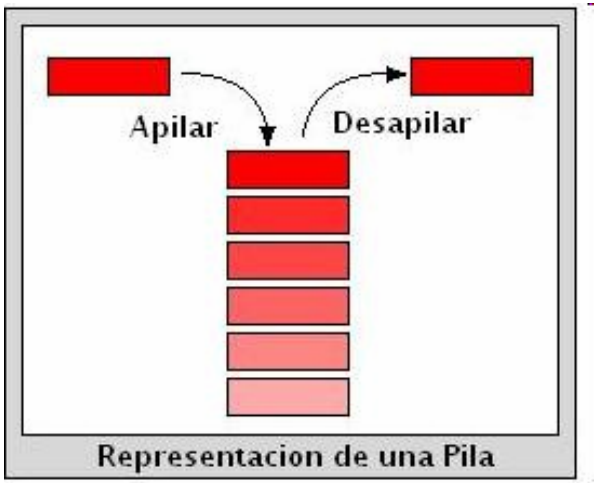
Funcionamiento de la Pila (o stack)

- Todo μP cuenta con una memoria de almacenamiento temporal denominada *Pila.*
- Es una estructura de datos de tipo secuencial (LIFO).
- Existen dos operaciones básicas posibles: *apilar* y *desapilar.*
- Solo se tiene acceso a la parte superior de la pila (último objeto apilado).
- Es necesaria para el funcionamiento de las instrucciones de llamado y vuelta de subrutinas (CALL y RET), las instrucciones PUSH y POP, entre otras.

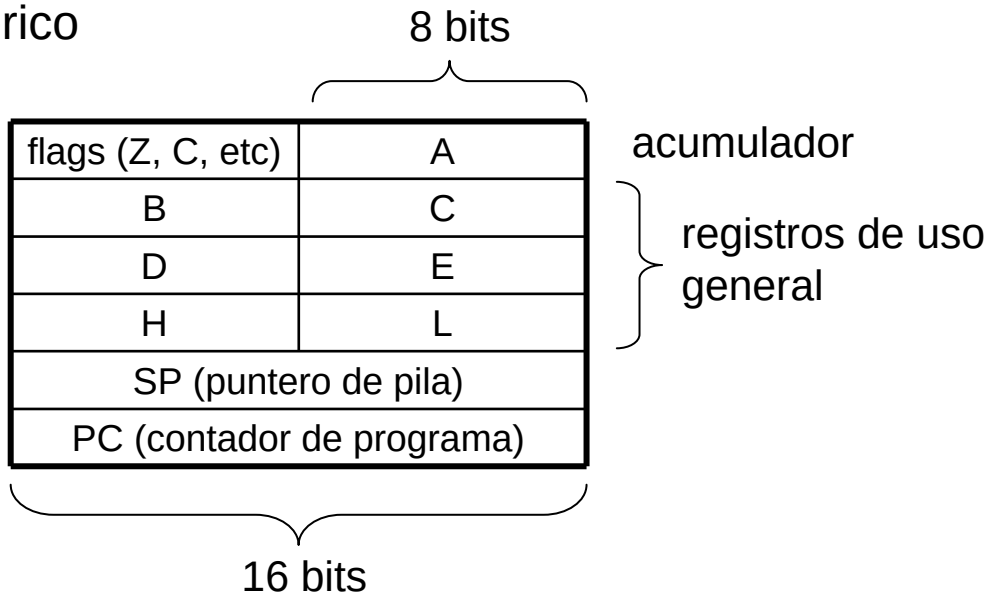
- En algunos μP la pila esta formada por un conjunto de registros internos (Ej. PICs).

- En otros casos (Intel 80XX, Z80) la pila utiliza memoria de datos (RAM) para la pila junto a un registro específico en el μP llamado *stack pointer* (puntero de pila).



Registros internos: stack pointer

Registros internos de un μP genérico (inspirado en el Intel 8085):



Funcionamiento de la instrucción PUSH usando el registro SP:

PUSH BC

Después de ejecutar la instrucción:

(SP-1) ← B
(SP-2) ← C
SP ← SP-2

Funcionamiento de la instrucción POP:

POP DE

Después de ejecutar la instrucción:

E ← (SP)
D ← (SP+1)
SP ← SP+2

Instrucciones PUSH y POP: los “registros pares”

Un ejemplito del uso de las instrucciones PUSH y POP:

```
2100h:  MVI   B,7
        MVI   A,0

        PUSH  BC
        MVI   B,3
        ADD   B
        POP   BC

        DCR   B
        JNZ   2100h
```

¿Qué valor queda en A?

¿Qué valor debería tener SP para que este ejemplo funcione?

En el caso del μP 8085, la instrucción PUSH “apila” siempre un par de registros.

Estos pares solo pueden ser:

B-C
D-E
H-L
flags-A

Por otro lado, la instrucción POP “desapila” registros también de a pares.

- A la unión de registros simples (de 8-bits) en pares se la denomina “registro-par”.
- Un “registro par” se puede pensar como un registro único de 16-bits.

Instrucciones CALL y RET: subrutinas

Las *subrutinas*

• Pueden pensarse como subprogramas dentro de un programa principal (PP).

• Se encargan, en general, de resolver tareas específicas.

• Según el lenguaje, se las conoce también como: procedimientos o funciones.

• Los μP cuentan en general con instrucciones para invocar subrutinas (CALL) e instrucciones para retornar de las subrutinas al PP (RET).

Funcionamiento de la instrucción CALL usando el registro SP:

CALL dir_sub

Después de ejecutar la instrucción (dir=PC+3):

(SP-1) ← dir_H
(SP-2) ← dir_L
SP ← SP-2
PC ← dir_sub

Funcionamiento de la instrucción RET:

RET

Después de ejecutar la instrucción:

PC ← dir, donde
dir_L ← (SP)
dir_H ← (SP+1)
SP ← SP+2

Llamado a subrutinas: uso de SP

Ejemplo:

Supongamos inicialmente SP=2000h

Dirección:	Instrucción:
.	
.	
.	
0102h:	CALL 3010h
0105h:	ADD B
.	
.	
.	
3010h:	
.	* subrutina *
.	
.	
	RET

Antes del CALL:

SP=2000h
PC=0102h

Memoria:	SP →	2000h:	xx
		1FFFh:	xx
		1FFEh:	xx
		1FFDh:	xx

Después del CALL:

SP=1FFEh
PC=3010h

Memoria:		2000h:	xx
		1FFFh:	01
	SP →	1FFEh:	05
		1FFDh:	xx

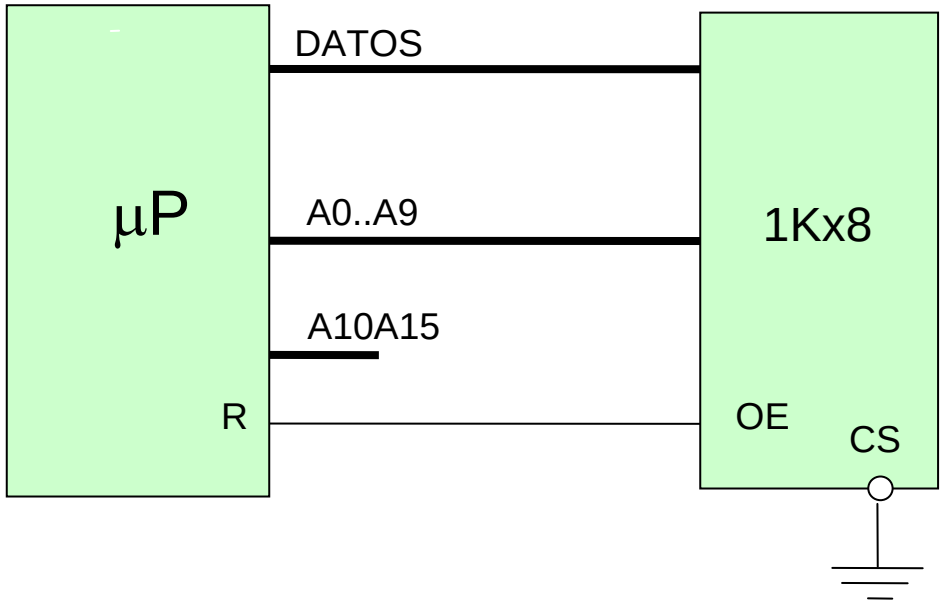
Después del RET:

SP=2000h
PC=0105h

Memoria:	SP →	2000h:	xx
		1FFFh:	01
		1FFEh:	05
		1FFDh:	xx

Decodificación de direcciones

• Los dispositivos conectados a un μP (memorias, E/S, etc.) tienen en general menos líneas de direcciones que el micro.
• Una memoria de 1K x 8 tiene 10 líneas de direcciones y ocho de datos. ¿Cómo hacemos para conectarla, suponiendo que está sola con un μP capaz de direccionar 64K?
• Una solución es la siguiente:



¿Qué dirección debe tener el μP para comunicarse con esta memoria?

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
x	x	x	x	x	x	0	0	0	0	0	0	0	0	0	0
x	x	x	x	x	x	0	0	0	0	0	0	0	0	0	1
…															…
…															…
x	x	x	x	x	x	1	1	1	1	1	1	1	1	1	0
x	x	x	x	x	x	1	1	1	1	1	1	1	1	1	1

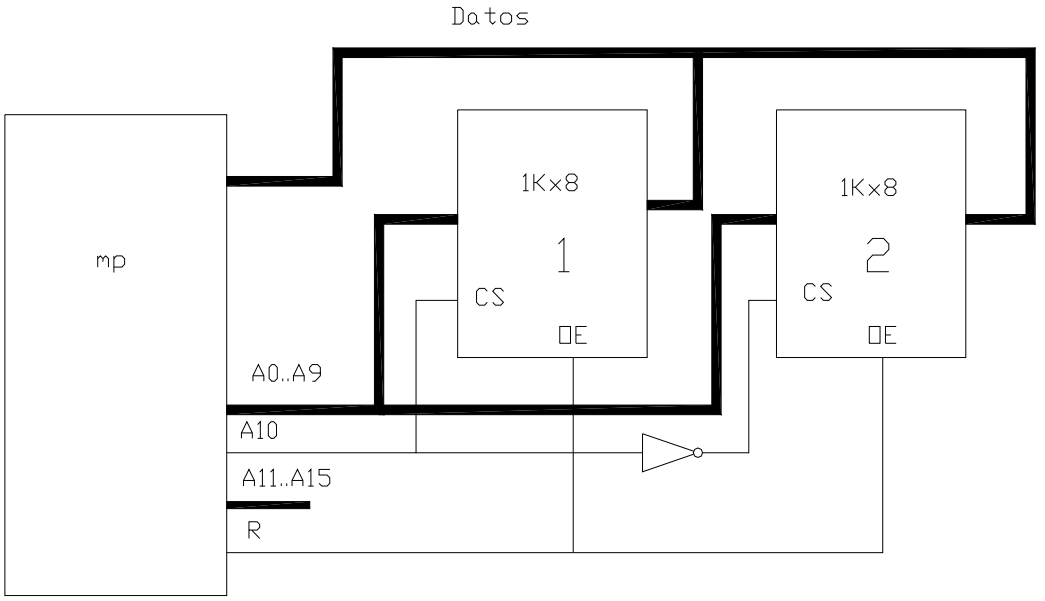
Decodificación de direcciones

Si el μP pone la dirección 0000h lee la primera dirección de la memoria, y si pone la dirección 03FFh lee la última.
¿Que pasa si el μP pone la dirección F000h?
¿En cuantas direcciones distintas lee el mismo dato?

Mapa de memoria:

	0000
	03FF
	0400
	07FF
	0800
	0BFF
	...
	...
	...
	...
	FC00
	FFFF

Como podríamos conectar el μP con dos memorias como las usadas en el ejemplo anterior?



Decodificación: dos memorias

Qué dirección debe tener el μP para comunicarse con estas memorias?

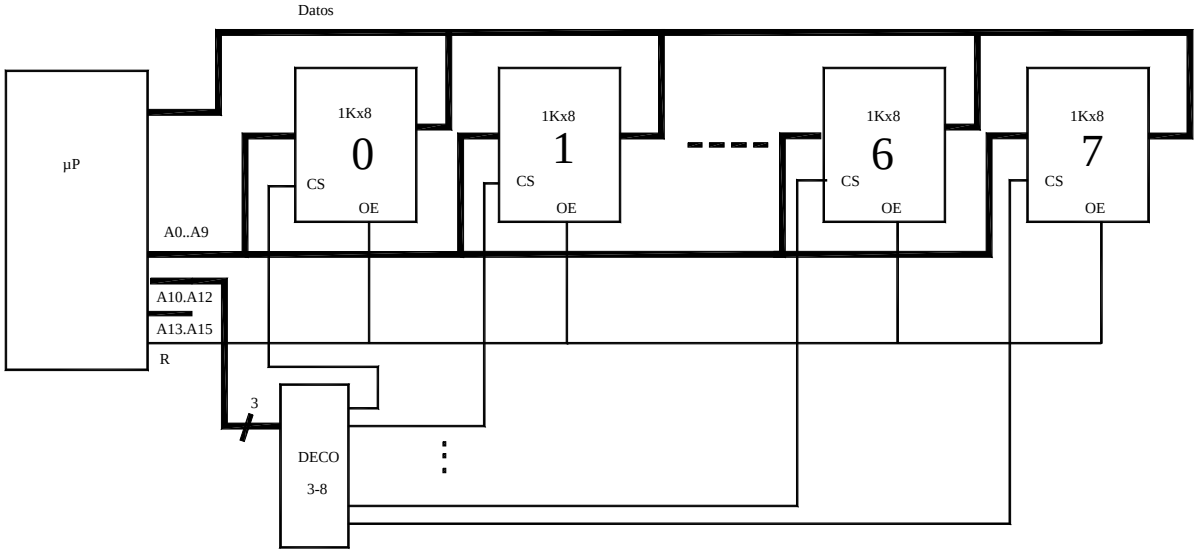
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	chip
x	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	2
x	x	x	x	x	0	0	0	0	0	0	0	0	0	0	1	
x	x	x	x	x	1	0	0	0	0	0	0	0	0	0	0	1
x	x	x	x	x	1	0	0	0	0	0	0	0	0	0	1	
. . .																. . .
x	x	x	x	x	0	1	1	1	1	1	1	1	1	1	0	2
x	x	x	x	x	0	1	1	1	1	1	1	1	1	1	1	
x	x	x	x	x	1	1	1	1	1	1	1	1	1	1	0	1
x	x	x	x	x	1	1	1	1	1	1	1	1	1	1	1	

Mapa de memoria:

Chip 2	0000
	03FF
Chip 1	0400
	07FF
	.
	.
	.
Chip 2	F800
	FBFF
Chip 1	FC00
	FFFF

Decodificación: 8 memorias

Como podríamos conectar 8 memorias como estas?



A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Chip
x	x	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x	x	x	0	0	0	1	1	1	1	1	1	1	1	1	1	
x	x	x	0	0	1	0	0	0	0	0	0	0	0	0	0	1
x	x	x	0	0	1	1	1	1	1	1	1	1	1	1	1	
x	x	x	0	1	0	0	0	0	0	0	0	0	0	0	0	2
x	x	x	0	1	0	1	1	1	1	1	1	1	1	1	1	
x	x	x	0	1	1	0	0	0	0	0	0	0	0	0	0	3
x	x	x	0	1	1	1	1	1	1	1	1	1	1	1	1	
x	x	x	1	0	0	0	0	0	0	0	0	0	0	0	0	4
x	x	x	1	0	0	1	1	1	1	1	1	1	1	1	1	
x	x	x	1	0	1	0	0	0	0	0	0	0	0	0	0	5
x	x	x	1	0	1	1	1	1	1	1	1	1	1	1	1	
x	x	x	1	1	0	0	0	0	0	0	0	0	0	0	0	6
x	x	x	1	1	0	1	1	1	1	1	1	1	1	1	1	
x	x	x	1	1	1	0	0	0	0	0	0	0	0	0	0	7
x	x	x	1	1	1	1	1	1	1	1	1	1	1	1	1	

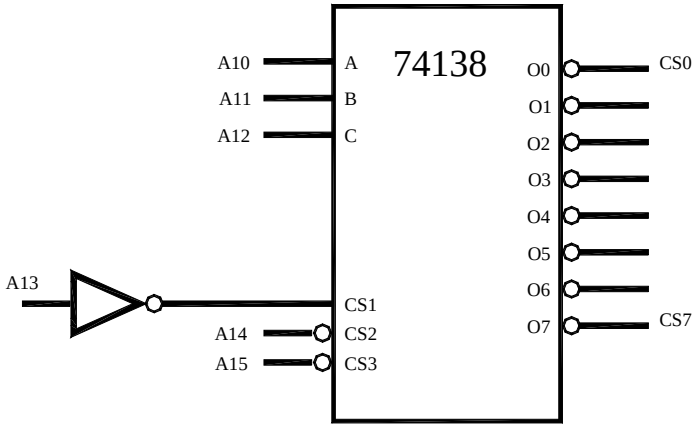
Decodificación: 8 memorias

Mapa de memoria:

Chip 0	0000
	03FF
Chip 1	0400
	07FF
Chip 2	0800
	0BFF
Chip 3	0C00
	0FFF
Chip 4	1000
	13FF
Chip 5	1400
	17FF
Chip 6	1800
	1BFF
Chip 7	1C00
	1FFF

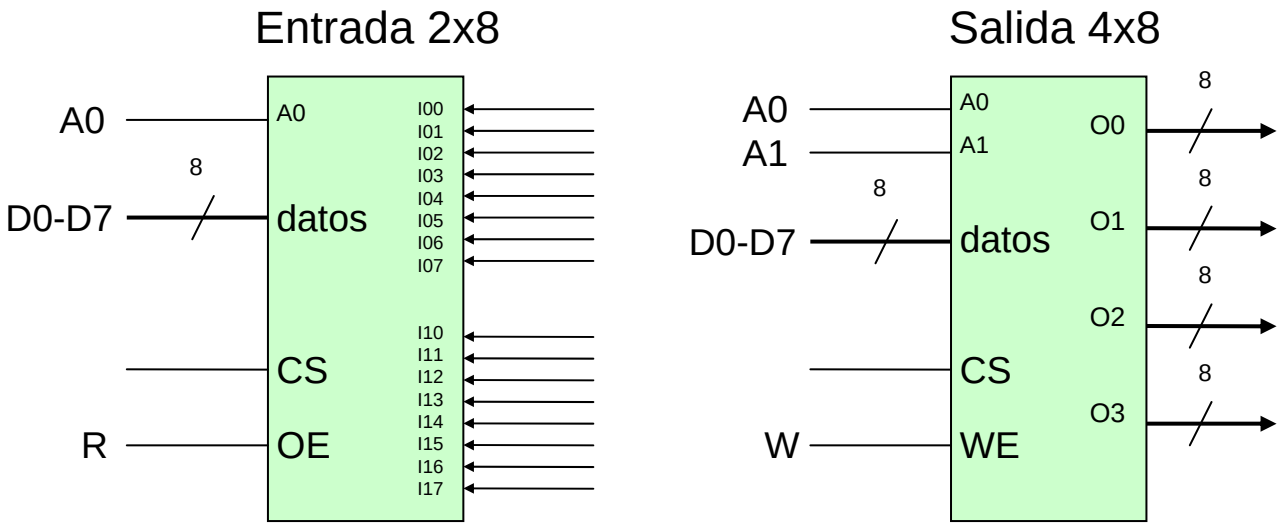
¿Que chip se activa cuando en el bus de direcciones está 20FFh? ¿y cuando está el valor FF00h?

¿Cómo se puede hacer para que exista una sola dirección?



Decodificación: bloques de entrada y salida

De la misma manera que tenemos direcciones específicas para cada memoria, también se hace lo mismo para los dispositivos de entrada salida:



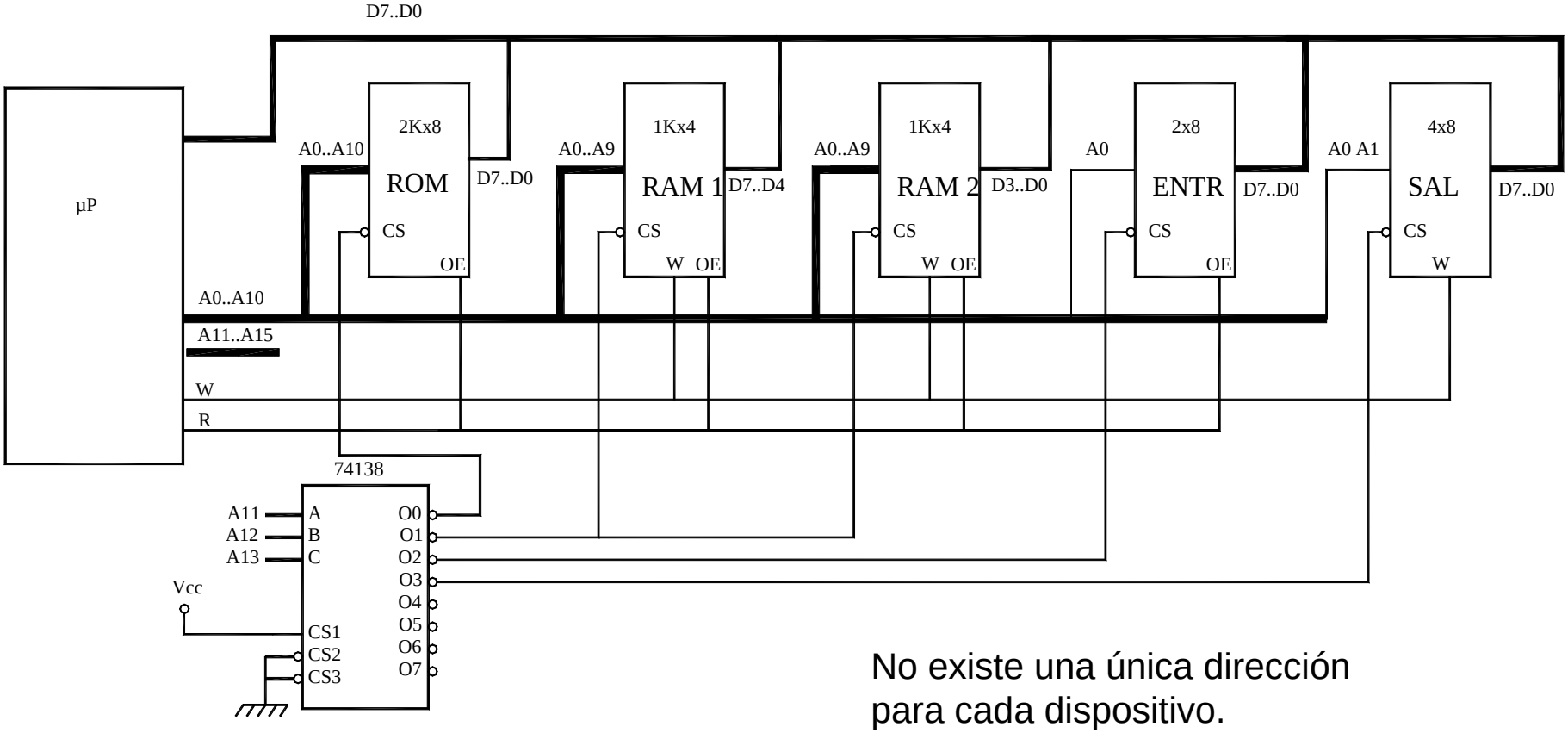
• Hay que lograr un diseño que permita acceder a todas las memorias y a todos los periféricos teniendo en cuenta que no se produzca un conflicto de direcciones.

• Un sistema puede trabajar bien si uno o mas dispositivos están mapeados en memoria mas de una vez (se simplifica el hardware).

Decodificación: ejemplos

Supongamos que queremos diseñar un sistema que consta de un µP de 8 bits de datos y 16 bits de direcciones y se lo quiere conectar con:

- 1 Memoria ROM de 2Kx8
- 2 Memorias RAM de 1Kx4
- 1 Puerto de entrada de 2x8
- 1 Puerto de Salida de 4x8



No existe una única dirección para cada dispositivo.

Decodificación: ejemplos

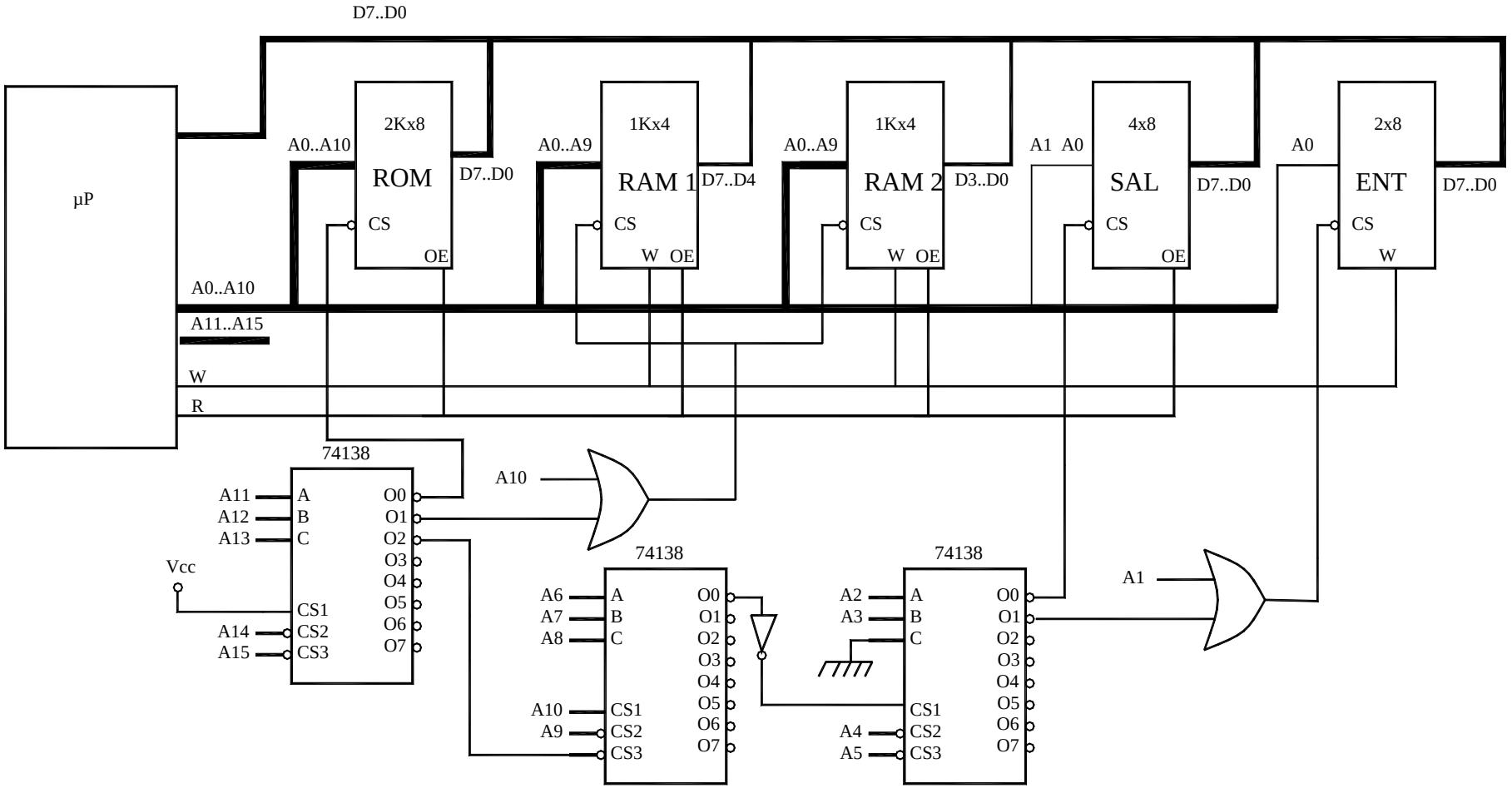
Mapa de memoria:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	CHIP
x	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ROM
x	x	0	0	0	1	1	1	1	1	1	1	1	1	1	1	
x	x	0	0	1	x	0	0	0	0	0	0	0	0	0	0	RAM 1
x	x	0	0	1	x	1	1	1	1	1	1	1	1	1	1	
x	x	0	0	1	x	0	0	0	0	0	0	0	0	0	0	RAM 2
x	x	0	0	1	x	1	1	1	1	1	1	1	1	1	1	
x	x	0	1	0	x	x	x	x	x	x	x	x	x	x	0	ENT
x	x	0	1	0	x	x	x	x	x	x	x	x	x	x	1	
x	x	0	1	1	x	x	x	x	x	x	x	x	x	0	0	SAL
x	x	0	1	1	x	x	x	x	x	x	x	x	x	1	1	

** No existe una única dirección para cada dispositivo.

Decodificación: ejemplos

Segundo caso: cada dispositivo tiene una única dirección:



Decodificación: ejemplos

Mapa de memoria (segundo caso):

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Chip
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ROM
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	RAM 1
0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	RAM 2
0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	
0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	ENT
0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	SAL
0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	

** Cada dispositivo tiene una única dirección a costa de un hardware muy complicado.

Decodificación: ejemplos

Primer caso:

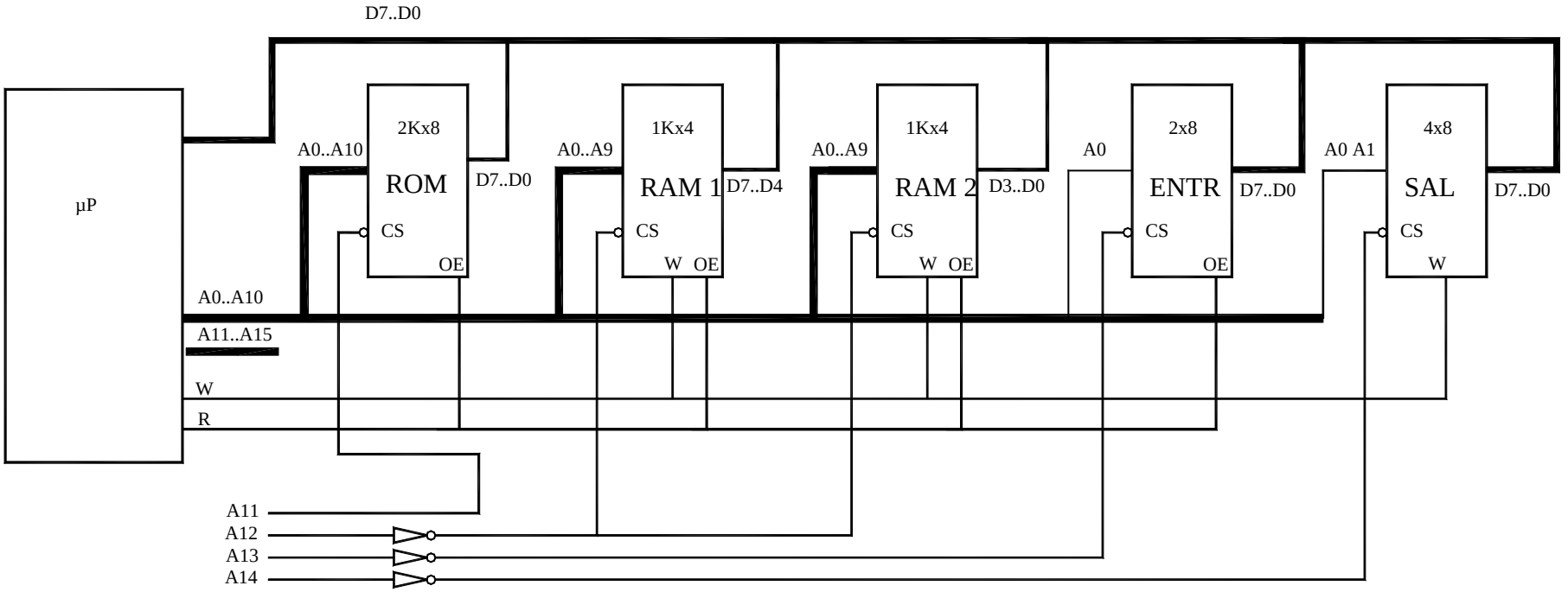
	ROM	0000
		07FF
x 2^1	RAM 1 y 2	0800
		0FFF
x 2^{10}	ENTRADA	1000/1
		17FE/F
x 2^9	SALIDA	1800/3
		1FFC/F
	NADA	2000
		3FFF
	IDEM	4000
		7FFF
	IDEM	8000
		BFFF
	IDEM	C000
		FFFF

Segundo caso:

	ROM	0000
		07FF
	RAM 1 y 2	0800
		0BFF
	NADA	0C00
		1403
	ENTRADA	1404
		1405
	NADA	1406
		1BFF
	SALIDA	1C00
		1C03
	NADA	1C04
		FFFF

Decodificación: ejemplos

Tercer caso: hardware mucho mas simple:



Decodificación: ejemplos

Mapa de memoria:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}					Chip
x	0	0	0	0					ROM
x	0	0	1	1					RAM 1
x	0	0	1	1					RAM 2
x	0	1	0	1					ENT
x	1	0	0	1					SAL

¿Que chip se habilita con la dirección F000h?

¿y en la dirección F800h?

- En este caso el programador debe cuidar de no usar las direcciones que puedan ocasionar un conflicto.
- La ventaja: se requiere menos hardware.
- El diseño queda limitado para el uso de muchas direcciones en caso de una ampliación posterior.