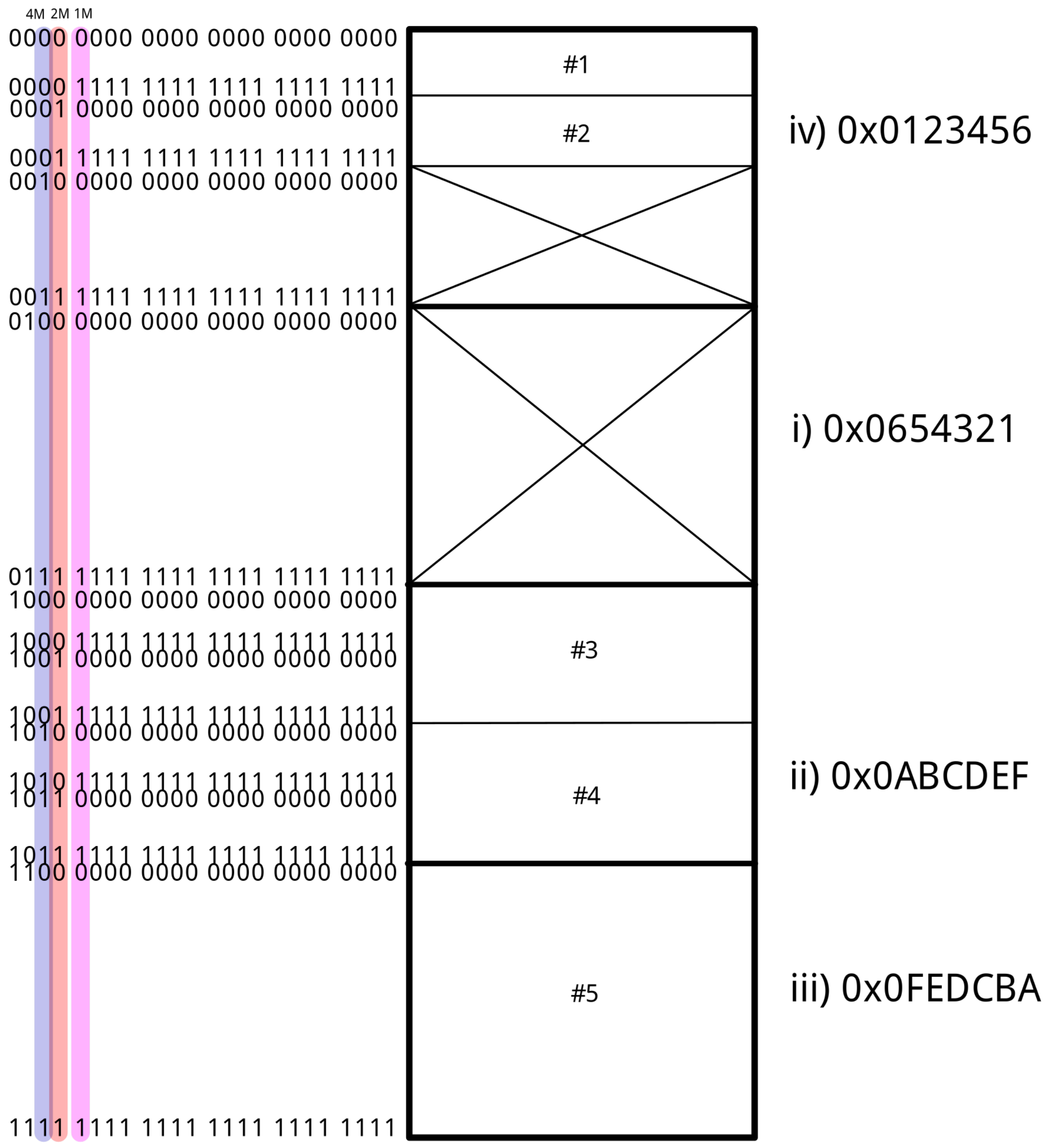
2^{24} * 16 bits
= 1M * 2^4 * 16 bits
= 1M * 16 * 16 bits
= 16Mx16bits

B)

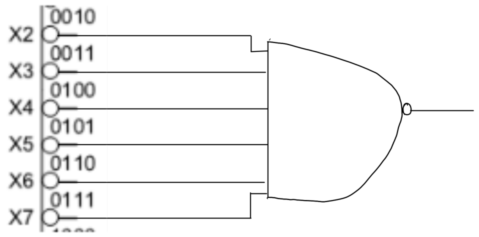

4M 2M 1M
0000 0000 0000 0000 0000 0000
0000 1111 1111 1111 1111 1111
0001 0000 0000 0000 0000 0000
0001 1111 1111 1111 1111 1111
0010 0000 0000 0000 0000 0000
0011 1111 1111 1111 1111 1111
0100 0000 0000 0000 0000 0000
0111 1111 1111 1111 1111 1111
1000 0000 0000 0000 0000 0000
1000 1111 1111 1111 1111 1111
1001 0000 0000 0000 0000 0000
1001 1111 1111 1111 1111 1111
1010 0000 0000 0000 0000 0000
1010 1111 1111 1111 1111 1111
1011 0000 0000 0000 0000 0000
1011 1111 1111 1111 1111 1111
1100 0000 0000 0000 0000 0000
1111 1111 1111 1111 1111 1111
#1
#2
#3
#4
#5

C)

i.	0x0654321	0110 0101 0100 0011 0010 0001
ii.	0x0ABCDEF	1010 1011 1100 1101 1110 1111
iii.	0x0FEDCBA	1111 1110 1101 1100 1011 1010
iv.	0x0123456	0001 0010 0011 0100 0101 0110
v.	0x2000000	0010 0000 0000 0000 0000 0000 0000



D)

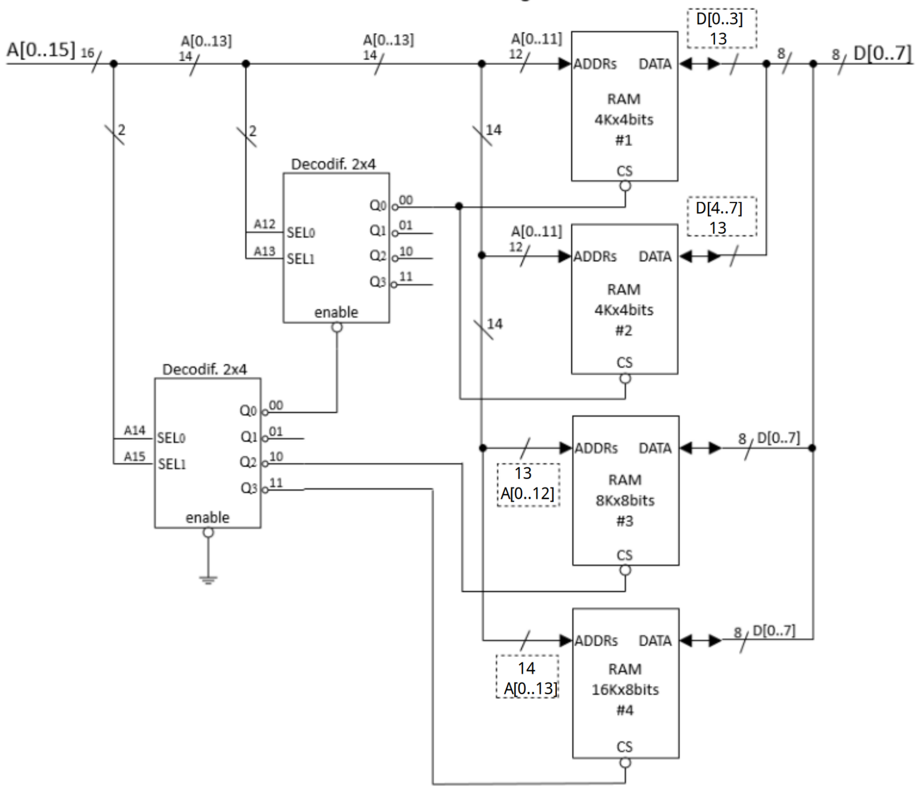


E)

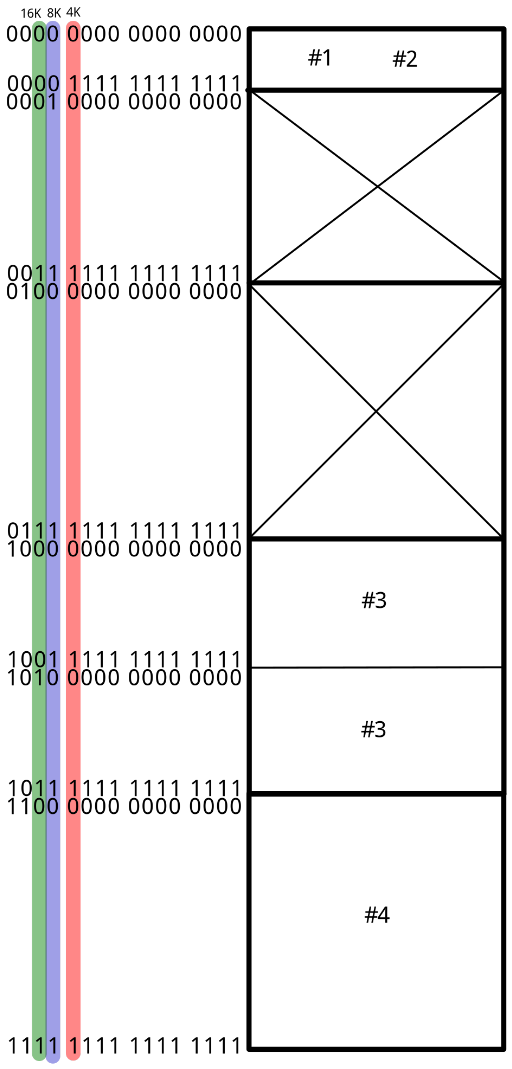
No, no genera posiciones imagen ya que nunca quedan bits del adress sin utilizar.

7)

A)



B)



C)

1 F

2 V

3 F

El procesador puede direccionar 64K palabras de 8bits

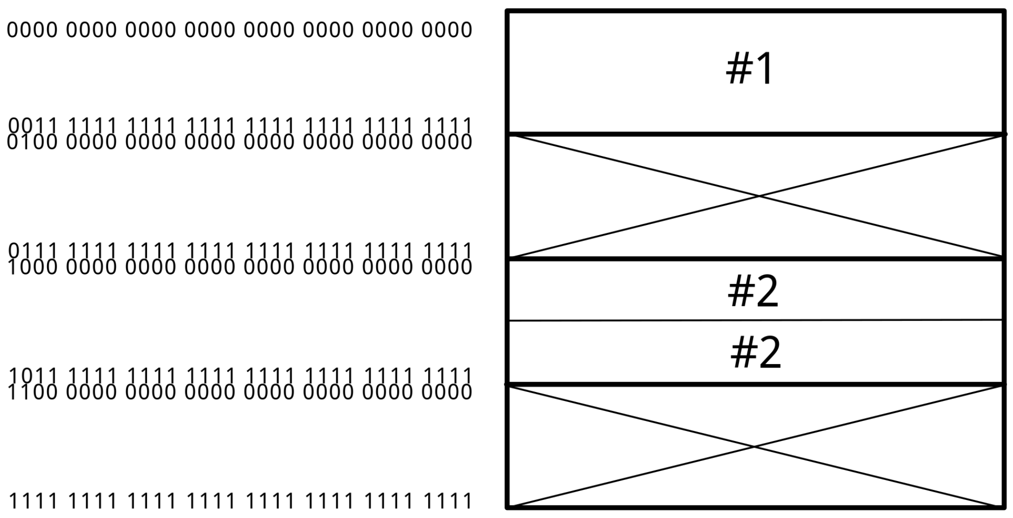
4 F

8)

A)

$$\begin{aligned}
&1Gx32bits + 512Mx32bits \\
&= 2^{30} * 32bits + 2^{20} * 2^{9} * 32bits \\
&= (2^{30} + 2^{20} * 2^{9}) * 32bits \\
&= (2^{30} + 2^{29}) * 32bits \\
&= (1610612736) * 32bits \\
&= 51539607552bits
\end{aligned}$$

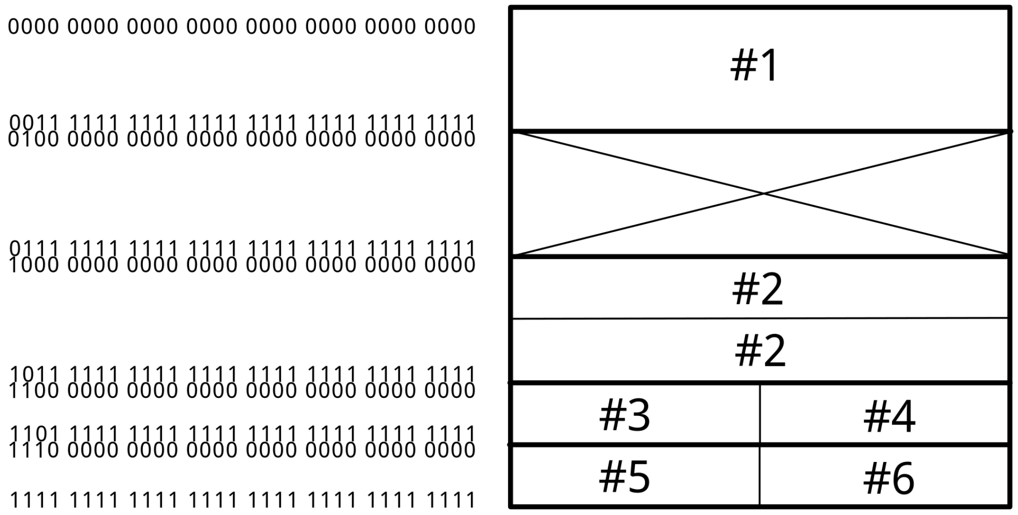
B)

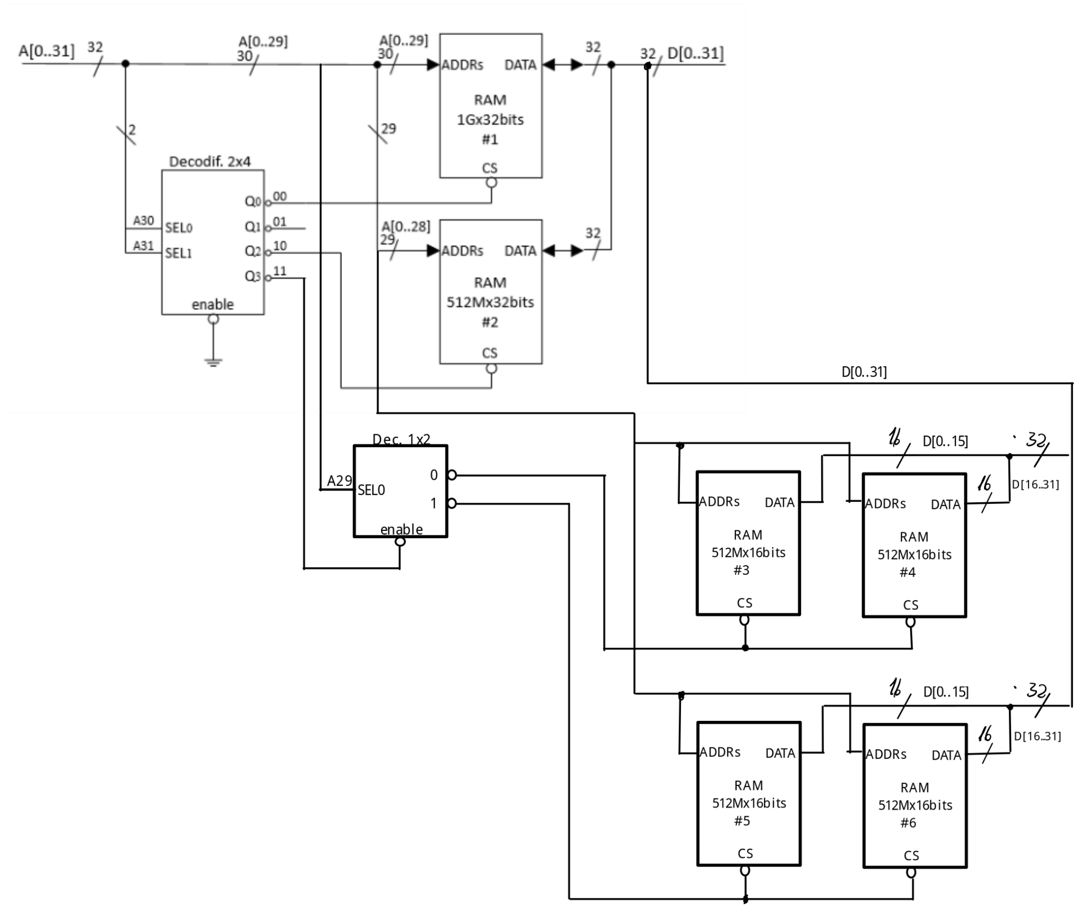


C)

Si, genera posiciones imagen en el bloque #2 que se encuentran en el rango 0x80000000 a 0xBFFFFFFF

D)


0000 0000 0000 0000 0000 0000 0000 0000
0011 1111 1111 1111 1111 1111 1111 1111
0100 0000 0000 0000 0000 0000 0000 0000
0111 1111 1111 1111 1111 1111 1111 1111
1000 0000 0000 0000 0000 0000 0000 0000
1011 1111 1111 1111 1111 1111 1111 1111
1100 0000 0000 0000 0000 0000 0000 0000
1101 1111 1111 1111 1111 1111 1111 1111
1110 0000 0000 0000 0000 0000 0000 0000
1111 1111 1111 1111 1111 1111 1111 1111
#1
#2
#2
#3
#4
#5
#6


A[0..31]
32
A[0..29]
30
A[0..29]
30
32
32
D[0..31]
2
29
ADDRs
DATA
RAM
1Gx32bits
#1
CS
Decodif. 2x4
Q0
Q1
Q2
Q3
00
01
10
11
A30
A31
SEL0
SEL1
enable
A[0..28]
29
32
ADDRs
DATA
RAM
512Mx32bits
#2
CS
D[0..31]
Dec. 1x2
A29
SEL0
0
1
enable
16
D[0..15]
32
16
D[16..31]
ADDRs
DATA
RAM
512Mx16bits
#3
CS
ADDRs
DATA
RAM
512Mx16bits
#4
CS
16
D[0..15]
32
16
D[16..31]
ADDRs
DATA
RAM
512Mx16bits
#5
CS
ADDRs
DATA
RAM
512Mx16bits
#6
CS