

Casos prácticos de Sistemas Electrónicos Digitales

J. J. García Domínguez
A. Gardel Vicente
J. L. Lázaro Galilea
C. Luna Vázquez
C. Mataix Gómez
J. M. Miguel Jiménez
(Eds.)

Casos prácticos de Sistemas Electrónicos Digitales

TEXTOS UNIVERSITARIOS
TECNOLOGÍA

UAH

Casos prácticos de Sistemas Electrónicos Digitales

J.J. García Domínguez
A. Gardel Vicente
J. L. Lázaro Galilea
C. Luna Vázquez
C. Mataix Gómez
J. M. Miguel Jiménez (Eds.)



Universidad
de Alcalá
SERVICIO DE PUBLICACIONES

El contenido de este libro no podrá ser reproducido,
ni total ni parcialmente, sin el previo permiso escrito del editor.
Todos los derechos reservados.

© Universidad de Alcalá, 2019
Servicio de Publicaciones
Plaza de San Diego, s/n
28801 Alcalá de Henares
www.uah.es

I.S.B.N.: 978-84-18254-69-7

Composición: Solana e Hijos, A. G., S.A.U.
Impresión y encuadernación: Solana e Hijos, A.G., S.A.U.
Impreso en España

ÍNDICE

CAPÍTULO 1. Sistemas electrónicos digitales programables	13
Introducción	13
Problema 1.1.	14
Problema 1.2.	15
Problema 1.3.	16
Problema 1.4.	17
Problema 1.5.	17
Problema 1.6.	18
Problema 1.7.	19
Problema 1.8.	21
Problema 1.9.	23
Problema 1.10.	23
Problema 1.11.	27
Problema 1.12.	29
CAPÍTULO 2. Arquitectura y modelo de programación del Cortex M3	33
Introducción	33
Problema 2.1.	34
Problema 2.2.	34
Problema 2.3.	35
Problema 2.4.	36
Problema 2.5.	37
Problema 2.6.	39
Problema 2.7.	42
Problema 2.8.	43
Problema 2.9.	46
Problema 2.10.	48

Problema 2.11.	50
Problema 2.12.	52
Problema 2.13.	53
Problema 2.14.	54
Problema 2.15.	57
Problema 2.16.	59
Problema 2.17.	60
Problema 2.18.	63
CAPÍTULO 3. El sistema de excepciones del Cortex M3	65
Introducción	65
Problema 3.1.	66
Problema 3.2.	66
Problema 3.3.	66
Problema 3.4.	68
Problema 3.5.	71
Problema 3.6.	74
Problema 3.7.	76
Problema 3.8.	80
Problema 3.9.	83
Problema 3.10.	85
Problema 3.11.	87
Problema 3.12.	89
Problema 3.13.	90
Problema 3.14.	93
Problema 3.15.	95
Problema 3.16.	96
Problema 3.17.	98
Problema 3.18.	102
Problema 3.19.	104
Problema 3.20.	107
Problema 3.21.	109
Problema 3.22.	112
Problema 3.23.	113
CAPÍTULO 4. Organización y gestión de la memoria	117
Introducción	117
SECCIÓN 1. Ampliación de memorias	118
Problema 4.1.	118

Problema 4.2.	118
Problema 4.3.	118
Problema 4.4.	119
SECCIÓN 2. Diseño de mapas de memoria	120
Problema 4.5.	120
Problema 4.6.	121
Problema 4.7.	121
Problema 4.8.	122
Problema 4.9.	124
Problema 4.10.	125
Problema 4.11.	127
SECCIÓN 3. Diseño de sistemas de memoria con el EMC	128
Problema 4.12.	128
Problema 4.13.	128
Problema 4.14.	129
Problema 4.15.	131
Problema 4.16.	133
Problema 4.17.	134
Problema 4.18.	136
Problema 4.19.	138
Problema 4.20.	140
SECCIÓN 4. Análisis de cronogramas de acceso.	141
Problema 4.21.	141
Problema 4.22.	143
Problema 4.23.	146
Problema 4.24.	148
Problema 4.25.	151
Problema 4.26.	154
Problema 4.27.	157
Problema 4.28.	159
SECCIÓN 5. Problemas globales.	161
Problema 4.29.	161
Problema 4.30.	165
Problema 4.31.	171
APÉNDICE A. SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 1 ..	177
Solución al Problema 1.1.....	177
Solución al Problema 1.2.....	178
Solución al Problema 1.3.....	179

Solución al Problema 1.4	179
Solución al Problema 1.5	180
Solución al Problema 1.6	180
Solución al Problema 1.7	181
Solución al Problema 1.8	182
Solución al Problema 1.9	183
Solución al Problema 1.10	184
Solución al Problema 1.11	185
Solución al Problema 1.12	188
APÉNDICE B. SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 2	191
Solución al Problema 2.1	191
Solución al Problema 2.2	192
Solución al Problema 2.3	193
Solución al Problema 2.4	194
Solución al Problema 2.5	194
Solución al Problema 2.6	197
Solución al Problema 2.7	198
Solución al Problema 2.8	199
Solución al Problema 2.9	200
Solución al Problema 2.10	200
Solución al Problema 2.11	201
Solución al Problema 2.12	202
Solución al Problema 2.13	203
Solución al Problema 2.14	204
Solución al Problema 2.15	205
Solución al Problema 2.16	206
Solución al Problema 2.17	207
Solución al Problema 2.18	208
APÉNDICE C. SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 3	211
Solución al Problema 3.1	211
Solución al Problema 3.2	211
Solución al Problema 3.3	212
Solución al Problema 3.4	212
Solución al Problema 3.5	214
Solución al Problema 3.6	214
Solución al Problema 3.7	215
Solución al Problema 3.8	216
Solución al Problema 3.9	218
Solución al Problema 3.10	219

Solución al Problema 3.11.....	220
Solución al Problema 3.12.....	221
Solución al Problema 3.13.....	222
Solución al Problema 3.14.....	224
Solución al Problema 3.15.....	225
Solución al Problema 3.16.....	226
Solución al Problema 3.17.....	227
Solución al Problema 3.18.....	228
Solución al Problema 3.19.....	230
Solución al Problema 3.20.....	231
Solución al Problema 3.21.....	233
Solución al Problema 3.22.....	235
Solución al Problema 3.23.....	236
APÉNDICE D. SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 4 ..	237
SECCIÓN 1. Ampliación de memorias	237
Solución al Problema 4.1.....	237
Solución al Problema 4.2.....	238
Solución al Problema 4.3.....	239
Solución al Problema 4.4.....	240
SECCIÓN 2. Diseño de mapas de memoria	241
Solución al Problema 4.5.....	241
Solución al Problema 4.6.....	244
Solución al Problema 4.7.....	245
Solución al Problema 4.8.....	247
Solución al Problema 4.9.....	249
Solución al Problema 4.10.....	251
Solución al Problema 4.11.....	253
SECCIÓN 3. Diseño de sistemas de memoria con el EMC	253
Solución al Problema 4.12.....	253
Solución al Problema 4.13.....	256
Solución al Problema 4.14.....	257
Solución al Problema 4.15.....	258
Solución al Problema 4.16.....	261
Solución al Problema 4.17.....	263
Solución al Problema 4.18.....	264
Solución al Problema 4.19.....	266
Solución al Problema 4.20.....	268

SECCIÓN 4. Análisis de cronogramas de acceso	270
Solución al Problema 4.21.....	270
Solución al Problema 4.22.....	270
Solución al Problema 4.23.....	271
Solución al Problema 4.24.....	272
Solución al Problema 4.25.....	273
Solución al Problema 4.26.....	274
Solución al Problema 4.27.....	275
Solución al Problema 4.28.....	277
SECCIÓN 5. Problemas globales	278
Solución al Problema 4.29.....	278
Solución al Problema 4.30.....	281
Solución al Problema 4.31.....	284

CAPÍTULO 1

SISTEMAS ELECTRÓNICOS DIGITALES PROGRAMABLES

INTRODUCCIÓN

En el capítulo 1 de la asignatura se introducen los conceptos básicos necesarios para poder afrontarla con éxito, comprendiendo todos los términos y a qué se refieren, cuándo son comentados y/o abordados. Estos conceptos tienen que ver con el microprocesador y microcontrolador y sus arquitecturas, modos de direccionamiento e instrucciones, elementos que constituyen un sistema digital programables, diseño de sistemas digitales programables, sistemas empotrados, etc.

En los problemas que se presentan en este capítulo se recogen cuestiones teóricas acerca de los conceptos comentados anteriormente, así como ejercicios de conexión de elementos constitutivos de sistemas electrónicos digitales a los buses del procesador.

PROBLEMA 1.1.

Marque en la Tabla 1.1 con una ‘X’, las afirmaciones verdaderas.

TABLA 1.1. CUESTIONES.

	Preguntas	V
1	La depuración sobre el dispositivo:	
	<ul style="list-style-type: none"> • Permite visualizar contenidos de memoria y registros • No permite depurar paso a paso ni poner puntos de ruptura • Aumenta la velocidad de ejecución del programa • Utilizan programas residentes en el sistema y un puerto de comunicaciones 	
2	Los microprocesador RISC	
	<ul style="list-style-type: none"> • Se desarrollaron primero que los CISC • Las instrucciones son sencillas • Tienen un set de instrucciones más grande que los CISC • Facilitan la superescalaridad 	
3	La segmentación:	
	<ul style="list-style-type: none"> • Posibilita que no sea necesario terminar una instrucción para comenzar otra • Obliga a repetir las unidades de ejecución o poner varias de diferentes tipos • Y la escalaridad no se puede utilizar de forma conjunta en un mismo microprocesador • No se usa en los microprocesadores modernos 	
4	Los sistemas empotrados:	
	<ul style="list-style-type: none"> • No se pueden utilizar en sistemas que requieran operaciones en tiempo real • Realizan varias tareas de forma concurrente • Se diseñan para que estén continuamente funcionando • El coste no es importante, ya que solo se utilizan en aplicaciones muy específicas como es el caso de naves espaciales 	

PROBLEMA 1.2.

Responda a las preguntas de test de la Tabla 1.2, marcando con una ‘X’, las afirmaciones verdaderas. Tenga en cuenta que son preguntas de respuesta múltiple.

TABLA 1.2. PREGUNTAS DE TEST.

Preguntas	
1.	Un sistema empotrado para un juguete básico debe cumplir las características de:
	<ul style="list-style-type: none"> • Bajo coste • Concurrencia • Bajo consumo • Alta fiabilidad
2	Los microprocesadores:
	<ul style="list-style-type: none"> • Incluyen periféricos en el encapsulado • Requieren de memoria externa • No permiten operaciones aritméticas • Implementan la misma función aunque se cambie el programa
3	Las instrucciones que ejecuta un microprocesador:
	<ul style="list-style-type: none"> • Son siempre básicas e ilimitadas • Son siempre complejas y limitadas • Se pueden ampliar con combinaciones de operaciones básicas • Son únicamente las operaciones lógicas AND y OR
4	Un microcontrolador generalmente contiene en su encapsulado:
	<ul style="list-style-type: none"> • Una CPU • Memoria de datos • Sensores de temperatura • Periféricos
5	Los sistemas digitales:
	<ul style="list-style-type: none"> • Están obsoletos • Leen información del exterior por medio de interfaces de entrada • Leen información del exterior por medio de interfaces de salida • Procesan datos de entrada para generar señales de salida

PROBLEMA 1.3.

Responda a las preguntas de test de la Tabla 1.3, marcando con una ‘X’, las afirmaciones verdaderas. Tenga en cuenta que son preguntas de respuesta múltiple.

TABLA 1.3. PREGUNTAS DE TEST.

Preguntas	
1.	La capacidad de direccionamiento de un microprocesador depende de:
	<ul style="list-style-type: none"> • De las dimensiones del chip en milímetros • El número de líneas del bus de datos • El número de líneas del bus de direcciones • El número de líneas del bus de control
2	Algunas ventajas de la utilización de subrutinas en la programación son:
	<ul style="list-style-type: none"> • Facilitar la depuración • No presenta ninguna ventaja • Estructurar el programa • Evitar que el programa sea demasiado largo
3	La pila:
	<ul style="list-style-type: none"> • Se controla con el registro contador de programa • Se emplea obligatoriamente en los bucles • Se almacena en memoria no volátil • Todas las respuestas son incorrectas
4	En un procesador segmentado de N etapas el rendimiento:
	<ul style="list-style-type: none"> • Se multiplica por N • Se multiplica por N^2 • Se multiplica por $2N$ • Es inferior al microprocesador no segmentado

PROBLEMA 1.4.

Responda a las preguntas de test de la Tabla 1.4, marcando con una ‘X’, las afirmaciones verdaderas. Tenga en cuenta que son preguntas de respuesta múltiple.

TABLA 1.4. PREGUNTAS DE TEST.

Preguntas	
1.	En un microprocesador de 24 líneas de direcciones:
	<ul style="list-style-type: none"> • El bus de control tiene 24 líneas • El número de posiciones a las que se puede acceder es 16M • Los datos son de 24 bits • Se pueden direccionar los datos de 24 formas distintas
2	El registro contador de programa
	<ul style="list-style-type: none"> • Carga el código de operación procedente de la memoria • Almacena el resultado de la ALU • Indica dónde se encuentra la siguiente instrucción a ejecutar • Almacena las instrucciones del programa temporalmente
3	La comunicación entre una unidad periférica y el microprocesador, cuando es iniciada a petición del primero se denomina:
	<ul style="list-style-type: none"> • Una unidad periférica nunca puede iniciar la comunicación con un microprocesador • Por sondeo • Por programa • Por interrupción
4	Las subrutinas:
	<ul style="list-style-type: none"> • Permiten estructurar el programa • Evitan tener que codificar repetidas veces una tarea • Equivalen a las funciones en un lenguaje de alto nivel • Están limitadas en número

PROBLEMA 1.5.

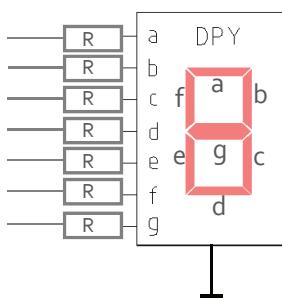
Responda a las preguntas de test de la Tabla 1.5, marcando con una ‘X’, las afirmaciones verdaderas. Tenga en cuenta que son preguntas de respuesta múltiple.

TABLA 1.5. PREGUNTAS DE TEST.

Preguntas	
1.	La pila:
	<ul style="list-style-type: none"> • Almacena las instrucciones del programa temporalmente • Se emplea obligatoriamente en instrucciones de escritura múltiple • Se almacena en memoria no volátil • Todas las respuestas son incorrectas
4	Las instrucciones se almacenan en la memoria de programa en formato:
	<ul style="list-style-type: none"> • Lenguaje C Estándar • Binario • Ensamblador • Lenguaje de bajo nivel

PROBLEMA 1.6.

A través del puerto de E/S de 16 bits (P0) de una interfaz se quiere conectar a un sistema un *display* de LEDs de 7 segmentos (Figura 1.1) para visualizar un número del 1 al 9, y 9 *switches* (sw1-sw9) para introducir al sistema el número que se desea visualizar (si se activa sw1 se visualizará el número 1, si se activa sw2 se visualizará el número 2, etc. Solo un *switch* puede estar activo).



El *display* se conectará por las líneas 0 (segmento a) a 6 (segmento g) del puerto P0 y los *switch* a las líneas 7 a 15 del mismo puerto.

FIGURA 1.1. CORRESPONDENCIA DE LOS SEGMENTOS DEL DISPLAY DE LED.

Conteste justificadamente a las siguientes preguntas:

- Indique con “X” en las casillas de la Tabla 1.6 la dirección (entrada/salida) con que se debería configurar cada uno de los pines del puerto. Indique también si sería conveniente conectar a los pines del puerto resistencias de *pull-up* o *pull-down*.

TABLA 1.6. CONFIGURACIÓN DE PINES DEL PUERTO P0.

PIN	P0.15	P0.14	P0.13	P0.12	P0.11	P0.10	P0.9	P0.8	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
Entrada																
Salida																
Pull-up/down																

- Indique en las casillas de la Tabla 1.7 el valor del registro de salida del Puerto P0 para que, inicialmente, el *display* se visualice el número 4.

TABLA 1.7. CONFIGURACIÓN DE PINES DEL PUERTO P0.

PIN	P0.15	P0.14	P0.13	P0.12	P0.11	P0.10	P0.9	P0.8	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
Registro de salida																

- Indique mediante un diagrama de flujo o lenguaje natural los pasos del proceso a seguir para leer los *switches* y representar el valor leído en el *display*.

PROBLEMA 1.7.

Se dispone de un microcontrolador al que se le han conectado un *display* y un pulsador tal y como muestra la Figura 1.2. Se desea que cada vez que el pulsador sea presionado el *display* cambie de la siguiente manera: si mostraba un ‘0’ que cambie a ‘1’, y si mostraba un ‘1’ que cambie a ‘0’.

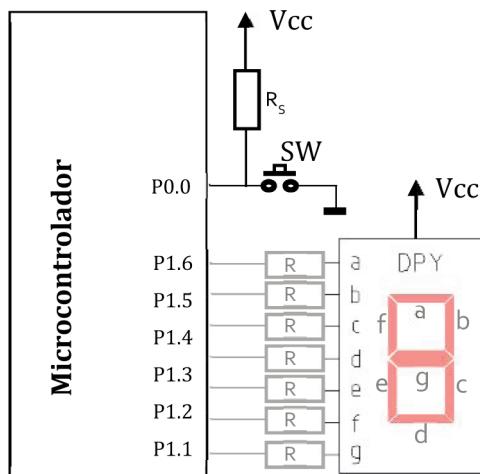


FIGURA 1.2. CIRCUITO DEL PROBLEMA 1.7.

Conteste justificadamente a las siguientes preguntas:

- Indique con “X” en las casillas de la Tabla 1.8 la dirección (entrada/salida) con que se debería configurar cada uno de los pines de los puertos y si sería necesario incluir resistencias de *pull-up*. Sabiendo que el microcontrolador dispone internamente de resistencias de *pull-up* o *pull-down* que pueden ser configuradas, indique si es correcto haber conectado la resistencia R_s .

TABLA 1.8. CONFIGURACIÓN DE PINES DEL PUERTO P.

PIN	P0.15	P0.14	P0.13	P0.12	P0.11	P0.10	P0.9	P0.8	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
Entrada																
Salida																
Pull-up																

- Realice un diagrama de flujo en el que se recojan los pasos a realizar en el programa principal (main) para obtener la funcionalidad que el enunciado general especifica.

NOTA: se deben especificar en todo momento los nombres de los pines concretos sobre los que se debe actuar (consultar, escribir...). Suponga que una subrutina llamada configura mostrada en la Figura 1.3 ya ha realizado todas las configuraciones especificadas en el punto 1

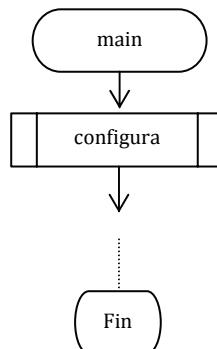


FIGURA 1.3. DIAGRAMA DE FLUJO A COMPLETAR.

PROBLEMA 1.8.

Se dispone de un microcontrolador al que se le han conectado un conjunto de *switches* y LEDs a través del puerto P, como muestra la Figura 1.4. La información introducida por los *switches* está codificada en BCD y se desea representarla en los LEDs en BCD exceso 3 (cada símbolo decimal se le suma 3 y se codifica en binario: ‘0’ → 0011b, ‘1’ → 0100, ..., ‘9’ → 1100b).

Conteste justificadamente a las siguientes preguntas:

1. Indique con “X” en las casillas de la Tabla 1.9 la dirección (entrada/salida) con que se debería configurar cada uno de los pines del puerto. Indique también si sería necesario que internamente dispusiera de resistencias de *pull-up* o *pull-down*, para poder ser configurado con o sin ellas, y en caso afirmativo indique cuál configuraría para cada pin.

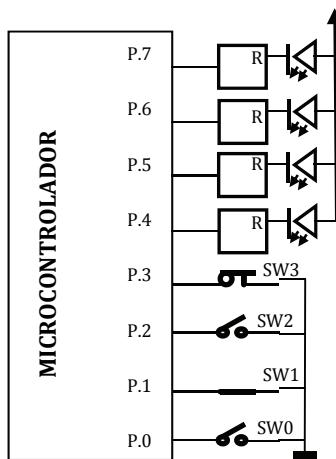


FIGURA 1.4. CIRCUITO DEL PROBLEMA 1.8.

TABLA 1.9. CONFIGURACIÓN DE PINES DEL PUERTO P.

PIN	P0.15	P0.14	P0.13	P0.12	P0.11	P0.10	P0.9	P0.8	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
Entrada																
Salida																
<i>Pull-up/down</i>																

2. Si las instrucciones en ensamblador y etiquetas que podríamos necesitar (no obligatoriamente) son las mostradas en la Tabla 1.10, escriba la secuencia de instrucciones necesarias para leer la información de los *switches* y almacenar en el registro R₁ del microcontrolador únicamente el valor del código de los *switches*.

TABLA 1.10. ETIQUETAS E INSTRUCCIONES DISPONIBLES.

Etiquetas	Instrucciones
Puerto_P (Dirección del puerto)	LDR R _{destino} ,=Etiqueta MOV R _{destino} , R _{fuente} MOV R _{destino} , #Inmediato STR R _{fuente} , [R _{destino}] ADD R _{destino} , #Inmediato AND R _{destino} , #Inmediato LDR R _{destino} , [R _{fuente}] LSL R _{destino} , #Inmediato

3. Escriba las instrucciones necesarias para convertir la información anterior a BCD exceso 3. Considere que el microcontrolador dispone de tantos registros (R_i) como sean necesarios.
4. Escriba las instrucciones necesarias para visualizar la información en los LEDs sin que afecte al resto de pines de P2 que pudieran estar configurados como salidas.

PROBLEMA 1.9.

Se desea conectar un *display* de 7 segmentos a un sistema digital basado en microprocesador. Complete en la en la Figura 1.5 el diagrama de bloques de la conexión, añadiendo los elementos que considere necesarios. Tenga en cuenta que el dato que el microprocesador va a escribir en el *display* ya está codificado en 7 segmentos. Explique brevemente el diseño realizado.

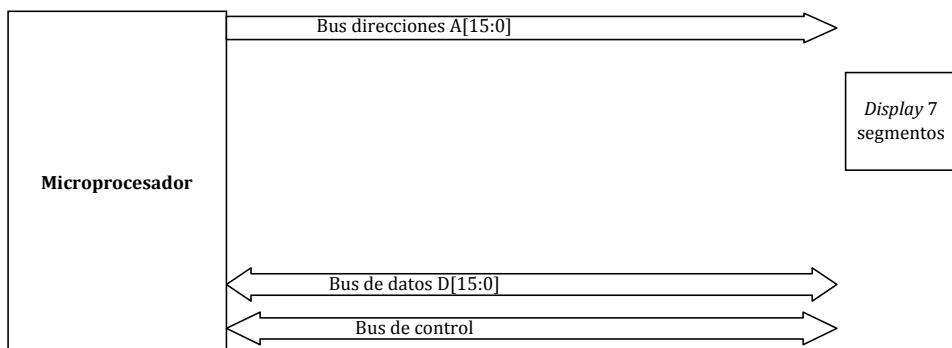


FIGURA 1.5. DIAGRAMA DE BLOQUES DE CONEXIONES.

PROBLEMA 1.10.

Se dispone de un determinado microprocesador con un bus de datos de 8 bits y cuyo formato de instrucción es *Instrucción Operando_fuente [,Operando_destino]*. Para este ejemplo, suponga que sólo podrían hacer falta algunas de las instrucciones que aparecen en la Tabla 1.11. En la Figura 1.6 aparece el microprocesador conectado a un bloque llamado “lógica de selección”, cuyas salidas CS1 y CS2 se activan a nivel bajo cuando el microprocesador realiza una operación de lectura en la dirección 1000H y escritura en la dirección 2000H, respectivamente.

TABLA 1.11. ALGUNAS DE LAS INSTRUCCIONES QUE DISPONE EL MICROPROCESADOR

Instrucción	Operando fuente	Operando destino
LEER	[dirección del mapa de memoria en hexadecimal]	Rx
ESCRIBIR	#Constante	[dirección del mapa de memoria en hexadecimal]
ESCRIBIR	Rx	[dirección del mapa de memoria en hexadecimal]

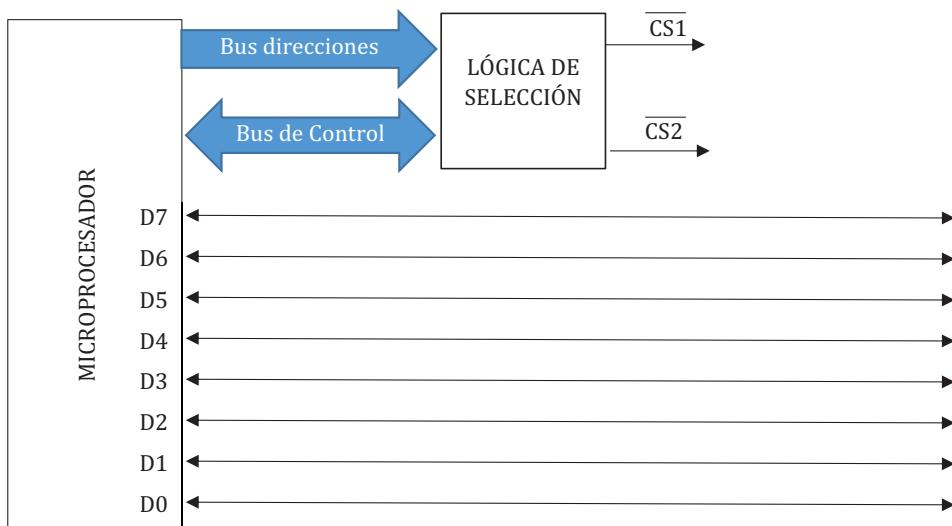


FIGURA 1.6. DIAGRAMA DE BLOQUES DE CONEXIONES.

Se pide:

1. Se desea conectar el microprocesador a un *display* como el de la Figura 1.7 mediante una interfaz que se debe diseñar. Para ello, utilice un circuito integrado de entre los de la figura 1.18, conectándolo adecuadamente al bus de datos del microprocesador. No olvide utilizar resistencias para polarizar el *display*. Nota: se recomienda realizar las conexiones asignando el siguiente orden: segmentos a, b, ..., f → bits d0, d1, ..., d6.

2. Repita el mismo proceso, en este caso para que el microprocesador pueda leer los niveles lógicos de 8 entradas digitales externas procedentes, por ejemplo, de 8 interruptores (no es necesario dibujar estos últimos).
3. Suponga que las direcciones dentro del mapa de memoria asignadas a la interface de los interruptores y al del *display* son 1000H y 2000H, respectivamente. Escriba las instrucciones necesarias para leer el estado de los interruptores y escribirlo directamente sobre el *display*.
4. Considerando que cada interruptor *SWx* en ‘ON’ fija un nivel alto en su entrada digital externa correspondiente, y que el orden de conexiones en el diseño es SW0, SW1, ..., SW6 → d0, d1, ..., d6, indicar cuál deberá ser la configuración de todos los interruptores para que en el *display* se visualice el número 2.

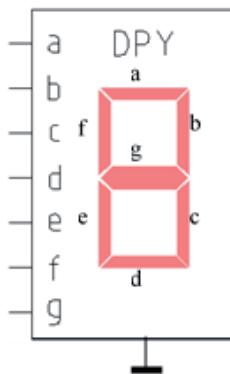


FIGURA 1.7. DISPLAY DE 7 SEGMENTOS.

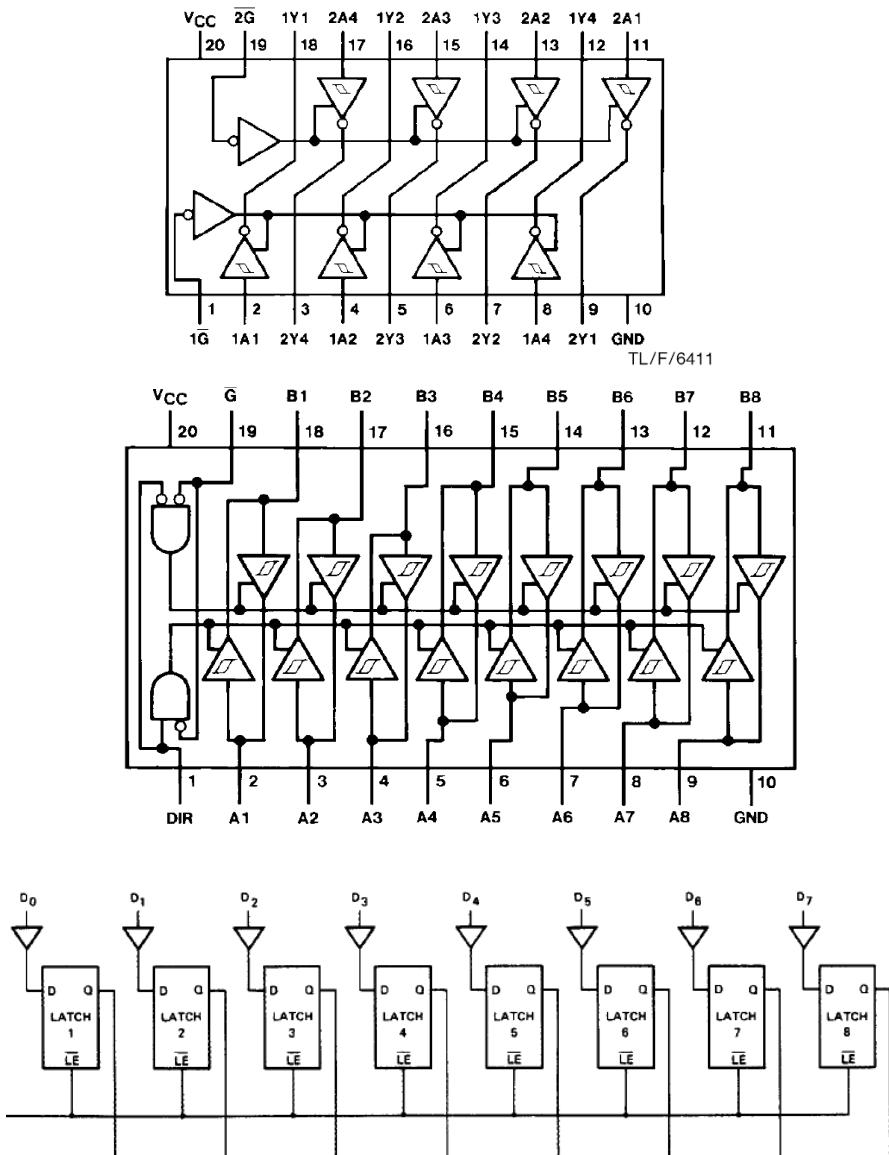


FIGURA 1.8. INTERFACES DISPONIBLES.

PROBLEMA 1.11.

Asuma que un procesador genérico está conectado como se muestra en la Figura 1.9, y que, a partir de un determinado momento en el que el PC contiene el valor 0x400, se ejecuta el código de la Tabla 1.12. Las instrucciones *ADD*, *SUB*, *DIV*, *OR* tienen dos operandos fuente, mientras que las instrucciones *LOAD* y *STORE* sólo uno.

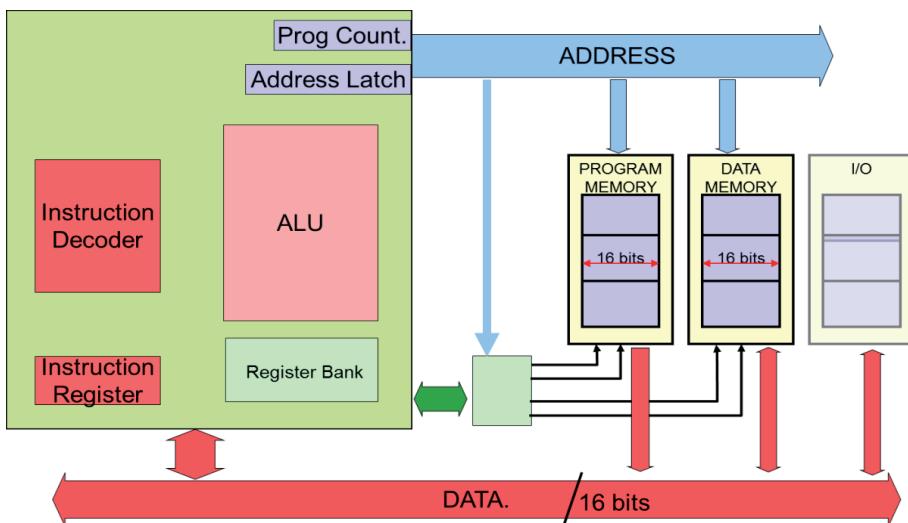


FIGURA 1.9. MICROPROCESADOR GENÉRICO.

TABLA 1.12. CÓDIGO EJECUTADO.

OPCODE	Operando Fuente	Operando Destino	Operando Destino	Tamaño Instrucción
<i>LOAD</i>	-	0x2004	R1	4 Bytes
<i>LOAD</i>	-	0x2008	R2	4 Bytes
<i>LOAD</i>	-	0x200C	R3	4 Bytes
<i>ADD</i>	R1	R2	R4	2 Bytes
<i>SUB</i>	R3	R4	R4	2 Bytes
<i>DIV</i>	R3	#0x03	R5	2 Bytes
<i>OR</i>	R4	R5	R5	2 Bytes
<i>STORE</i>	-	R5	0x2010	4 Bytes

- Indique el tipo de operación que realiza cada instrucción (*control*, *manipulación de bit*, etc.) y los modos de direccionamiento utilizados.

2. Describa cuál es el proceso que se desencadena para ejecutar las instrucciones de carga *LOAD*, cuál es la cantidad total de bytes que se transfieren por el bus DATA, y el número de transferencias que se realizan hasta ejecutar completamente cada una de ellas. Tenga en cuenta que el tamaño de datos de las operaciones es de 2 Bytes.

1^a Fase	El valor del PC se coloca en el bus de direcciones y se selecciona la dirección 400. La memoria de Código se selecciona y pone la primera instrucción (parte de ella) en el bus de datos. La instrucción llega al registro de instrucciones. Se detecta que esa instrucción aún no ha llegado completamente (tiene 4 Bytes)	1 ^a transf. (2bytes)
2^a Fase		
3^a Fase		

3. Haga lo mismo que en la cuestión anterior, ahora para las instrucciones 4 a 7.

4. Repítalo para la instrucción 8 y diga cuál es el número de transacciones total realizado

5. Si el código hubiese sido el proporcionado en la Tabla 1.13, describa **el proceso para realizar la primera instrucción** (desde búsqueda hasta guardar resultado), y calcule cuál sería el número total de transferencias realizado a través del bus DATA considerando que el tamaño de los datos a operar sigue siendo 2 bytes.

TABLA 1.13. CÓDIGO MODIFICADO.

OPCODE	Operando Fuente	Operando Fuente	Operando Destino	Tamaño Instrucción
ADD	0x2004	0x2008	0x2014	8 Bytes
SUB	0x200C	0x2014	0x2014	8 Bytes
DIV	0x200C	#0X03	0x2010	8 Bytes
OR	0x2014	0x2010	0x2010	8 Bytes

6. Si los datos para operar hubieran sido de 4 bytes, ¿cuántas transferencias hubieran sido necesarias para ejecutar todas las instrucciones de la Tabla 1.13?

PROBLEMA 1.12.

Se dispone de un determinado microprocesador con un bus de datos de 8 bits (mostrado en la Figura 1.10) cuyo bus de direcciones se decodifica en un circuito ya diseñado, de manera que cuando se accede a la dirección 0xCD00 en operación de lectura se activa a nivel alto la señal “Selección interruptores” y cuando se accede a la misma dirección en escritura se activa también a nivel alto la señal “Selección LEDs”.

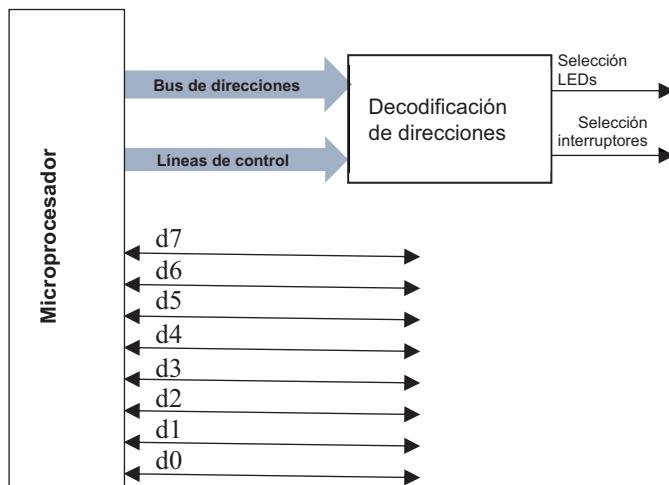


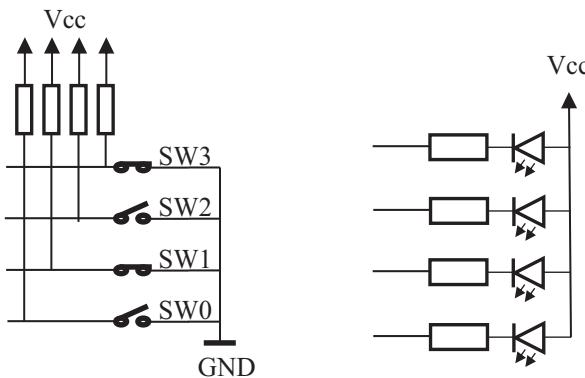
FIGURA 1.10. MICROPROCESADOR.

El formato de instrucciones que maneja es: *Instrucción Operando_fuente , Operando_destino*. Algunas de las instrucciones que soporta se muestran en la Tabla 1.14.

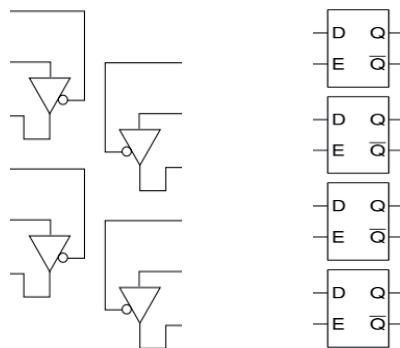
TABLA 1.14. INSTRUCCIONES QUE SOPORTA EL MICROPROCESADOR.

Instrucción	Operando fuente	Operando destino
LEER	[dirección del mapa de memoria en hexadecimal]	Rx
ESCRIBIR	#Constante	[dirección del mapa de memoria en hexadecimal]
ESCRIBIR	Rx	[dirección del mapa de memoria en hexadecimal]
SUMAR	#Constante	Rx
MULTIPLICAR	#Constante	Rx
ADD	#Constante	Rx
XOR	#Constante	Rx

Al procesador se quieren conectar 4 LEDs y 4 interruptores (con los circuitos mostrados en la Figura 1.11), de manera que los LED se conecten a 4 líneas del bus de datos y los interruptores a otras 4 diferentes.

**FIGURA 1.11. INTERRUPTORES Y LEDs.**

- Realice la conexión ayudándose de los circuitos mostrados en la Figura 1.12 y un inversor para construir las interfaces necesarias, de manera que los LEDs se enciendan cuando se escriba un nivel bajo.

**FIGURA 1.12. INTERFACES DE CONEXIÓN.**

- Repita el mismo proceso, asumiendo que el número de LEDs son 8.
- Como se ha indicado en el enunciado, los accesos a la dirección 0xCD00 en operación de lectura activan la señal “Selección interruptores” y “Selección LEDs” cuando se accede a la misma dirección en escritura. Realice un diagrama de flujo de las operaciones a realizar para leer el código de los interruptores (el peso de los bits en el siguiente orden: SW3,SW2,SW1,SW0) y representar en los LEDs ese valor incrementado en 2.
- Escriba la secuencia de instrucciones necesarias (usando las indicadas de la Tabla 1.14) para que se realice la operación.

5. Si la señal “Selección LEDs” se activase cuando se accediera a la dirección 0xCC00 en escritura (y no al acceder a 0xCD00), ¿qué cambiaría en el código anterior?

CAPÍTULO 2

ARQUITECTURA Y MODELO DE PROGRAMACIÓN DEL CORTEX M3

INTRODUCCIÓN

A la hora de estudiar los procesadores digitales para poder hacer uso de ellos en el diseño de sistemas electrónicos, se ha dividido su análisis en dos partes. En la primera de ellas, que es la que concierne a este capítulo, se presentan ejercicios relacionados con el manejo del modelo de programación del ARM Cortex M3, del puntero de pila, el registro de enlace y el contador de programa, así como de sus registros especiales.

Otras partes importantes analizadas y a las cuales también van enfocados los ejercicios son al uso de los diferentes niveles de privilegio y modelos de programación. Sin duda, los aspectos que dan mucha potencialidad a un microprocesador son sus modos de direccionamiento, su set de instrucciones y fundamentalmente el manejo de pila. Pues bien, de todos ellos se recogen aquí ejercicios, que al final del capítulo se combinan con el uso de los propios puertos que el chip LPC-1768 pone a disposición del diseñador.

PROBLEMA 2.1.

Indique de forma concisa y clara qué ocurre al ejecutar secuencialmente las instrucciones de la Tabla 2.1 en un procesador Cortex M-3

TABLA 2.1. INSTRUCCIONES DE CORTEX M-3 PARA EL PROBLEMA 2.1.

<i>MOV r0, #0x20</i>	
<i>MSR BASEPRI, R0</i>	
<i>MOV r0, #0x01</i>	
<i>MSR FAULTMASK, R0</i>	
<i>MOV r0, #0x02</i>	
<i>MSR CONTROL, R0</i>	
<i>MOV r0, #0x01</i>	
<i>MSR CONTROL, R0</i>	
<i>MOV r0, #0x02</i>	
<i>MSR CONTROL, R0</i>	

PROBLEMA 2.2.

Asúmase que se trabaja con un procesador Cortex-M3 y que inicialmente toda la memoria RAM está inicializada a 0x00 y organizada en *Little Endian*. Cuando se ejecuta el código mostrado en la Tabla 2.2, indique los registros y las posiciones de memoria que se modifican y el valor que toman tras la ejecución de las líneas {8, 9, 10, 11, 12, 16, 17, 18, 20, 21, 24, 25, y 26}.

TABLA 2.2. CÓDIGO PARA SU EJECUCIÓN.

Nota 1: Tenga en cuenta que el programa comienza en la dirección 0x0000_0008 y que las tres primeras instrucciones ocupan 4 bytes.

Nota 2: No es necesario que indique cuando se modifica, ni el valor que tiene el contador de programa.

Nota 3: Suponga que inicialmente la memoria de datos está a 0x00

```

01 STACK_TOP EQU 0x10004000
02         AREA RESET, CODE
03         DCD STACK_TOP
04         DCD start
05         ENTRY
06
07         start
08         LDR  R0, =0x10000000
09         LDR  R1, =0x10002000
10         LDR  R1, [R0, #0x00]
11         BL  salto1
12         ADD R0, #24
13         inf
14         B  inf
15         salto1
16         PUSH (R0, R1, LR)
17         ADD  R1, #1
18         STR  R1, [R0], #4
19         MOVT R2, #0x20
20         MOVW R3, #0x3344
21         MOV   R4, #0x5566
22         MOV   R7, #0xEE11
23         MOV   R8, #0X03
24         STMIA R0!, {R7-R8}
25         STMIA R0, {R2-R4}
26         POP  {R0, R1, PC}
27
28         END

```

PROBLEMA 2.3.

Deduzca cual será el estado de la memoria y los registros, si inicialmente se encuentran en la situación mostrada en la Tabla 2.4, tras la ejecución de las instrucciones de la Tabla 2.3. Realícelo tanto si la ordenación es *big endian* como *Little endian*.

TABLA 2.3. INSTRUCCIONES DE CORTEX M-3 PARA EL PROBLEMA 2.3.

<i>PUSH { R0,R2-R4, LR}</i>
<i>ANDS R0, R6, R7</i>
<i>ADDS R9, #1</i>
<i>ADD.W R11, #0X12</i>
<i>MOV.N R6, R0</i>
<i>STRH R6, [R10],#4</i>
<i>LDMIA.W R10!, {R5,R1}</i>
<i>MOVW.W R8,#0x0010</i>
<i>MOVT.W R8,#0x0001</i>
<i>LDRH R1,[R8,#0]</i>
<i>LDRH.W R4,[R8,#2]!</i>
<i>LDRH.W R5,[R8,#2]</i>

TABLA 2.4. SITUACIÓN INICIAL DE MEMORIA Y REGISTROS PARA EL PROBLEMA 2.3.

Dirección	<i>Little endian</i>				Registros																				
	Memoria antes de la ejecución																								
0x00010000	<table border="1"> <tr><td><i>R0</i></td><td>FFFFAA</td></tr> <tr><td><i>R1</i></td><td>EEE111</td></tr> <tr><td><i>R2</i></td><td>DDDD2222</td></tr> <tr><td><i>R3</i></td><td>CCCC4444</td></tr> <tr><td><i>R4</i></td><td>BBBB5555</td></tr> <tr><td><i>R5</i></td><td>AAAA6666</td></tr> <tr><td><i>R6</i></td><td>000FF000</td></tr> <tr><td><i>R7</i></td><td>5555AAAA</td></tr> <tr><td><i>R8</i></td><td>00000000</td></tr> <tr><td><i>R9</i></td><td>00000001</td></tr> </table>	<i>R0</i>	FFFFAA	<i>R1</i>	EEE111	<i>R2</i>	DDDD2222	<i>R3</i>	CCCC4444	<i>R4</i>	BBBB5555	<i>R5</i>	AAAA6666	<i>R6</i>	000FF000	<i>R7</i>	5555AAAA	<i>R8</i>	00000000	<i>R9</i>	00000001
<i>R0</i>	FFFFAA																								
<i>R1</i>	EEE111																								
<i>R2</i>	DDDD2222																								
<i>R3</i>	CCCC4444																								
<i>R4</i>	BBBB5555																								
<i>R5</i>	AAAA6666																								
<i>R6</i>	000FF000																								
<i>R7</i>	5555AAAA																								
<i>R8</i>	00000000																								
<i>R9</i>	00000001																								
A2	03	00	01																						
FF	00	88	11																						
00	00	12	34																						
56	78	9A	BC																						
DE	F0	12	34																						
00	01	22	33																						
44	55	66	77																						
FF	00	AA	11																						
76	54	32	10																						
0x00010004																					
	00	01	11	11																					
	11	11	11	11																					
	00	01	22	33																					
	44	55	66	77																					
	FF	00	AA	11																					
	76	54	32	10																					
																					
	00	00	AA	11																					
	FF	00	AA	11																					
0x30001000	<table border="1"> <tr><td><i>R0</i></td><td>FFFFAA</td></tr> <tr><td><i>R1</i></td><td>EEE111</td></tr> <tr><td><i>R2</i></td><td>DDDD2222</td></tr> <tr><td><i>R3</i></td><td>CCCC4444</td></tr> <tr><td><i>R4</i></td><td>BBBB5555</td></tr> <tr><td><i>R5</i></td><td>AAAA6666</td></tr> <tr><td><i>R6</i></td><td>000FF000</td></tr> <tr><td><i>R7</i></td><td>5555AAAA</td></tr> <tr><td><i>R8</i></td><td>00000000</td></tr> <tr><td><i>R9</i></td><td>00000001</td></tr> </table>	<i>R0</i>	FFFFAA	<i>R1</i>	EEE111	<i>R2</i>	DDDD2222	<i>R3</i>	CCCC4444	<i>R4</i>	BBBB5555	<i>R5</i>	AAAA6666	<i>R6</i>	000FF000	<i>R7</i>	5555AAAA	<i>R8</i>	00000000	<i>R9</i>	00000001
<i>R0</i>	FFFFAA																								
<i>R1</i>	EEE111																								
<i>R2</i>	DDDD2222																								
<i>R3</i>	CCCC4444																								
<i>R4</i>	BBBB5555																								
<i>R5</i>	AAAA6666																								
<i>R6</i>	000FF000																								
<i>R7</i>	5555AAAA																								
<i>R8</i>	00000000																								
<i>R9</i>	00000001																								
A2	03	00	01																						
FF	00	88	11																						
00	00	12	34																						
56	78	9A	BC																						
DE	F0	12	34																						
00	01	22	33																						
44	55	66	77																						
FF	00	AA	11																						
76	54	32	10																						
...																						
Dirección	<i>Big endian</i>				Registros																				
	Memoria antes de la ejecución																								
0x00010000	<table border="1"> <tr><td><i>R0</i></td><td>FFFFAA</td></tr> <tr><td><i>R1</i></td><td>EEE111</td></tr> <tr><td><i>R2</i></td><td>DDDD2222</td></tr> <tr><td><i>R3</i></td><td>CCCC4444</td></tr> <tr><td><i>R4</i></td><td>BBBB5555</td></tr> <tr><td><i>R5</i></td><td>AAAA6666</td></tr> <tr><td><i>R6</i></td><td>000FF000</td></tr> <tr><td><i>R7</i></td><td>5555AAAA</td></tr> <tr><td><i>R8</i></td><td>00000000</td></tr> <tr><td><i>R9</i></td><td>00000001</td></tr> </table>	<i>R0</i>	FFFFAA	<i>R1</i>	EEE111	<i>R2</i>	DDDD2222	<i>R3</i>	CCCC4444	<i>R4</i>	BBBB5555	<i>R5</i>	AAAA6666	<i>R6</i>	000FF000	<i>R7</i>	5555AAAA	<i>R8</i>	00000000	<i>R9</i>	00000001
<i>R0</i>	FFFFAA																								
<i>R1</i>	EEE111																								
<i>R2</i>	DDDD2222																								
<i>R3</i>	CCCC4444																								
<i>R4</i>	BBBB5555																								
<i>R5</i>	AAAA6666																								
<i>R6</i>	000FF000																								
<i>R7</i>	5555AAAA																								
<i>R8</i>	00000000																								
<i>R9</i>	00000001																								
A2	03	00	01																						
FF	00	88	11																						
00	00	12	34																						
56	78	9A	BC																						
DE	F0	12	34																						
00	01	22	33																						
44	55	66	77																						
FF	00	AA	11																						
76	54	32	10																						
0x00010004																					
	00	01	11	11																					
	11	11	11	11																					
	00	01	22	33																					
	44	55	66	77																					
	FF	00	AA	11																					
	76	54	32	10																					
																					
	00	00	AA	11																					
	FF	00	AA	11																					
0x30001000	<table border="1"> <tr><td><i>R0</i></td><td>FFFFAA</td></tr> <tr><td><i>R1</i></td><td>EEE111</td></tr> <tr><td><i>R2</i></td><td>DDDD2222</td></tr> <tr><td><i>R3</i></td><td>CCCC4444</td></tr> <tr><td><i>R4</i></td><td>BBBB5555</td></tr> <tr><td><i>R5</i></td><td>AAAA6666</td></tr> <tr><td><i>R6</i></td><td>000FF000</td></tr> <tr><td><i>R7</i></td><td>5555AAAA</td></tr> <tr><td><i>R8</i></td><td>00000000</td></tr> <tr><td><i>R9</i></td><td>00000001</td></tr> </table>	<i>R0</i>	FFFFAA	<i>R1</i>	EEE111	<i>R2</i>	DDDD2222	<i>R3</i>	CCCC4444	<i>R4</i>	BBBB5555	<i>R5</i>	AAAA6666	<i>R6</i>	000FF000	<i>R7</i>	5555AAAA	<i>R8</i>	00000000	<i>R9</i>	00000001
<i>R0</i>	FFFFAA																								
<i>R1</i>	EEE111																								
<i>R2</i>	DDDD2222																								
<i>R3</i>	CCCC4444																								
<i>R4</i>	BBBB5555																								
<i>R5</i>	AAAA6666																								
<i>R6</i>	000FF000																								
<i>R7</i>	5555AAAA																								
<i>R8</i>	00000000																								
<i>R9</i>	00000001																								
A2	03	00	01																						
FF	00	88	11																						
00	00	12	34																						
56	78	9A	BC																						
DE	F0	12	34																						
00	01	22	33																						
44	55	66	77																						
FF	00	AA	11																						
76	54	32	10																						
...																						

PROBLEMA 2.4.

En un sistema digital basado en el LPC1768, una de las posibles fuentes de interrupción es la del Systick, cuya ISR se suministra en la Tabla 2.5. Además, se sabe que justo antes de ejecutarse la primera instrucción de su rutina la memoria y registros se encuentran en la situación que se muestra en la Tabla 2.6.

TABLA 2.5. INSTRUCCIONES DE LA ISR.

Fila	Contenido
1	PUSH {R4, R2, LR}
2	LDR R0,=0x10000000
3	LDMDB R0!, {R4,R2}
4	MOVT R4,# 0x01AC
5	ADD R2, #0X0C
6	STMIA R0, {R2}
7	
8	

TABLA 2.6. VALORES INICIALES DE MEMORIA Y REGISTROS.

Memoria		Registros	Contenido
Dirección	Contenido	Registro	Valor
0FFF FFF4	89 AB CD EF	R0	00 00 00 00
0FFF FFF8	AA 00 BB 11	R1	11 11 11 11
0FFF FFFC	CC 22 DD 33	R2	00 11 22 33
1000 0000	EE 44 FF 55	R3	44 55 66 77
1000 0004	66 77 88 99	R4	88 99 AA BB
1000 0008	00 11 23 45	R5	CC DD EE FF
1000 000C	67 89 AB CD	R6	CC 22 DD 33
.....		R7	EE 44 FF 55
1000 1FF4	66 77 88 99	R8	66 77 88 99
1000 1FF8	00 11 23 45	R9	00 11 23 45
1000 1FFC	67 89 AB CD	R10	AA AA AA AA
1000 2000	EF 01 23 45	R11	01 23 45 67
1000 2004	CC DD EE FF	R13	10 00 20 04
1000 2008	01 23 45 67	R14	FF FF FF F9
1000 200C	89 AB CD EF		
1000 2010	EF 01 23 45		
.....			
1000 6004	00 00 00 00		
1000 6008	00 00 00 00		
1000 600C	00 00 00 00		
1000 6010	00 00 00 00		
1000 6004	00 00 00 00		

1. Complete la(s) instrucción(es) que faltan al final de la ISR (Tabla 2.5) para que el sistema continúe funcionando correctamente.
2. Determine qué posiciones de memoria o registros se modifican cuando se ejecuta la rutina, y con qué valor. Indíquelo en la Tabla 2.7.

TABLA 2.7. REGISTROS Y DIRECCIONES DE MEMORIA MODIFICADOS; VALORES.

Línea de código	Registro o Dir. memoria	Contenido (Hexadecimal)			
1					
2					
3					
4					
5					
6					

PROBLEMA 2.5.

Trabajando con un procesador Cortex-M3 y asumiendo que inicialmente toda la memoria RAM está inicializada a 0x00, Conteste justificadamente a las siguientes preguntas:

1. Indique en la Tabla 2.8 los registros que se modifican, y con qué valor, tras ejecutar cada una de las instrucciones de dicha tabla.

TABLA 2.8. INSTRUCCIONES DE CORTEX M-3 PARA EL PROBLEMA 2.5.

	Registros	Valor (hex)
MOV R2,#5		
MOV R3,#0x04		
SUB R4,R3,R2		
ADDLE R3,R4		
NOP		

2. Indique qué registros y qué posiciones de memoria se modifican en las líneas {11, 12, 15, 16, 17, 25} indicando el valor que adquieren cuando se ejecuta el código de la Tabla 2.9.

TABLA 2.9. INSTRUCCIONES DE CORTEX M-3 PARA EL PROBLEMA 2.5.

1	STACK_TOP EQU 0x10000000		
2	AREA RESET, CODE		
3	DCD STACK_TOP		
4	DCD Reset_Handler		
5			
6	ENTRY		
7			
8			
9			
10	Reset Handler	Registro/Dir memoria	Valores
11	LDR R0.=0x10004000		
12	LDR R1.=0x10002000		
13			
14	MOV R2,#0x22		
15	MOVW R3,#0x3344		
16	MOVT R3,#0x55AA		
17	MOV R4,#0x5566		
18	MOV R5,#0xAABB		
19	MOV R6,#0xCCDD		
20	MOV R7,#0xEE11		
21	MOV R8,#0X03		
22	LDR R9.=0X10001000		
23			
24	MSR PSP, R9		
25	MSR CONTROL, R8		
26			
27			
28	STMIA R1, {R2-R4}		
29	STMIA R0!, {R5-R7}		
30	LDMDB R0, {R2-R4}		
31	LDMDB R1!, {R5-R7}		
32			
33	PUSH {R5,R3,R1,R4,R2}		
34	POP {R1,R5,R3,R2,R4}		
35	NOP		
36			
37	AREA Datos,DATA		
38	ALIGN 4		
39			
40	END		

3. Indique qué registros y qué posiciones de memoria se modifican en las líneas {11, 12, 15, 16, 17, 25} indicando el valor que adquieren cuando se ejecuta el código de la Tabla 2.10.

TABLA 2.10. INSTRUCCIONES DE CORTEX M-3 PARA EL PROBLEMA 2.5.

Linea	Instrucción	Registro/Dir memoria	Valores
1	STACK_TOP EQU 0x10000000		
2			
3	AREA RESET, CODE		
4	DCD STACK_TOP		
5	DCD Reset_Handler		
6			
7	ENTRY		
8			
9			
10	Reset_Handler	Registro/Dir memoria	Valores
11	LDR R0,=0x10003000		
12	LDR R1,=0x10001000		
13			
14	MOV R2,#0x5566		
15	MOVW R3,#0x3344		
16	MOVT R3,#0x55AA		
17	MOV R4,#0x22		
18	MOV R5,#0x1122		
19	MOV R6,#0x3344		
20	MOV R7,#0xEE11		
21	MOV R8,#0X03		
22	LDR R9,=0X10002000		
23			
24	MSR PSP, R9		
25	MSR CONTROL, R8		
26			
27			
28	STMIA R1, {R2-R4}		
29	STMIA R0!, {R5-R7}		
30	LDMDB R0, {R2-R4}		
31	LDMDB R1!, {R5-R7}		
32			
33	PUSH {R5,R3,R1,R4,R2}		
34	POP {R1,R5,R3,R2,R4}		
35	NOP		
36			
37	AREA Datos,DATA		
38	ALIGN 4		
39			
40	END		

PROBLEMA 2.6.

Dado el código de la Tabla 2.11 suministrado en ensamblador para el Cortex M3 y sabiendo que los valores de memoria y registros antes de ejecutarse la instrucción de la línea 244 son los reflejados en la Tabla 2.12 y que antes de esa línea no ha habido ninguna transferencia con la pila.

TABLA 2.11. CÓDIGO PARA EL 0.1.

Fila	Contenido
1	STACK_TOP EQU 0x10002000
2	STACK_PSP EQU 0X10006000
3	Tabla EQU 0x10000000
4	
5	AREA Reset, CODE
6	DCD STACK_TOP
7	DCD Main
8	DCD NMI
9	DCD HF
11	ENTRY
12	Main

240	inicio
244	LDR R0,=STACK_PSP
245	MSR PSP, R0
246	LDR R1,=Tabla
247	LDMDB R1!, {R2-R4}
248	ADD R1, #0X0C
249	STMIA R1, {R6-R7}
250	PUSH {R3,R6}
251	MOV R0, #1
252	MSR FAULTMASK, R0
253	LSL R0, #2
254	MSR BASEPRI, R0
255	MOV R0, #3
256	MSR CONTROL, R0
257	BL Funcion
258
259	
260	deadloop
261	B deadloop
262	
263	
264	Funcion
265	STMDB R13!, {R6-R9, R11}
266
267
268
269	LDMIA R13!, {R6-R9, R11}
270	BX LR
271	

TABLA 2.12. VALORES INICIALES.

Dir memoria	Contenido
0FFF FFE0	00 11 22 33
0FFF FFE4	44 55 66 77
0FFF FFE8	88 99 AA BB
0FFF FFEC	CC DD EE FF
0FFF FFF0	01 23 45 67
0FFF FFF4	89 AB CD EF
0FFF FFF8	AA 00 BB 11
0FFF FFFC	CC 22 DD 33
1000 0000	EE 44 FF 55
1000 0004	66 77 88 99
1000 0008	00 11 23 45
1000 000C	67 89 AB CD
1000 0010	EF 01 23 45
1000 0014	67 89 AB CD
1000 0018	EF 00 00 00
1000 001C	00 00 00 00
.....	
1000 1FE8	AA 00 BB 11
1000 1FEC	CC 22 DD 33
1000 1FF0	EE 44 FF 55
1000 1FF4	66 77 88 99
1000 1FF8	00 11 23 45
1000 1FFC	67 89 AB CD
1000 2000	EF 01 23 45
1000 2004	CC DD EE FF
1000 2008	01 23 45 67
1000 200C	89 AB CD EF
1000 2010	EF 01 23 45
1000 2014	67 89 AB CD
.....	
1000 5FF0	FF FF FF FF
1000 5FF4	FF FF FF FF
1000 5FF8	FF FF FF FF
1000 5FFC	FF FF FF FF
1000 6000	FF FF FF FF
1000 6004	00 00 00 00
1000 6008	00 00 00 00
1000 600C	00 00 00 00
1000 6010	00 00 00 00
1000 6004	00 00 00 00
Registros	
R0	00 00 00
R1	11 11 11 11
R2	00 11 22 33
R3	44 55 66 77
R4	88 99 AA BB
R5	CC DD EE FF
R6	CC 22 DD 33
R7	EE 44 FF 55
R8	66 77 88 99
R9	00 11 23 45
R10	AA AA AA AA
R11	01 23 45 67

Se pide:

- Indique qué posiciones de memoria y/o registros se modifican al ejecutarse las instrucciones de las líneas indicadas en la Tabla 2.13, y cuál es su nuevo contenido.

TABLA 2.13. DIRECCIONES, REGISTROS Y CONTENIDO.

Línea del código	Registro o Dir. memoria	Contenido (Hexadecimal)			
244					
245					
246					

- Indique qué posiciones de memoria y/o registros se modifican al ejecutarse las instrucciones de las líneas indicadas en Tabla 2.14 y cuál es su nuevo contenido. Indíquelos en el orden que se realizan las transferencias de datos.

TABLA 2.14. DIRECCIONES, REGISTROS Y CONTENIDO.

Línea del código	Registro o Dir. de memoria	Contenido (Hexadecimal)			
247					
248					
249					
250					

- Indique cuál es el efecto que se produce en la configuración de funcionamiento/estado del procesador al ejecutar las instrucciones de la Tabla 2.15, en la secuencia dada.

TABLA 2.15. DIRECCIONES, REGISTROS Y CONTENIDO.

Línea del código	Nueva situación
251	
252	
253	
254	
255	
256	

4. Sustituya las 3 instrucciones de las de líneas 265, 269 y 270 por 2 instrucciones (en 265 y 269) de manera que realicen la misma función.

PROBLEMA 2.7.

En un sistema digital basado en el LPC1768 el contenido de memoria y registros en un momento determinado es el mostrado en la Tabla 2.16. En ese instante se produce un salto a una subrutina cuyo código se muestra en la Tabla 2.17.

Se pide:

1. Indique razonadamente si el tratamiento y manejo de pila son correctos o si podría ocasionar algún problema.
2. Indique razonadamente si se podría reducir el número de instrucciones sin cambiar la funcionalidad.
3. Si se ha ejecutado la rutina de la Tabla 2.18, determine qué posiciones de memoria o registros se han modificado, y con qué valor, tras la ejecución de las instrucciones de las líneas 3 y 7.

TABLA 2.16. VALORES INICIALES DE MEMORIA Y REGISTROS.

Memoria				Registros	
Dirección	Contenido			Registro	Valor
0FFF FFF4	89	AB	CD	R0	00 00 00 00
0FFF FFF8	AA	11	00	R1	10 00 20 00
0FFF FFFC	CC	D2	22	R2	00 11 22 33
1000 0000	44	FF	55	EE	44 55 66 77
1000 0004	99	77	88	R4	88 99 AA BB
1000 0008	00	89	11	R5	CC DD EE FF
1000 000C	AB	67	CD	R6	CC 22 DD 33
1000 000E	EF	CD	23	R7	EE 44 FF 55
1000 0010	EF	01	89	R8	66 77 88 99
			R9	00 11 23 45
1000 1FFC	67	CD	D1	R10	AA AA AA AA
1000 2000	EF	EE	AB	R11	01 23 45 67
1000 2004	FF	CC	89	R13	10 00 00 04
1000 2008	01	01	CD	R14	FF FF FF F9
1000 200C	89	23	67		
1000 2010	23	45	EF		

TABLA 2.17. CÓDIGO DE LA RUTINA.

Fila	Contenido
1	STMDB R13!, {R3-R5,R14}
2	LDMIA R1, {R2-R5}
3	ADD R1, #16
4	STRH R2, [R1],#2
5	STRH R3, [R1],#2
6	STRB R4, [R1],#1
7	STRB R5, [R1],#1
8	POP {R5,R3,PC}

TABLA 2.18. POSICIONES DE MEMORIA Y REGISTROS MODIFICADOS.

Línea de código	Registro o Dir. memoria	Contenido (Hexadecimal)			
3					
7					

PROBLEMA 2.8.

El fragmento de código ensamblador para el Cortex M3 de la Tabla 2.19 rea-liza continuamente una determinada operación sobre el dato almacenado en la dirección apuntada por la etiqueta *entrada* que se recibe de un equipo remoto, y almacena el resultado en la dirección apuntada por la etiqueta *salida* a enviar al mismo equipo remoto. Cada vez que se recibe un nuevo dato una subrutina de interrupción, que aquí no se representa, lo indica poniendo la variable *esperando=1*. Finalmente, ante la llegada de un dato con un valor especial, en este caso el 0xFF, el programa detiene su funcionamiento normal y se queda en un bucle sin hacer nada.

Suponga que los valores de las etiquetas son los indicados en la Tabla 2.20. Considere asimismo el resto de valores iniciales que aparecen en la misma tabla. Los demás registros de propósito general inicialmente están a valor 0. Suponga que todas las instrucciones se codifican con 32 bits.

Se pide:

1. Indique cuál es el valor de la etiqueta calcular y el tamaño en bytes de la rutina.
2. Considerando que ha llegado un dato nuevo, indique en la Tabla 2.21 los registros que se modifican en cada una de las instrucciones, así como el valor que adquieran suponiendo que se trata de la primera ejecución del programa. Si lo requiere puede comentar la evolución del programa en la columna “Notas” de la tabla del código, pero no es obligatorio.

TABLA 2.19. CÓDIGO PARA EL PROBLEMA 2.8.

Fila	Contenido	Notas
	__main LDR R0,=esperando	
2	LDR R1, [R0]	
3	SUBS R1, #1	
4	BNE __main	
5	MOV R1, #0	
6	STR R1, [R0]	
7	LDR R0,=entrada	
8	LDR R1, [R0]	
9	CMP R1, #0xFF	
10	BEQ fin	
11	PUSH {R1}	
12	BL calcular	
13	B __main	
14	fin B fin	
15		
16	calcular POP {R0}	
17	PUSH {R1, R2}	
18	EOR R1, R0, 0xFFFFFFFF	
19	ADD R1, #1	
20	LDR R2,=salida	
21	STR R1, [R2]	
22	POP {R1, R2}	
23	BX LR	

TABLA 2.20. ETIQUETAS Y VALORES INICIALES.

Etiquetas	
entrada	= 0x10000000
salida	= 0x10000004
esperando	= 0x10000008
fin	= 0x00000134
Valores iniciales	
SP = 0x10002000	
PC = 0x00000100 (dirección de <i>main()</i>)	
Posición de memoria	
0x10000000	Valor
0xA5005A00	
0x10000008	0x00000001

TABLA 2.21. INSTRUCCIONES, REGISTROS Y VALORES MODIFICADOS PARA EL PROBLEMA 2.8.

Fila	Instrucción	Registros modificados y valores
1	LDR R0,= esperando	
2	LDR R1, [R0]	
11	PUSH {R1}	
12	BL calcular	

3. Complete la Tabla 2.22 con el contenido de la pila justo antes de ejecutarse la instrucción de la fila 22 (POP {R1, R2}). Represente también la posición del registro SP.

TABLA 2.22. CONTENIDO DE LA PILA ANTES DE LA INSTRUCCIÓN 22 DEL PROBLEMA 2.8.

Dirección ↑	Dirección	Valor	Notas
	0x1000_2000	XX XX XX XX	

4. Indique en la Tabla 2.23 el contenido de las posiciones de memoria tras ejecutarse por primera vez la subrutina *calcular*, y suponiendo que no llegan datos adicionales del equipo remoto.

TABLA 2.23. CONTENIDO DE LAS POSICIONES DE MEMORIA TRAS LA RUTINA CALCULAR DEL PROBLEMA 2.8.

Dirección	Valor
0x1000_0000	
0x1000_0004	
0x1000_0008	

5. Indique justificadamente la operación que realiza la rutina *calcular*.

PROBLEMA 2.9.

En un sistema digital basado en el LPC1768, se ejecuta el programa cuyo código se muestra en la Tabla 2.25. El contenido de memoria y registros, antes de ejecutarse la primera instrucción de la rutina es el mostrado en la Tabla 2.24, así como el valor de las etiquetas empleadas en el programa.

TABLA 2.24. VALORES INICIALES DE MEMORIA Y REGISTROS.

Memoria					Registros	
Dirección	Contenido				Registro	Valor
0000 0090	07	06	05	03	R0	00 00 00 00
0000 0094	09	02	01	00	R1	10 00 20 00
0000 0098	07	00	00	00	R2	00 11 22 33
.....					R3	44 55 66 77
1000 3FE8	00	00	00	00	R4	88 99 AA BB
1000 3FEC	00	00	00	00	R5	CC DD EE FF
1000 3FF0	00	00	00	00	R6	CC 22 DD 33
1000 3FF4	00	00	00	00	R7	EE 44 FF 55
1000 3FF8	00	00	00	00	R8	66 77 88 99
1000 3FFC	00	00	00	00	R9	00 11 23 45
1000 4000	00	00	00	00	R10	AA AA AA AA
.....					R11	01 23 45 67
1000 0000	00	00	00	00	R12	AB CB DE F0
1000 0004	00	00	00	00	R13	10 00 40 00
1000 0008	00	00	00	00	R14	FF FF FF FF
1000 000C	00	00	00	00		
Etiquetas						
start	0x00 00 00 2E					
Ret1	0x00 00 00 38					
Ret2	0x00 00 00 40					
AVGSQ	0x00 00 00 3A					
SQUARE	0x00 00 00 44					
Values	0x00 00 00 90					
Count	0x00 00 00 98					
SQR	0x10 00 00 00					

TABLA 2.25. CÓDIGO DE EJECUCIÓN.

Fila	Instrucción	
1	start	LDR R0,Count
2		LDR R1,=Values
3		LDR R2,=SQR
4		BL AVGSQ
5	Ret1	B Ret1
6		
7	AVGSQ	PUSH {R1,LR}
8		BL SQUARE
9	Ret2	LDR R1,=SQR
10		POP {R1,PC}
11		
12	SQUARE	PUSH {R0-R3,LR}
13	Loop	LDRB R3,[R1],#1
14		MUL R3,R3
15		STRB R3,[R2],#1
16		SUBS R0,R0,#1
17		BNE Loop
28		POP {R0-R3,PC}

Se pide

1. Complete la Tabla 2.26 indicando el contenido de la pila justo antes de ejecutarse la instrucción de la línea 15.

TABLA 2.26. CONTENIDO DE LA PILA.

Dirección ↓	Dirección	Contenido			
		4N	4N+1	4N+2	4N+3

2. Indique en la Tabla 2.27 el valor de los 7 primeros bytes a partir de la posición dada por la etiqueta **SQR** (0x1000_0000) después de ejecutarse la rutina **SQUARE**.

TABLA 2.27. CONTENIDO DE LA MEMORIA.

Dirección	Contenido			
	4N	4N+1	4N+2	4N+3
0x1000_0000				
0x1000_0004				

PROBLEMA 2.10.

En un sistema digital basado en el LPC1768 en el que los registros tienen el valor inicial mostrado en la Tabla 2.28 se ejecuta el código mostrado en la Tabla 2.29. A partir del análisis del código, responda a las preguntas siguientes en el espacio reservado para cada una.

TABLA 2.28. VALOR INICIAL DE LOS REGISTROS.

Registro	Valor
R0	0x0000_0000
R1	0x1000_2000
R2	0x0011_2233
R3	0x4455_6677
R4	0x8899_AABB
R5	0xCCDD_EEFF
R6	0xCC22_DD33
R7	0xEE44_FF55
R8	0x6677_8899
R9	0x0011_2345
R10	0xAAAA_AAAA
R11	0x0123_4567
MSP	0x1000_2000
PSP	0x1000_4000
R14	0xFFFF_FFF9

TABLA 2.29. CÓDIGO FUENTE.

Dirección	Cod. Inst.	Instrucción
0x00000400	0xF04F0002	MOV r0,#0x02
0x00000404	0xF3808814	MSR CONTROL,r0
0x00000408	0xF44F5000	MOV r0,#0x2000
0x0000040C	0xF44F6180	MOV r1,#0x400
0x00000410	0xB403	PUSH {r0-r1}
0x00000412	0xF000F806	BL.W 0x00000422
0x00000416	0xF2400000	MOV r0,#0x0000 W
0x0000041A	0xF2C10000	MOV r0,#0x1000 T
0x0000041E	0x7002	STRB r2,[r0,#0x00]
0x00000420	0xE7FE	B 0x00000420
0x00000422	0xBC03	POP {r0-r1}
0x00000424	0xB500	PUSH {lr}
0x00000426	0xFB0F0F1	UDIV r0,r0,r1
0x0000042A	0x4602	MOV r2,r0
0x0000042C	0xBD00	POP {pc}

Se pide

1. Indique qué pila se utiliza en el programa y cuál es la dirección más alta empleada de la misma.
2. ¿Cuántas llamadas a rutina hay, en qué direcciones de memoria se encuentran las instrucciones de llamada a las mismas, y en qué dirección comienza cada rutina?
3. ¿Cómo se pasan los parámetros a la/s rutina/s y por dónde se devuelve/n el/los resultado/s?
4. Sabiendo que la zona de memoria reservada para la pila está inicializada a 0x00, complete la siguiente Tabla 2.30 con los valores de las 12 últimas posiciones de la pila cuando el contador de programa tiene el valor 0x0000_0426. Indique la posición a la que apunta el registro puntero de pila.

TABLA 2.30. VALOR DE LAS POSICIONES DE MEMORIA.

Dirección	Contenido			
	4N	4N+1	4N+2	4N+3

5. Inicialmente el dato almacenado a partir de la dirección 0x1000_0000 es 0xAABB_CCDD. Indique en la Tabla 2.31 cuál es el nuevo contenido de dicha dirección al finalizar la ejecución del programa.

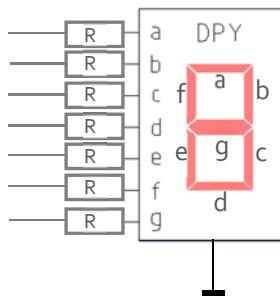
TABLA 2.31. VALOR DE LAS POSICIONES DE MEMORIA.

Dirección	4N	4N+1	4N+2	4N+3
0x1000_0000				

PROBLEMA 2.11.

A través del puerto P0 de una tarjeta basada en un procesador LPC1768 se quiere conectar un *display* de LEDs de 7 segmentos (Figura 2.1) para visualizar un número del 0 al 9. Además, se van a conectar 10 *switches* (sw0-sw9) para introducir al sistema el número que se desea visualizar (si se activa sw0 se visualizará el número 0, si se activa sw1 se visualizará el número 1, etc. Solo un *switch* puede estar activo).

El *display* se conectará por las líneas 0 (*segmento a*) a 6 (*segmento g*) del puerto P0 (GPIO0) y los *switches* por las líneas 16 a 25 del mismo puerto.

**FIGURA 2.1. CORRESPONDENCIA DE LOS SEGMENTOS DEL DISPLAY DE LED.**

Se pide:

1. Indique en la Tabla 2.32 el contenido que se debe escribir en los bits de los registros correspondientes de manera que todos los pines del puerto P0 se configuren para operar como GPIO, teniendo en cuenta que las líneas de entrada se configurarán con resistencias de *pull-up* y las de salida *sin pull-up ni pull-down*.

NOTA: Indique únicamente el valor de los bits que se deben modificar/configurar. No realice la configuración de salidas OD.

TABLA 2.32. CONFIGURACIÓN DE PINES DEL PUERTO P0.

2. Configure en la Tabla 2.33 los registros que sean necesarios para conectar el *display* a 7 líneas de salida y los *switches* a 10 líneas de entrada, indicando los valores de dicha configuración.

NOTA: Indique únicamente el valor de los bits que se deben modificar/configurar.

TABLA 2.33. CONFIGURACIÓN DE LÍNEAS PARA *DISPLAY* Y *SWITCHES*.

Registro	b 31	b 19	b 13	b 0

3. Configure en la Tabla 2.34 los registros que sean necesarios para que inicialmente el visualizador indique el número 7.

NOTA: Indique únicamente el valor de los bits que se deben modificar/configurar.

TABLA 2.34. CONFIGURACIÓN DEL VALOR INICIAL DEL DISPLAY.

Registro	b 31	b 19	b 13	b 0

PROBLEMA 2.12.

Se dispone de un microcontrolador LPC1768 al que se le han conectado un conjunto de *switches* y LEDs como muestra la Figura 2.2. La información introducida por los *switches* está codificada en BCD (SW3 es el bit más significativo) y se desea representarla en los LEDs también en BCD (P2.7 es el bit más significativo). Se pide:

1. A partir de la forma en que se encuentran conectados los *switches* y LEDs al microcontrolador, indique en la Tabla 2.35 el nombre de los registros y el contenido a programar en los mismos, para seleccionar la función GPIO, configurar los pines necesarios como entradas o salidas, y garantizar que al abrir un *switch* determinado se fije un nivel alto en el pin correspondiente. Ponga el resto de bits a ‘X’.

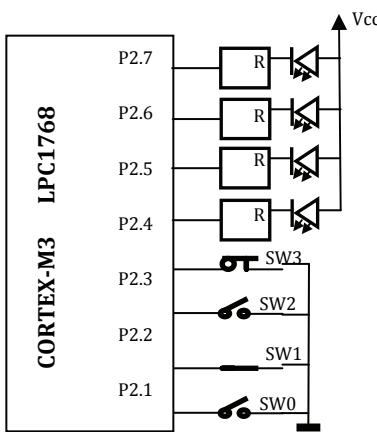
**FIGURA 2.2. CIRCUITO DEL EJERCICIO 2.8.**

TABLA 2.35. CONFIGURACIÓN DE PINES DEL PUERTO P2.

Registro	b 31	b 19	b 13	b 0

Para los siguientes apartados, las instrucciones en ensamblador y etiquetas que podríamos necesitar (no obligatoriamente) son las mostradas en la siguiente Tabla 2.36:

TABLA 2.36. ETIQUETAS E INSTRUCCIONES A UTILIZAR EN EL PROBLEMA 2.12.

Etiquetas	Instrucciones	Instrucciones
FIO2PIN = 0x2009_C054	LDR R _{destino} ,=Etiqueta	ADD R _{destino} , #Inmediato
FIO2SET = 0x2009_C058	MOV R _{destino} , R _{fuente}	AND R _{destino} , #Inmediato
FIO2CLR = 0x2009_C05C	MOV R _{destino} , #Inmediato	LDR R _{destino} , [R _{fuente}]
	STR R _{fuentes} , [R _{destino}]	LSL R _{destino} , #Inmediato

2. Escriba las instrucciones necesarias para leer la información de los *switches* y almacenar en R7 únicamente el valor del código de los *switches*.
3. Escriba las instrucciones necesarias para adecuar la información para presentarla en el *display*.

PROBLEMA 2.13.

Se dispone de un microcontrolador LPC1768 al que se le ha conectado un *display* y un pulsador tal y como muestra la Figura 2.3. Se desea que cada vez que el pulsador sea presionado el *display* cambie de la siguiente manera: si mostraba un ‘0’ que cambie a ‘1’, y si mostraba un ‘1’ que cambie a ‘0’.

Se pide:

1. Indique en la Tabla 2.37 qué registros y con qué valores se deben configurar para seleccionar adecuadamente como entrada o salida los pinos utilizados del puerto P1 (poner el resto a x), y para que inicialmente el *display* permanezca apagado, asegurando que el estado del resto de pinos de P1 que pudieran estar configurados como salida no se vean afectados.

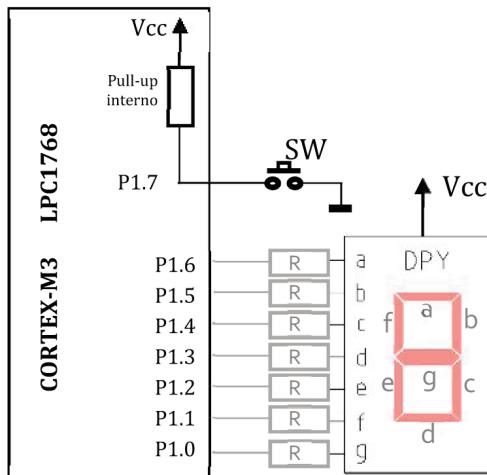


FIGURA 2.3. CIRCUITO DEL EJERCICIO 2.9.

TABLA 2.37. CONFIGURACIÓN DE PINES DEL PUERTO P1.

Registro	b 31	b 19	b 13	b 0

2. Realice un diagrama de flujo en el que se recojan los pasos a realizar para obtener la funcionalidad que el enunciado general especificas en el punto 1.

NOTA: se deben especificar en todo momento los nombres de los pines concretos sobre los que se debe actuar (consultar, escribir...). Suponga que una subrutina llamada configura realiza todas las configuraciones especificadas en el punto 1.

PROBLEMA 2.14.

A partir del procesador LPC1768 se desarrolla un sistema digital que incluye como entradas/salidas, 3 diodos LED, un *display* de 7 segmentos y dos pulsadores, conectados a determinados pines de los puertos del procesador, como se muestra en la Figura 2.4. En la Tabla 2.38 parte del código de inicialización del sistema y en la Tabla 2.39 se suministran las direcciones de algunos de los registros internos del procesador.

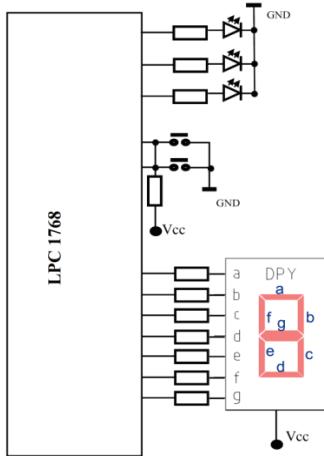


FIGURA 2.4. CIRCUITO DEL PROBLEMA 2.14.

TABLA 2.38. CÓDIGO PARA EL PROBLEMA 2.14.

Fila	Instrucción
1	LDR R0, _r = 0x4002C010
2	LDR R1,[R0]
3	AND R1, # ~((3<<0) (3<<2) (3<<4))
4	STR R1,[R0]
5	LDR R0, _r = 0x4002C050
6	LDR R1,[R0]
7	AND R1, # ~((1<<0) (1<<2) (1<<4))
8	ORR R1, # ((1<<1) (1<<3) (1<<5))
9	STR R1,[R0]
10	LDR R0, _r = 0x2009C040
11	LDR R1,[R0]
12	ORR R1, #((1<<0) (1<<1) (1<<2))
13	STR R1,[R0]
14	LDR R0, _r = 0x2009C058
15	MOV R1, #0X07
16	STR R1,[R0]
17	LDR R0, _r = 0x2009C05C
18	MOV R1, #0X07
19	STR R1,[R0]

31	LDR R0, _r = 0x4002C010
32	LDR R1,[R0]
33	AND R1, # ~((3<<20) (3<<22))
34	STR R1,[R0]
35	LDR R0, _r = 0x4002C050
36	LDR R1,[R0]
37	AND R1, # ~((3<<20) (3<<22))
38	STR R1,[R0]
39	LDR R0, _r = 0x2009C040
40	LDR R1,[R0]
41	AND R1, #~((1<<10) (1<<11))
42	STR R1,[R0]

TABLA 2.39. DIRECCIONES DE REGISTROS PARA 2.10.

Direcciones de registros	
PINSEL0 - 0x4002 C000	FIO0DIR - 0x2009 C000
PINSEL1 - 0x4002 C004	FIO1DIR - 0x2009 C020
PINSEL2 - 0x4002 C008	FIO2DIR - 0x2009 C040
PINSEL3 - 0x4002 C00C	FIO3DIR - 0x2009 C060
PINSEL4 - 0x4002 C010	
PINSEL7 - 0x4002 C01C	FIO0SET - 0x2009 C018
	FIO1SET - 0x2009 C038
PINMODE0 0x4002 C040	FIO2SET - 0x2009 C058
PINMODE1 0x4002 C044	FIO3SET - 0x2009 C078
PINMODE2 0x4002 C048	
PINMODE3 0x4002 C04C	FIO0CLR - 0x2009 C01C
PINMODE4 0x4002 C050	FIO1CLR - 0x2009 C03C
PINMODE5 0x4002 C054	FIO2CLR - 0x2009 C05C
PINMODE6 0x4002 C058	FIO3CLR - 0x2009 C07C
PINMODE7 0x4002 C05C	

Se pide:

- Indique en la Tabla 2.40 qué registros del LPC1768 se están configurando con cada conjunto de instrucciones

TABLA 2.40. REGISTROS CONFIGURADOS.

Filas conjunto instrucciones	Registro configurado
1 a 4	
5 a 9	
10 a 13	
14 a 16	
17 a 19	
31 a 34	
35 a 38	
39 a 42	

- A partir del análisis del primer segmento de código (líneas 1 a 19) determine:
 - ¿Qué periférico y a qué pines de qué puerto está conectado? Justifíquelo.
 - Tras la inicialización ¿los LEDs se encuentran encendidos o apagados? Justifíquelo.
- A partir del análisis del segundo segmento de código (líneas 31 a 42) determine qué periférico se está configurando y a qué pines se conecta.

4. Tras realizar una prueba de depuración de hardware y software se determina que en la conexión de los pulsadores existe un error con la resistencia externa, ya que cortocircuita los dos pulsadores. Considerando el fragmento de software de configuración de los mismos, ¿qué modificaciones habría que hacer sobre el hardware para resolverlo? ¿Sería necesario mantener la resistencia de *pull-up* externa?

PROBLEMA 2.15.

A los puertos de un procesador LPC176x se conectan los periféricos que se muestran en la Figura 2.5. De ellos se sabe que: la entrada del pulsador debe leer un nivel bajo cuando no está pulsado; las entradas de los interruptores leen un nivel alto cuando están abiertos, y los diodos se deben encender con un nivel bajo. En la propia Figura 2.5 se indica a qué pin está conectado cada elemento y en la Tabla 2.41 se suministran las direcciones de algunos de los registros internos del procesador.

Se pide:

1. Sobre la Figura 2.5 complete las conexiones que faltan en las 3 cajas grises de trazo discontinuo para que el sistema funcione como se ha especificado.

TABLA 2.41. DIRECCIONES DE REGISTROS DEL LPC176x.

PINSEL0 - 0x4002 C000	FIO0DIR - 0x2009 C000
PINSEL1 - 0x4002 C004	FIO1DIR - 0x2009 C020
PINSEL2 - 0x4002 C008	FIO2DIR - 0x2009 C040
PINSEL3 - 0x4002 C00C	FIO3DIR - 0x2009 C060
PINSEL4 - 0x4002 C010	
PINSEL7 - 0x4002 C01C	FIO0SET - 0x2009 C018
	FIO1SET - 0x2009 C038
PINMODE0 0x4002 C040	FIO2SET - 0x2009 C058
PINMODE1 0x4002 C044	FIO3SET - 0x2009 C078
PINMODE2 0x4002 C048	
PINMODE3 0x4002 C04C	FIO0CLR - 0x2009 C01C
PINMODE4 0x4002 C050	FIO1CLR - 0x2009 C03C
PINMODE5 0x4002 C054	FIO2CLR - 0x2009 C05C
PINMODE6 0x4002 C058	FIO3CLR - 0x2009 C07C
PINMODE7 0x4002 C05C	

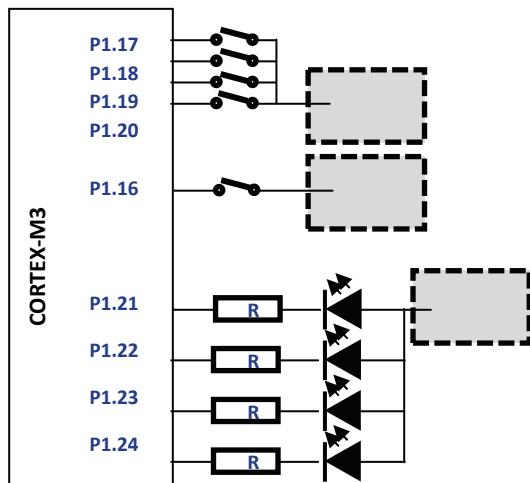


FIGURA 2.5. CONEXIÓN DEL CIRCUITO PARA 2.12.

2. Indique en las Tabla 2.42 qué registros habría que configurar y con qué valores, para definir el funcionamiento del puerto. Debe incluir la selección de funcionalidad del pin, conexión o no de resistencias de *pull-up/pull-down* y dirección de los pines. Además, inicialmente los diodos LED's deben estar apagados. Indique solamente los bits que sería necesario modificar, dejando en blanco los restantes.

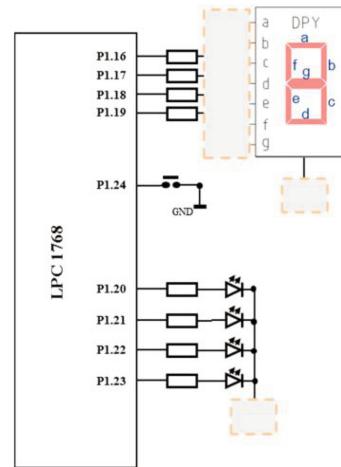
TABLA 2.42. CONFIGURACIÓN DE REGISTROS INTERNOS DEL PROCESADOR.

PROBLEMA 2.16.

A los puertos de un procesador LPC176x se conectan los periféricos que se muestran en la Figura 2.6. De ellos se sabe que: el *display* es de ánodo común y que solamente se representarán en él los números 1 y 4; la entrada del pulsador debe leer un nivel alto cuando no está pulsado; y los diodos se deben encender o apagar en función del nivel enviado. En la propia Figura 2.6 se indica a qué pin está conectado cada elemento y en la Tabla 2.43 se suministran las direcciones de algunos de los registros internos del procesador. En el anexo se suministra información sobre los códigos de configuración de los registros de los puertos.

TABLA 2.43. DIRECCIONES DE REGISTROS INTERNOS.

PINSEL0 - 0x4002 C000	FIO0DIR - 0x2009 C000
PINSEL1 - 0x4002 C004	FIO1DIR - 0x2009 C020
PINSEL2 - 0x4002 C008	FIO2DIR - 0x2009 C040
PINSEL3 - 0x4002 C00C	FIO3DIR - 0x2009 C060
PINSEL4 - 0x4002 C010	
PINSEL7 - 0x4002 C01C	FIO0SET - 0x2009 C018
	FIO1SET - 0x2009 C038
PINMODE0 0x4002 C040	FIO2SET - 0x2009 C058
PINMODE1 0x4002 C044	FIO3SET - 0x2009 C078
PINMODE2 0x4002 C048	
PINMODE3 0x4002 C04C	FIO0CLR - 0x2009 C01C
PINMODE4 0x4002 C050	FIO1CLR - 0x2009 C03C
PINMODE5 0x4002 C054	FIO2CLR - 0x2009 C05C
PINMODE6 0x4002 C058	FIO3CLR - 0x2009 C07C
PINMODE7 0x4002 C05C	



**FIGURA 2.6.
CONEXIÓN DEL CIRCUITO.**

Se pide:

1. Sobre la Figura 2.6 complete las conexiones que faltan en las 3 cajas grietas de trazo discontinuo para que el sistema funcione como se ha especificado.
2. Complete sobre la Tabla 2.44 el programa en ensamblador de configuración para que funcione de la forma descrita en el enunciado (que incluya la selección de funcionalidad del pin, conexión o no de resistencias de *pull-up/pull-down*, entrada/salida, etc.), sabiendo que la parte de código que se suministra se corresponde con la configuración de la funcionalidad de los pines. Los bits no implicados no se deben modificar.

TABLA 2.44. PROGRAMA DE CONFIGURACIÓN.

Fila	Instrucción
1	LDR R0, = 0x4002C00C
2	LDR R1,[R0]
3	MOV R2, # 0x0000
4	MOVT R2, # 0xFFFF
4	AND R1, R2
6	STR R1,[R0]

.....

3. Realice sobre la Tabla 2.45 el programa en ensamblador de inicialización para que el *display* muestre el número 4 y los diodos estén apagados. Los bits no implicados no se deben modificar.

TABLA 2.45. PROGRAMA DE INICIALIZACIÓN DEL DISPLAY.

Fila	Instrucción

PROBLEMA 2.17.

Un determinado sistema que utiliza un procesador Cortex-M3 se conecta con dos circuitos, uno que genera 2 señales y otro que recibe tres señales, como se muestra en la Figura 2.7. En la Tabla 2.46 se suministran las direcciones de los registros de configuración y acceso a los puertos y en la Tabla 2.47 parte del código de programación de dicho sistema.

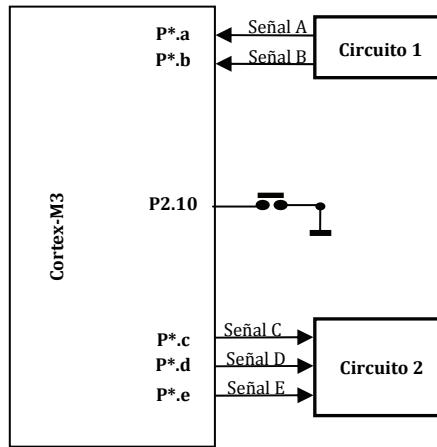


FIGURA 2.7. CIRCUITO.

TABLA 2.46.
DIRECCIONES DE REGISTROS.

PINSEL0 - 0x4002 C000	FIO0DIR - 0x2009 C000
PINSEL1 - 0x4002 C004	FIO1DIR - 0x2009 C020
PINSEL2 - 0x4002 C008	FIO2DIR - 0x2009 C040
PINSEL3 - 0x4002 C00C	FIO3DIR - 0x2009 C060
PINSEL4 - 0x4002 C010	
PINSEL7 - 0x4002 C01C	FIO0SET - 0x2009 C018
	FIO1SET - 0x2009 C038
PINMODE0 0x4002 C040	FIO2SET - 0x2009 C058
PINMODE1 0x4002 C044	FIO3SET - 0x2009 C078
PINMODE2 0x4002 C048	
PINMODE3 0x4002 C04C	FIO0CLR - 0x2009 C01C
PINMODE4 0x4002 C050	FIO1CLR - 0x2009 C03C
PINMODE5 0x4002 C054	FIO2CLR - 0x2009 C05C
PINMODE6 0x4002 C058	FIO3CLR - 0x2009 C07C
PINMODE7 0x4002 C05C	FIO0 PIN - 0X2009 C014
	FIO1 PIN - 0X2009 C034
	FIO2 PIN - 0X2009 C054
	FIO3 PIN - 0X2009 C074

TABLA 2.47.
CÓDIGO ENSAMBLADOR.

```

1   FUNCION_A    EQU 0x4002C008
2   RESISTENCIA_A EQU 0x4002C048
3   DIRECCION_A   EQU 0x2009C020
4   DESACTIVAR_A  EQU 0x2009C03C
5   ACTIVAR_A    EQU 0x2009C038
6   LEER_A        EQU 0X2009C034
7   FUNCION_B    EQU 0x4002C010
8   RESISTENCIA_B EQU 0x4002C050
9   DIRECCION_B   EQU 0x2009C040
10  LEER_B       EQU 0X2009C054
11
12  STACK_TOP    EQU 0x10002000
13
14  AREA_RESET, CODE
15  DCD STACK_TOP
16  DCD start
17
18  ENTRY
19  start
20      LDR R0, =FUNCION_A
21      LDR R1, [R0]
22      MOV R2, # 0xfc00
23      AND R1, R2
24      STR R1, [R0]
25      LDR R0, =RESISTENCIA_A
26      LDR R1, [R0]
27      MOV R2, # 0x02AA
28      ORR R1, R2
29      MOV R2, # 0xFEAA
30      AND R1, R2
31      STR R1, [R0]
32      LDR R0, =DIRECCION_A
33      LDR R1, [R0]
34      ORR R1, # ((1<<2) | (1<<3) | (1<<4))
35      AND R1, # ~ (3)
36      STR R1, [R0]
37      .....

```

Se pide:

1. Analizando el código de la Tabla 2.47 determine qué pines se están configurando y cómo, y asócielos con las conexiones de la Figura 2.7.
2. Explique cómo y en qué registros configuraría el pin P2.10 para que funcione correctamente según su conexión en la Figura 2.7 (no utilice más de 10 palabras para ello).
3. Si las señales generadas por el circuito 1 (Figura 2.8) son las indicadas en la Figura 2.7, determine analizando el código de la Tabla 2.48, cuáles serían las señales generadas por el procesador y represéntelas en la misma Figura 2.8 para dos situaciones: 1) asumiendo que el pulsador de P2.10 está pulsado; y 2) asumiendo que estaba pulsado y se deja de pulsar.

SITUACIÓN 1

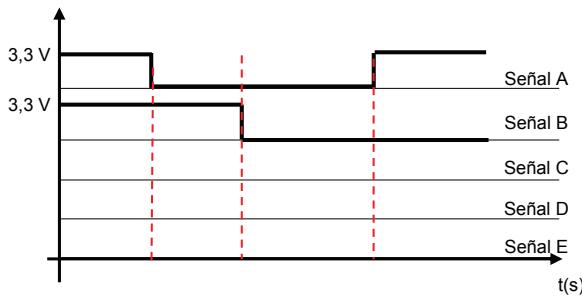


TABLA 2.48.

CÓDIGO ENSAMBLADOR.

```

60 PROCESA
61   LDR R0, =LEER_B
62   LDR R1, [R0]
63   ANDS R1, # (1<<10)
64   BNE PROCESA
65   LDR R0, =LEER_A
66   LDR R1, [R0]
67   AND R1, #3
68   AND R2, R1, #1
69   LSR R1, #1
70   EOR R2, R1
71   LSL R2, #4
72   ORR R2, #(1<<2)
73   STR R2, [R0]
74   B PROCESA
75

```

SITUACIÓN 2

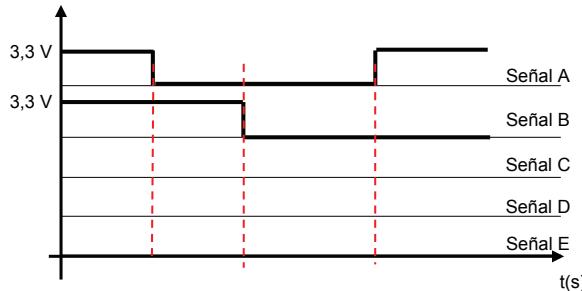


FIGURA 2.8. SEÑALES GENERADAS POR EL CIRCUITO.

PROBLEMA 2.18.

Se ha diseñado un sistema digital basado en el microcontrolador LPC1768 al que se conectan dos pulsadores activos a nivel bajo y tres diodos LED's como se representa en la Figura 2.9. Los dos pulsadores deben funcionar provocando una interrupción al ser pulsados.

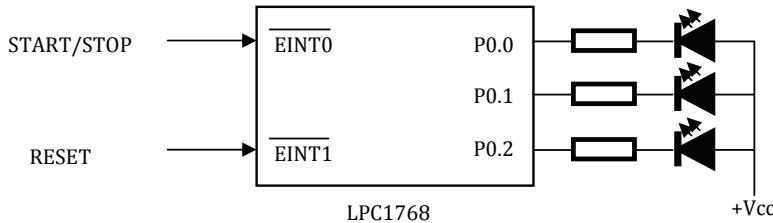


FIGURA 2.9. SISTEMA DIGITAL BASADO EN EL LPC1768.

El funcionamiento del sistema es el siguiente:

- Al iniciar el sistema, comienza en un estado de reposo en el que todos los LED's están apagados.
- Al presionar la primera vez el pulsador de **Start/Stop**, los LED's comienzan a cambiar de estado con una cadencia determinada para cada LED.
- Al volver a presionar el pulsador **Start/Stop** los LEDs mantienen su último estado.
- El pulsador de **Reset** sitúa al sistema en el estado de reposo inicial.

Para implementar este sistema se hace uso del temporizador Systick, el cual se configura para que provoque una interrupción cada milisegundo.

Se pide:

1. Complete la Tabla 2.49 con los valores en hexadecimal a los que habría que configurar los registros del Systick para que funcione según las especificaciones.

TABLA 2.49. VALORES DE CONFIGURACIÓN DEL SYSTICK.

Registro	Valor (hexadecimal)

2. Indique en la Tabla 2.50 qué registros del módulo pin connect Block habría que configurar y con qué valores, para definir el funcionamiento de los terminales empleados. Indique solamente los bits que sería necesario modificar, dejando en blanco los restantes. No es necesario configurar los registros *PINMODE_ODx*.

TABLA 2.50. VALORES DE CONFIGURACIÓN DE REGISTROS.

Bit Registro	3 1	... 2	2 0	1 6	1 2	8	4	0

3. Sabiendo que la interrupción más prioritaria es la del pulsador de **Reset**, y que debe desalojar a cualquiera de las otras dos interrupciones hardware, que la siguiente en prioridad es la interrupción del Systick, que la interrupción del pulsador **Start/Stop** no se ve desalojada por la interrupción del Systick, y que el grupo de prioridad se ha configurado a valor 6, complete la Tabla 2.51 indicando la prioridad en binario que se debería asignar a cada interrupción.

TABLA 2.51.

Interrupción	Valor 8 bits (binario)

4. Suponiendo que las prioridades asignadas han sido 0x60, 0x70 y 0x71, configure todos los registros necesarios, tanto del *NVIC* como del *core*, para que esas interrupciones puedan ser atendidas.

CAPÍTULO 3

EL SISTEMA DE EXCEPCIONES DEL CORTEX M3

INTRODUCCIÓN

Para poder desarrollar aplicaciones con un microcontrolador es fundamental comprender cómo funciona su sistema de gestión de excepciones e interrupciones, conocer cómo se tratan sus prioridades y cómo y dónde se configuran, saber dónde está la tabla de vectores y saber realizar funciones de atención a las diferentes excepciones. También es importante conocer las interrupciones externas de las que dispone y cómo se configuran.

Una excepción es un evento que se puede producir por diferentes causas y que, cuando se produce, si está habilitado, provoca la ejecución inmediata de una rutina de atención a la excepción interrumpiendo el programa que se estuviera ejecutando en ese momento. En la mayor parte de las excepciones, al finalizar la rutina de atención a la misma, se produce un retorno al programa que se estaba ejecutando.

Las interrupciones son un caso particular de excepciones donde el evento es producido por un periférico de la CPU (situados dentro del microcontrolador) o por interrupciones externas.

Los ejercicios que se proponen en este capítulo están basados en el sistema de excepciones del microcontrolador LPC1768. Para resolverlos se requiere el manual de usuario “LPC176x/5x User manual”. En particular es necesario consultar el capítulo 6 de dicho manual: “Chapter 6: LPC176x/5x Nested Vectored Interrupt Controller (NVIC)”

PROBLEMA 3.1.

Suponiendo que después de un *Reset* no se ha modificado el registro VTOR de un microprocesador LPC17xx, indique qué significado tienen los valores almacenados en las localizaciones de memoria indicadas en la Tabla 3.1:

TABLA 3.1. SIGNIFICADO DE POSICIONES DE MEMORIA.

DIR	Contenido	Significado
0x0000_0000	0x1000_8000	
0x0000_0004	0x0000_01A8	
0x0000_0008	0x0000_02B4	

PROBLEMA 3.2.

Un sistema basado en el microcontrolador LPC17xx dispone de 3 entradas de IRQs externas: EINT1, EINT2 e EINT3 cuyos valores de prioridad son:

Prioridad EINT1: 0x40

Prioridad EINT2: 0x20

Prioridad EINT3: 0x60

Si se produce la secuencia de estados de las mismas (en el orden indicado en la Tabla 3.2), cuando los eventos son los indicados, determine razonadamente cuál o cuáles pueden ser los grupos de prioridad.

TABLA 3.2. ESTADOS DE LAS INTERRUPCIONES.

Orden	Evento	Estado (x, P, A)		
		EINT1	EINT2	EINT3
1	Solicita atención EINT3	x	x	A
2	Solicita atención EINT1	P	x	A
3	Solicita atención EINT2	P	A	A
4			

x → (ha terminado o no ha solicitado); P → Pending; A → Active

PROBLEMA 3.3.

Dado el código en lenguaje C para un LPC1768 y el segmento de memoria correspondiente a la tabla de vectores mostrado en la Figura 3.1:

```
#include <LPC17xx.H>
char a=1, b=1, c=1;

void NMI_Handler()          {a++;}
void EINT3_IRQHandler()    {b++;}
void TIMER1_IRQHandler()   {c=a+b;}

main ()      {
    NVIC_SetPriorityGrouping(4);
    NVIC_SetPriority(EINT3_IRQn, 0x0);
    NVIC_SetPriority(TIMER1_IRQn, 0x2);
    NVIC_EnableIRQ(EINT3_IRQn);
    NVIC_EnableIRQ(TIMER1_IRQn);
    while(1); }
```

0x0000000000:	68 02 00 10
0x0000000004:	6D 01 00 00
0x0000000008:	CB 01 00 00
0x000000000C:	73 01 00 00
0x0000000010:	75 01 00 00
0x0000000014:	77 01 00 00
0x0000000018:	79 01 00 00
0x000000001C:	00 00 00 00
0x0000000020:	00 00 00 00
0x0000000024:	00 00 00 00
0x0000000028:	00 00 00 00
0x000000002C:	7B 01 00 00
0x0000000030:	7D 01 00 00
0x0000000034:	00 00 00 00
0x0000000038:	7F 01 00 00
0x000000003C:	81 01 00 00
0x0000000040:	83 01 00 00
0x0000000044:	83 01 00 00
0x0000000048:	E7 01 00 00
0x000000004C:	83 01 00 00
0x0000000050:	83 01 00 00

FIGURA 3.1. TABLA DE VECTORES (LITTLE ENDIAN).

Conteste justificadamente a las siguientes preguntas:

1. Ordene las interrupciones de mayor a menor prioridad.
2. Indique qué interrupciones pueden interrumpir (desalojar) al servicio de atención de la interrupción del Timer 1

3. ¿Cuál es el valor inicial del stack pointer?
4. ¿Cuál es la dirección de inicio de la rutina de atención de la interrupción del NMI?
5. Suponiendo que el contador de programa (PC) tiene el valor 0x0000_01E6, que en esa dirección se encuentra el segmento de código en ensamblador mostrado en la Figura 3.2 correspondiente a una de las rutinas de atención a la interrupción del código dado y que no ha habido ninguna interrupción previa. **Nota:** Tenga en cuenta el número de vector que corresponde a cada excepción: NMI el 2, TIMER1 el 18 y EINT3 el 37.

```

0x0000001E6 481F      LDR      r0,[pc,#124] ; 0x000000264
0x0000001E8 7800      LDRB     r0,[r0,#0x00]
0x0000001EA 491F      LDR      r1,[pc,#124] ; 0x000000268
0x0000001EC 7809      LDRB     r1,[r1,#0x00]
0x0000001EE 4408      ADD      r0,r0,r1
0x0000001F0 491E      LDR      r1,[pc,#120] ; 0x00000026C
0x0000001F2 7008      STRB    r0,[r1,#0x00]
0x0000001F4 4770      BX       lr

```

FIGURA 3.2. FRAGMENTO DE CÓDIGO EN ENSAMBLADOR.

- ¿Qué rutina de atención a la interrupción se está ejecutando?
- ¿Qué valor tiene el stack pointer al inicio del segmento de código?

PROBLEMA 3.4.

Se ha programado una aplicación para el microcontrolador LPC1788 configurado en *little endian* que hace uso de la interrupción externa EINT1. El sistema está conectado a un periférico de entrada y uno de salida según se muestra en la Figura 3.3, ambos de 32 bits y mapeados en las direcciones que se indican en la figura.

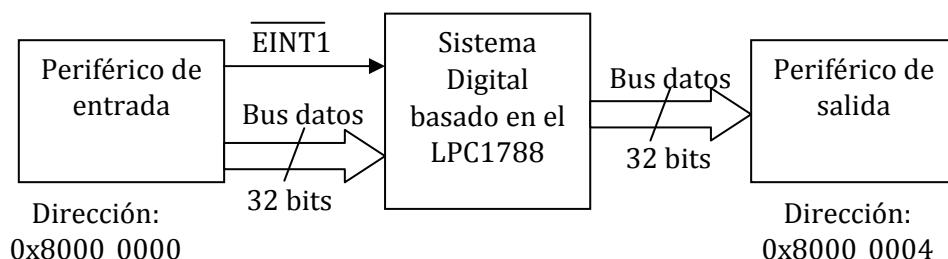


FIGURA 3.3. DIAGRAMA DEL SISTEMA DIGITAL DISEÑADO.

Cuando el periférico de entrada tiene un dato (codificado en complemento a 2) listo solicita interrupción, la rutina de atención a la interrupción (ISR) lee el dato y calcula su valor absoluto y a continuación lo escribe en el periférico de salida. El sistema se quedará esperando nuevas interrupciones.

En la Tabla 3.3 se muestra un fragmento del código que se ejecuta en el microcontrolador. No se ha incluido el código de la rutina de configuración de las interrupciones (ConfigInt), ni el de la rutina de borrado de flag (BorraFlag).

Como puede observar en el código, el programa principal llama a la rutina ConfigInt y a continuación se queda en un bucle infinito, esperando a que se produzcan interrupciones. Cuando el periférico de entrada interrumpe, se ejecuta la rutina ISR_EINT1, que a su vez llama a la rutina BorraFlag que únicamente borra el flag de la interrupción y luego llama la rutina Absoluto para calcular el valor absoluto del dato de entrada.

TABLA 3.3. CÓDIGO EN ENSAMBLADOR.

Fila	Dirección	COP	Instrucción		
1	0x00001008	0xF000F815	ProgPrincipal	BL	ConfigInt
2	0x0000100C	0xE7FE	Deadloop	B	Deadloop
3			; ISR de la interrupción	EINT1	
4	0x0000100E	0xB500	ISR_EINT1	PUSH	{LR}
5	0x00001010	0xF000F814		BL	BorraFlag
6	0x00001014	0xF04F4100		LDR	R1,=0x80000000
7	0x00001018	0xF9910000		LDR	R0, [R1]
8	0x0000101C	0xF000F805		BL	Absoluto
9	0x00001020	0x4909		LDR	R1,=0x80000004
10	0x00001022	0x7008		STR	R0, [R1]
11	0x00001024	0XF85DEB04		POP	{LR}
12	0x00001028	0x4770		BX	LR
13			; Rutina para calcular el valor absoluto		
14	0x0000102A	0xB402	Absoluto	PUSH	{R1}
15	0x0000102C	0x2800		CMP	R0,#0
16	0x0000102E	0xD503		BPL	Positivo
17	0x00001030	0XF04F0100		MOV	R1,#0
18	0x00001034	0XEBA10000		SUB	R0,R1,R0
19	0x00001038	0xBC02	Positivo	POP	{R1}
20	0x0000103A	0x4770		BX	LR

Se pide:

1. Teniendo en cuenta que el valor inicial de los registros después de un *Reset* es el que se muestra a continuación, y que las únicas excepciones que se van a producir en el sistema serán el *Reset* y la interrupción EINT1 (corresponde con en número de vector 35), indique qué posiciones de la tabla de vectores es necesario inicializar, así como su valor.

R0: 0x00000000	R5: 0x00000000	R10: 0x00000000	R15: 0x00001008
R1: 0x00000000	R6: 0x00000000	R11: 0x00000000	xPSR: 0x01000000
R2: 0x00000000	R7: 0x00000000	R12: 0x00000000	
R3: 0x00000000	R8: 0x00000000	R13: 0x10004000	
R4: 0x00000000	R9: 0x00000000	R14: 0xFFFFFFFF	

2. Suponiendo que se ha producido la interrupción EINT1, represente en la Tabla 3.4 el contenido de la pila cuando el microcontrolador esté ejecutando la instrucción BPL Positivo (fila 16). Indique dirección y registros almacenados. Si un registro está almacenado varias veces indique desde qué rutina se ha almacenado cada vez.

TABLA 3.4. CONTENIDO DE LA PILA.

3. Suponiendo que es la primera vez que se ejecuta la ISR y que el dato que entrega el periférico es el 0xFFFF_FFFE, indique el valor del dato que se escribe en el periférico de salida, en hexadecimal.
 4. A partir de la información proporcionada por el programa ensamblador determine el tamaño de las rutinas ISR_EINT1 y Absoluto en decimal.

PROBLEMA 3.5.

Dado el programa de la Figura 3.4:

```
#include <LPC17xx.H>
char a=1, b=1, c=1;
int n=0;

void SysTick_Handler (void) {
    n++;
}

void inicio()
{
    LPC_PINCON->PINSEL4 |= 1 << (12*2);
    LPC_PINCON->PINSEL4 |= 1 << (11*2);
    LPC_PINCON->PINSEL4 |= 1 << (10*2);

    NVIC_SetPriorityGrouping(3);
    NVIC_SetPriority(EINT0_IRQn, 0x2);
    NVIC_SetPriority(EINT1_IRQn, 0x4);
    NVIC_SetPriority(EINT2_IRQn, 0x5);
    NVIC_SetPriority (SysTick_IRQn, 0);

    NVIC_EnableIRQ(EINT0_IRQn);
    NVIC_EnableIRQ(EINT1_IRQn);
    NVIC_EnableIRQ(EINT2_IRQn);

    SysTick->LOAD = SysTick->CALIB;
    SysTick->VAL = 0;
    SysTick->CTRL = 0x7;
}

void EINT0_IRQHandler()
{
    LPC_SC->EXTINT |= (1 << 0);
    n=0;
    while(a) {
        if (n==200) { a =0; n=0; }
    }
    a=1;
}

void EINT1_IRQHandler()
{
    LPC_SC->EXTINT |= (1 << 1);
    n=0;
    while(b) {
        if (n==100) { b =0; n=0; }
    }
    b=1;
}

void EINT2_IRQHandler()
{
    LPC_SC->EXTINT |= (1 << 2);
    n=0;
    while(c) {
        if (n==300) { c =0; n=0; }
    }
    c=1;
}

main ()
{
    inicio();
    while(1);
}
```

FIGURA 3.4. PROGRAMA EN LENGUAJE C.

1. A partir del cronograma de las señales en las entradas EINT0, EINT1 y EINT2, que se muestra en la Figura 3.5, rellene en la Tabla 3.5 el estado de cada una de dichas interrupciones (A-Activa, P-Pendiente o X- otro estado), para los instantes de tiempos mostrados en la primera columna. Tenga en cuenta que el fin de las rutinas de atención a las interrupciones EINT0, EINT1 y EINT2, depende del valor de n que se modifica en la rutina de atención del Systick y que la función NVIC_SetPriority() pone el valor de prioridad en los 5 bits más significativos del registro de prioridad correspondiente a la interrupción en cuestión.

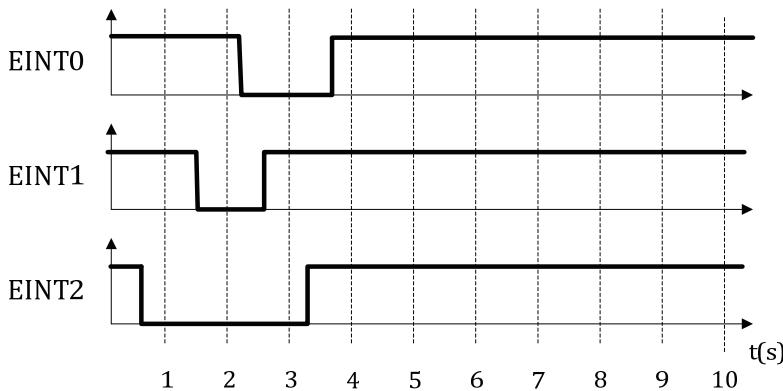


FIGURA 3.5. CRONOGRAMA DE SEÑALES DE INTERRUPCIÓN.

TABLA 3.5. ESTADOS DE LAS INTERRUPCIONES.

t (s)	Estado EINT0	Estado EINT1	Estado EINT2
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

2. Teniendo en cuenta que no se ha modificado el registro VTOR, que el contenido de la memoria a partir de la dirección 0x0000_0000 es la que se muestra en la Figura 3.6 y que los números de vectores Systick, EINT0, EINT1 y EINT2 son 15, 34, 35 y 36 respectivamente, diga:
- En qué direcciones comienzan las subrutinas de atención de las interrupciones Systick, EINT0, EINT1 y EINT2.
 - En qué dirección comienza el programa principal (valor inicial del contador del programa tras un *Reset*).
 - Valor inicial del Puntero de Pila tras un *Reset*.

```
:| 0x00000000: 68 02 00 10  
:| 0x00000004: 6D 01 00 00  
:| 0x00000008: 71 01 00 00  
:| 0x0000000C: 73 01 00 00  
:| 0x00000010: 75 01 00 00  
:| 0x00000014: 77 01 00 00  
:| 0x00000018: 79 01 00 00  
:| 0x0000001C: 00 00 00 00  
:| 0x00000020: 00 00 00 00  
:| 0x00000024: 00 00 00 00  
:| 0x00000028: 00 00 00 00  
:| 0x0000002C: 7B 01 00 00  
:| 0x00000030: 7D 01 00 00  
:| 0x00000034: 00 00 00 00  
:| 0x00000038: 7F 01 00 00  
:| 0x0000003C: 2B 03 00 00  
:| 0x00000040: 83 01 00 00  
:| 0x00000044: 83 01 00 00  
:| 0x00000048: 83 01 00 00  
:| 0x0000004C: 83 01 00 00  
:| 0x00000050: 83 01 00 00  
:| 0x00000054: 83 01 00 00  
:| 0x00000058: 83 01 00 00  
:| 0x0000005C: 83 01 00 00  
:| 0x00000060: 83 01 00 00  
:| 0x00000064: 83 01 00 00  
:| 0x00000068: 83 01 00 00  
:| 0x0000006C: 83 01 00 00  
:| 0x00000070: 83 01 00 00  
:| 0x00000074: 83 01 00 00  
:| 0x00000078: 83 01 00 00  
:| 0x0000007C: 83 01 00 00  
:| 0x00000080: 83 01 00 00  
:| 0x00000084: 83 01 00 00  
:| 0x00000088: CD 03 00 00  
:| 0x0000008C: 0B 04 00 00  
:| 0x00000090: 49 04 00 00  
:| 0x00000094: 83 01 00 00
```

FIGURA 3.6. CONTENIDO DE LA TABLA DE VECTORES.

PROBLEMA 3.6.

Dado el programa de la Figura 3.7:

```
#include <LPC17xx.H>
char a=1, b=1, c=1;
int n=0;

void SysTick_Handler (void) {
    n++;
}

void inicio()
{
    LPC_PINCON->PINSEL4 |= 1 << (12*2);
    LPC_PINCON->PINSEL4 |= 1 << (11*2);
    LPC_PINCON->PINSEL4 |= 1 << (10*2);

    NVIC_SetPriorityGrouping(4);
    NVIC_SetPriority(EINT0_IRQn, 0x2);
    NVIC_SetPriority(EINT1_IRQn, 0x4);
    NVIC_SetPriority(EINT2_IRQn, 0x5);
    NVIC_SetPriority(SysTick_IRQn, 0);

    NVIC_EnableIRQ(EINT0_IRQn);
    NVIC_EnableIRQ(EINT1_IRQn);
    NVIC_EnableIRQ(EINT2_IRQn);

    SysTick->LOAD = SysTick->CALIB;
    SysTick->VAL = 0;
    SysTick->CTRL = 0x7;
}
}

void EINT0_IRQHandler()
{
    LPC_SC->EXTINT |= (1 << 0);
    n=0;
    while(a) {
        if (n==200) { a =0; n=0; }
    }
    a=1;
}

void EINT1_IRQHandler()
{
    LPC_SC->EXTINT |= (1 << 1);
    n=0;
    while(b) {
        if (n==100) { b =0; n=0; }
    }
    b=1;
}

void EINT2_IRQHandler()
{
    LPC_SC->EXTINT |= (1 << 2);
    n=0;
    while(c) {
        if (n==300) { c =0; n=0; }
    }
    c=1;
}

main ()
{
    inicio();
    while(1);
}
```

FIGURA 3.7. PROGRAMA EN LENGUAJE C.

1. A partir del cronograma de las señales en las entradas EINT0, EINT1 y EINT2, mostrado en la Figura 3.8, rellene en la Tabla 3.6 el estado de cada una de dichas interrupciones (A-Activa, P-Pendiente o X- otro estado), para los instantes de tiempos mostrados en la primera columna. Tenga en cuenta que el fin de las rutinas de atención a las interrupciones EINT0, EINT1 y EINT2, depende del valor de n que se modifica en la rutina de atención del Systick y que la función NVIC_SetPriority() pone el valor de prioridad en los 5 bits más significativos del registro de prioridad correspondiente a la interrupción en cuestión.

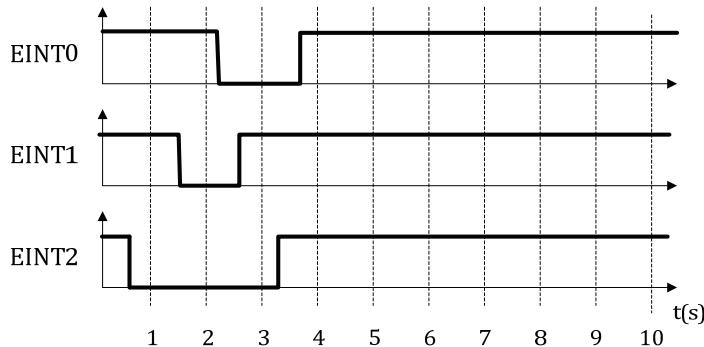


FIGURA 3.8. CRONOGRAMA DE SEÑALES DE INTERRUPCIÓN.

TABLA 3.6. ESTADOS DE LAS INTERRUPCIONES.

t (s)	Estado EINT0	Estado EINT1	Estado EINT2
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

2. Teniendo en cuenta que no se ha modificado el registro VTOR, que el contenido de la memoria a partir de la dirección 0x0000_0000 es el que se muestra a en la Figura 3.9 y que los números de vectores Systick, EINT0, EINT1 y EINT2 son 15, 34, 35 y 36 respectivamente, diga:
- En qué direcciones comienzan las subrutinas de atención de las interrupciones Systick, EINT0, EINT1 y EINT2.
 - En qué dirección comienza el programa principal (valor inicial del contador del programa tras un *Reset*).
 - Valor inicial del Puntero de Pila tras un *Reset*.

```

0x0000000000: 00 02 00 10
0x0000000004: 6D 01 00 00
0x0000000008: 71 01 00 00
0x000000000C: 73 01 00 00
0x0000000010: 75 01 00 00
0x0000000014: 77 01 00 00
0x0000000018: 79 01 00 00
0x000000001C: 00 00 00 00
0x0000000020: 00 00 00 00
0x0000000024: 00 00 00 00
0x0000000028: 00 00 00 00
0x000000002C: 7B 01 00 00
0x0000000030: 7D 01 00 00
0x0000000034: 00 00 00 00
0x0000000038: 7F 01 00 00
0x000000003C: 2B 03 00 00
0x0000000040: 83 01 00 00
0x0000000044: 83 01 00 00
0x0000000048: 83 01 00 00
0x000000004C: 83 01 00 00
0x0000000050: 83 01 00 00
0x0000000054: 83 01 00 00
0x0000000058: 83 01 00 00
0x000000005C: 83 01 00 00
0x0000000060: 83 01 00 00
0x0000000064: 83 01 00 00
0x0000000068: 83 01 00 00
0x000000006C: 83 01 00 00
0x0000000070: 83 01 00 00
0x0000000074: 83 01 00 00
0x0000000078: 83 01 00 00
0x000000007C: 83 01 00 00
0x0000000080: 83 01 00 00
0x0000000084: 83 01 00 00
0x0000000088: DD 03 00 00
0x000000008C: 1B 04 00 00
0x0000000090: 59 04 00 00
0x0000000094: 83 01 00 00

```

FIGURA 3.9. CONTENIDO DE LA TABLA DE VECTORES.

PROBLEMA 3.7.

Dado el siguiente programa:

```

#include <LPC17xx.H>
char a=1, b=1, c=1;
int n=0;

void SysTick_Handler (void)  {
    n++;
}

```

```
void inicio()
{
    a=1; b=1; c=1;
    LPC_PINCON->PINSEL4 |= 1 << (12*2);
    LPC_PINCON->PINSEL4 |= 1 << (11*2);
    LPC_PINCON->PINSEL4 |= 1 << (10*2);

    NVIC_SetPriorityGrouping(4);
    NVIC_SetPriority(EINT0_IRQn, 0x2);
    NVIC_SetPriority(EINT1_IRQn, 0x4);
    NVIC_SetPriority(EINT2_IRQn, 0x5);
    NVIC_SetPriority (SysTick_IRQn, 0);

    NVIC_EnableIRQ(EINT0_IRQn);
    NVIC_EnableIRQ(EINT1_IRQn);
    NVIC_EnableIRQ(EINT2_IRQn);

// SysTick->LOAD = SysTick->CALIB;
    SysTick->LOAD = 100000;
    SysTick->VAL = 0;
    SysTick->CTRL = 0x7;
}
void EINT0_IRQHandler() {
    LPC_SC->EXTINT |= (1 << 0);
    n=0;
    while(a) {
        if (n==200){
            a =0;
            n=0;
        }
    }
    a=1; }

void EINT1_IRQHandler() {
    LPC_SC->EXTINT |= (1 << 1);
    n=0;
    while(b) {
        if (n==100){
            b =0;
            n=0;
        }
    }
}
```

```

b=1; }

void EINT2_IRQHandler() {
    LPC_SC->EXTINT |= (1 << 2);
    n=0;
    while(c) {
        if (n==300){
            c =0;
            n=0;
        }
    }
    c=1;
}

main () {
    inicio();
    while(1); }

```

1. A partir del cronograma de las señales en las entradas EINT0, EINT1 y EINT2, mostrado en la Figura 3.10, rellene en la Tabla 3.7 el estado de cada una de dichas interrupciones (A-Activa, P-Pendiente o X- otro estado), para los instantes de tiempos mostrados en la primera columna. Tenga en cuenta que el fin de las rutinas de atención a las interrupciones EINT0, EINT1 y EINT2, depende del valor de n que se modifica en la rutina de atención del Systick y que la función NVIC_SetPriority() pone el valor de prioridad en los 5 bits más significativos del registro de prioridad correspondiente a la interrupción en cuestión.

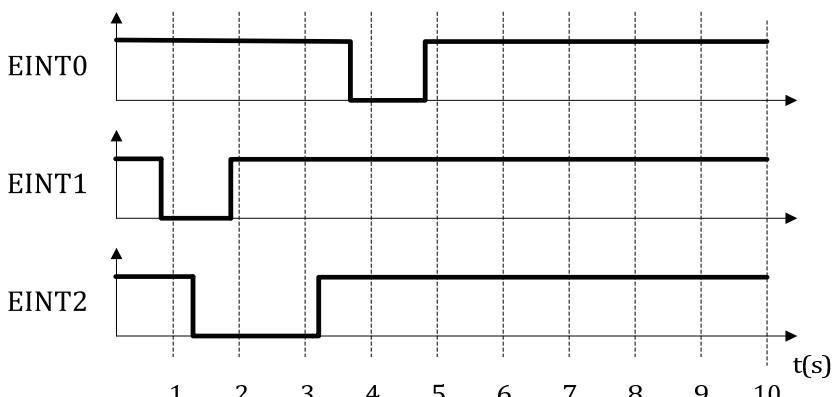


FIGURA 3.10. CRONOGRAMA DE SEÑALES DE INTERRUPCIÓN.

TABLA 3.7. ESTADOS DE LAS INTERRUPCIONES.

Estado EINT0	Estado EINT1	Estado EINT2
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

2. Teniendo en cuenta que no se ha modificado el registro VTOR, que el contenido de la memoria a partir de la dirección 0x0000_0000 es el mostrado en la Figura 3.11 y que los números de vectores Systick, EINT0, EINT1 y EINT2 son 15, 34, 35 y 36 respectivamente, diga:
- En qué direcciones comienzan las subrutinas de atención de las interrupciones Systick, EINT0, EINT1 y EINT2.
 - En qué dirección comienza el programa principal (valor inicial del contador del programa tras un *Reset*).
 - Valor inicial del Puntero de Pila tras un *Reset*.

```

: 0x0000000000: 68 02 00 10
: 0x0000000004: 6D 01 00 00
: 0x0000000008: 71 01 00 00
: 0x000000000C: 73 01 00 00
: 0x0000000010: 75 01 00 00
: 0x0000000014: 77 01 00 00
: 0x0000000018: 79 01 00 00
: 0x000000001C: 00 00 00 00
: 0x0000000020: 00 00 00 00
: 0x0000000024: 00 00 00 00
: 0x0000000028: 00 00 00 00
: 0x000000002C: 7B 01 00 00
: 0x0000000030: 7D 01 00 00
: 0x0000000034: 00 00 00 00
: 0x0000000038: 7F 01 00 00
: 0x000000003C: 2B 03 00 00
: 0x0000000040: 83 01 00 00
: 0x0000000044: 83 01 00 00
: 0x0000000048: 83 01 00 00
: 0x000000004C: 83 01 00 00
: 0x0000000050: 83 01 00 00
: 0x0000000054: 83 01 00 00
: 0x0000000058: 83 01 00 00
: 0x000000005C: 83 01 00 00
: 0x0000000060: 83 01 00 00
: 0x0000000064: 83 01 00 00
: 0x0000000068: 83 01 00 00
: 0x000000006C: 83 01 00 00
: 0x0000000070: 83 01 00 00
: 0x0000000074: 83 01 00 00
: 0x0000000078: 83 01 00 00
: 0x000000007C: 83 01 00 00
: 0x0000000080: 83 01 00 00
: 0x0000000084: 83 01 00 00
: 0x0000000088: CD 03 00 00
: 0x000000008C: 0B 04 00 00
: 0x0000000090: 49 04 00 00
: 0x0000000094: 83 01 00 00

```

FIGURA 3.11. CONTENIDO DE LA TABLA DE VECTORES.**PROBLEMA 3.8.**

1. A partir del programa de la Figura 3.12 y del cronograma de las señales en las entradas EINT0, EINT1 y EINT2, que se muestran en la Figura 3.13, complete dicho cronograma con los valores de a y b y rellene en la Tabla 3.8 el estado de cada una de dichas interrupciones (A-Activa, P-Pendiente o X- otro estado), para los instantes de tiempos mostrados en la primera columna.

Nota 1: Tenga en cuenta que la función NVIC_SetPriority() pone el valor de prioridad en los 5 bits más significativos del registro de prioridad correspondiente a la interrupción en cuestión y que las solicitudes de interrupciones externas son activas con el flanco de bajada.

Nota 2: La frecuencia del reloj del sistema es 100 MHZ.

```

01 #include <LPC17xx.H>
02 char a=1, b=1;
03 int n=0;
04
05 void SysTick_Handler (void) {    n++;
06 }
07
08 void inicio() {
09     LPC_PINCON->PINSEL4 |= 1 << (12*2);
10    LPC_PINCON->PINSEL4 |= 1 << (11*2);
11    LPC_PINCON->PINSEL4 |= 1 << (10*2);
12
13    NVIC_SetPriorityGrouping(3);
14    NVIC_SetPriority(EINT0_IRQn, 0x2);
15    NVIC_SetPriority(EINT1_IRQn, 0x4);
16    NVIC_SetPriority(EINT2_IRQn, 0x5);
17    NVIC_SetPriority (SysTick_IRQn, 0);
18
19    NVIC_EnableIRQ(EINT0_IRQn);
20    NVIC_EnableIRQ(EINT1_IRQn);
21    NVIC_EnableIRQ(EINT2_IRQn);
22 }
23
24 void EINT0_IRQHandler() {
25     LPC_SC->EXTINT |= (1 << 0);
26     SysTick->LOAD = 1000000;
27     SysTick->VAL = 0;
28     SysTick->CTRL = 0x7;
29     NVIC_DisableIRQ(EINT0_IRQn);
30 }
31
32 void EINT1_IRQHandler() {
33     LPC_SC->EXTINT |= (1 << 1);
34     n=0;
35     while(n<100) {    a =0; }
36     a=1;
37 }
38
39 void EINT2_IRQHandler() {
40     LPC_SC->EXTINT |= (1 << 2);
41     n=0;
42     while(n<200) {      b =0; }
43     b=1;
44 }
45
46 main () {
47     inicio();
48     while(1);
49 }
```

FIGURA 3.12. CÓDIGO ENSAMBLADOR.

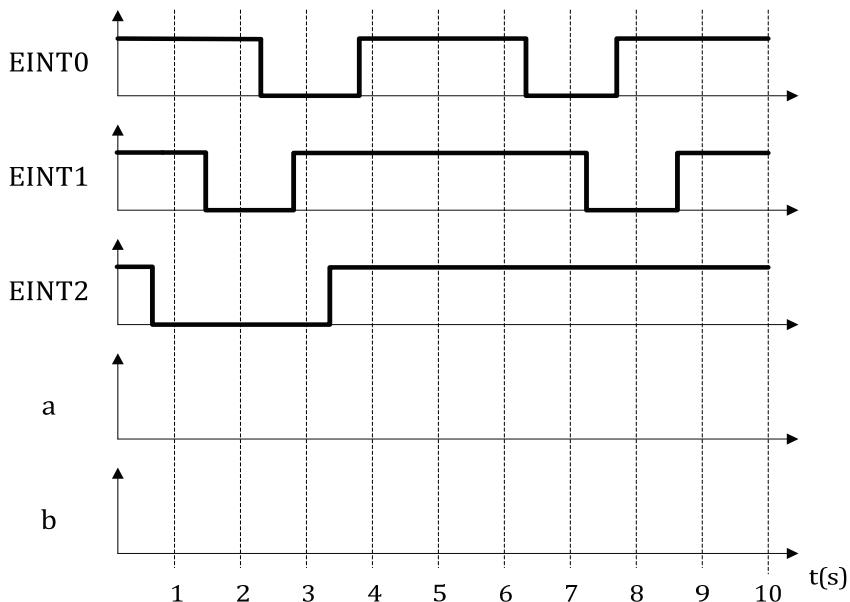


FIGURA 3.13. CRONOGRAMA DE ACTIVACIÓN DE LAS INTERRUPCIONES.

TABLA 3.8. ESTADOS DE LAS INTERRUPCIONES.

t (s)	Estado EINT0	Estado EINT1	Estado EINT2
0	X	X	X
1			
2			
3			
4			
5			
6			
7			
9			
10			

2. Teniendo en cuenta que no se ha modificado el registro VTOR, que los números de vectores Systick, EINT0, EINT1 y EINT2 son 15, 34, 35 y 36 respectivamente, que el programa principal (valor inicial del contador del programa tras un *Reset*) comienza en la dir 16Ch, que las subrutinas de atención de las interrupciones Systick, EINT0, EINT1 y EINT2 comienzan

en las direcciones 32Ah, 3B4h, 3F0h y 422h respectivamente y que el valor inicial del Puntero de Pila tras un *Reset* es 0x1000_0200, complete la Tabla 3.9 con la tabla de vectores indicando las direcciones de memoria que es obligatorio llenar y los valores que toman las mismas.

TABLA 3.9. TABLA DE VECTORES.

Dirección	Valor	Justificación

PROBLEMA 3.9.

Un sistema digital basado en un microprocesador de la familia Cortex M3 está configurado para trabajar con un conjunto de excepciones. Dispone de memoria de programa y datos permanentes (ROM), y memoria de datos de lectura y escritura (RAM), ocupando los rangos de direcciones mostrados en la Tabla 3.10:

TABLA 3.10. UBICACIÓN DE LA MEMORIA.

Memoria	Dir. inicio	Dir. Fin
ROM	0x0000_0000	0x0000_FFFF
RAM ESTATICA	0x0001_0000	0x0001_FFFF

Además de las excepciones de prioridad negativa, solamente van a poder aparecer la interrupción del Systick y las interrupciones hardware 2 y 3, cuyas ISR comienzan en las direcciones 1400H, 2C00H y 3800H respectivamente.

Se pide:

1. Calcule el tamaño necesario de la tabla de vectores, e indique en qué direcciones del sistema se podría ubicar la misma, teniendo en cuenta las recomendaciones de ARM para el offset de la tabla de vectores.
2. Complete la Tabla 3.11 con el contenido de la tabla de vectores correspondiente a las excepciones contempladas en el sistema (las indicadas en el enunciado), suponiendo que:

- la tabla de vectores ocupa 256 bytes y la ISR del *Reset* comienza justo a continuación;
 - la tabla de vectores se ubica en la dirección 0x0000_0000;
 - el puntero de pila se inicializa de manera que se aproveche al máximo la memoria RAM.

Nota: Si no se conoce alguna dirección para inicializar la tabla indique XX

TABLA 3.11. TABLA DE VECTORES.

3. Si se configura el Systick para que interrumpa periódicamente, indique razonadamente cuál de las siguientes asignaciones de prioridad (opción 1, 2 o 3) mostradas en la Tabla 3.12 es la adecuada para que siempre que solicite interrupción sea atendida sin quedar pendiente, e indique por qué las otras no lo son. Considere que el número de bits implementados por el fabricante para programar la prioridad en los registros son 3 bits y que se ha programado con un valor Priority Group =6.

TABLA 3.12. ASIGNACIONES DE PRIORIDAD.

	Systick priority (8 bits)	Int_2 priority (8 bits)	Int_3 priority (8 bits)
Opción 1	0x00	0x20	0x40
Opción 2	0x00	0xC0	0x80
Opción 3	0x80	0x00	0x20

4. Si se realiza la siguiente asignación de prioridades: Systick=0x00; Int_2=0x20; Int_3=0x40; indique razonadamente los valores que deben tener los registros PRIMASK, FAULTMASK y el margen de valores de BASEPRI para que no se atiendan las interrupciones externas 2 y 3.

PROBLEMA 3.10.

Un sistema digital basado en el LPC1768 está configurado para trabajar con interrupciones. En un determinado momento de la ejecución se produce una solicitud de interrupción del ADC que es atendida, y justo antes de ejecutar la primera instrucción de la ISR los valores de los registros y de una determinada zona de memoria son los que se muestran en la Figura 3.14 y Figura 3.15.

Register	Value
Core	
R0	0x01500000
R1	0x4002C000
R2	0xE000E000
R3	0xE000E000
R4	0x00000000
R5	0x10000008
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000790
R11	0x00000000
R12	0x10000048
R13 (SP)	0x10000248
R14 (LR)	0xFFFFFFF9
R15 (PC)	0x00000240
xPSR	0x61000026
N	0
Z	1
C	1
V	0
Q	0
T	1
IT	Disabled
ISR	38
Banked	

FIGURA 3.14. VALORES DE LOS REGISTROS.

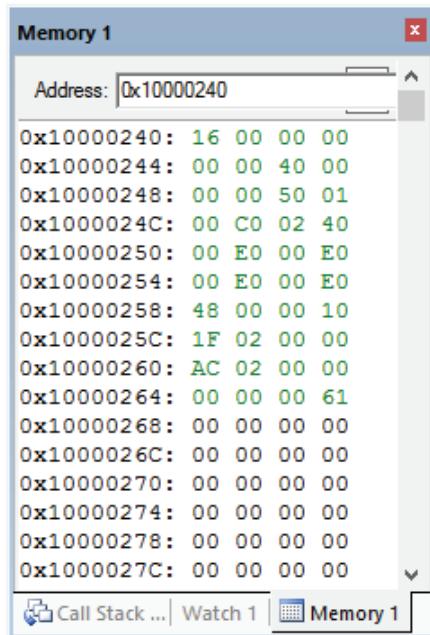


FIGURA 3.15. ZONA DE MEMORIA.

Responda justificadamente a las siguientes preguntas:

1. Sabiendo que el registro VTOR no se ha modificado desde el *Reset*, ¿qué dirección de la tabla de vectores se ha configurado, y con qué valor, para que se esté ejecutando dicha rutina?
2. ¿Cuál es el valor del registro SP justo antes de atenderse la solicitud de esta interrupción?
3. ¿Cuál es el valor del registro PC justo antes de atenderse la solicitud de esta interrupción?
4. Suponiendo que además del ADC pueden interrumpir el Systick, el Timer 0 y la entrada externa EINT2, y que se ha escrito el valor 0x5 en los bits 8:10 del registro AIRCR del NVIC, configure las prioridades de las excepciones anteriores para que:
 - Systick sea el más prioritario y pueda desalojar a todos los demás,
 - el ADC desaloje a EINT2 y Timer 0,
 - EINT2 sea más prioritaria pero no desaloje a Timer 0.

PROBLEMA 3.11.

Dado el siguiente código en lenguaje C para un LPC1768, suponiendo que los pines dedicados a las interrupciones implicadas han sido previamente configurados como tales, y activos por flanco descendente, y que el contenido parcial de la tabla de vectores es el mostrado en la Figura 3.16, se pide:

Nota: La función *delay_ms(Num)* introduce un retardo de *Num* milisegundos.

```
#include <LPC17xx.H>
char a=1, b=1;

void EINT1_IRQHandler() {
    LPC_SC->EXTINT |=(1<<1);
    NVIC_EnableIRQ(EINT3_IRQn);
    while(a==1);
    delay_ms(10);
    b=0;
}

void EINT2_IRQHandler() {
    LPC_SC->EXTINT |=(1<<2);
    while(b==1);
    delay_ms(10);
}

void EINT3_IRQHandler() {
    LPC_SC->EXTINT |=(1<<3);
    a=0;
    delay_ms(10);
}

main () {
    NVIC_SetPriorityGrouping(4);
    NVIC_SetPriority(EINT1_IRQn, 0x7);
    NVIC_SetPriority(EINT2_IRQn, 0x4);
    NVIC_SetPriority(EINT3_IRQn, 0x2);
    NVIC_EnableIRQ(EINT1_IRQn);
    NVIC_EnableIRQ(EINT2_IRQn);
    while(1);
}
```

0x00000008:	75 01 00 00
0x0000000C:	77 01 00 00
0x00000010:	79 01 00 00
0x00000014:	7B 01 00 00
0x00000018:	7D 01 00 00
0x0000001C:	00 00 00 00
0x00000020:	00 00 00 00
0x00000024:	00 00 00 00
0x00000028:	00 00 00 00
0x0000002C:	7F 01 00 00
0x00000030:	81 01 00 00
0x00000034:	00 00 00 00
0x00000038:	83 01 00 00
0x0000003C:	85 01 00 00
0x00000040:	87 01 00 00
0x00000044:	87 01 00 00
0x00000048:	87 01 00 00
0x0000004C:	87 01 00 00
0x00000050:	87 01 00 00
0x00000054:	87 01 00 00
0x00000058:	87 01 00 00
0x0000005C:	87 01 00 00
0x00000060:	87 01 00 00
0x00000064:	87 01 00 00
0x00000068:	87 01 00 00
0x0000006C:	87 01 00 00
0x00000070:	87 01 00 00
0x00000074:	87 01 00 00
0x00000078:	87 01 00 00
0x0000007C:	87 01 00 00
0x00000080:	87 01 00 00
0x00000084:	87 01 00 00
0x00000088:	87 01 00 00
0x0000008C:	D3 01 00 00
0x00000090:	FF 01 00 00
0x00000094:	19 02 00 00
0x00000098:	87 01 00 00
0x0000009C:	87 01 00 00
0x000000A0:	87 01 00 00

FIGURA 3.16. TABLA DE VECTORES (LITTLE ENDIAN).

1. Indique el contenido que se configurará en el campo de bits PRIGROUP del registro AIRCR.
 2. Especifique los bits de prioridad y subprioridad que quedarán destinados dentro de los registros IPR implicados.
 3. ¿A cuál de las tres fuentes de interrupción se le asignará más prioridad? ¿Por qué?
 4. Complete el cronograma de la Figura 3.17 indicando el estado de cada rutina de atención a la interrupción, ISR: A (activa), P (pendiente), X (no se está ejecutando).
- Nota:** Considere que las interrupciones externas son activas por flanco de bajada.

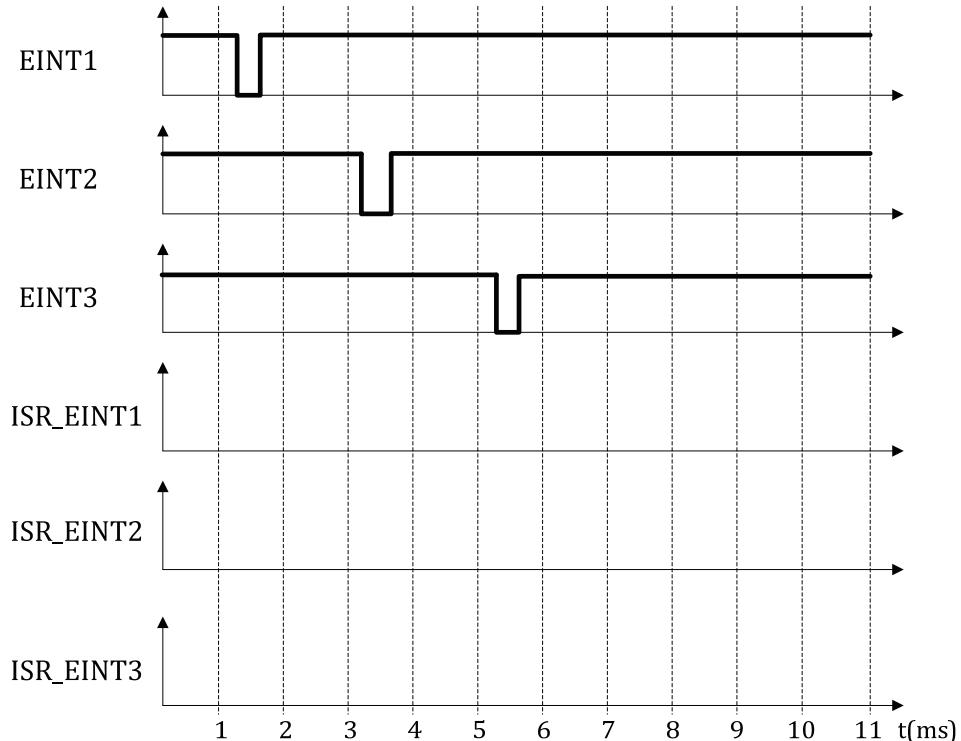


FIGURA 3.17. CRONOGRAMA DE SEÑALES.

5. Sabiendo que los números de vector para las interrupciones EINT1, EINT2 y EINT3 son 35, 36 y 37, respectivamente, ¿cuál será la dirección de comienzo de las respectivas ISR?

6. Conociendo que la ejecución del programa comienza en la dirección 0x16D y que inicialmente el valor de PSP=0x1000_1000 y MSP=0x1000_2000, rellene en la Tabla 3.13 las dos primeras entradas de la tabla de vectores:

TABLA 3.13. TABLA DE VECTORES.

Dirección	Contenido (Little endian)			

PROBLEMA 3.12.

En un sistema digital basado en el LPC1768, una de las posibles fuentes de interrupción es la del Systick, cuya ISR se suministra a continuación.

```

PUSH {R4, R2, LR}          // Linea 1
LDR R0,=0x10000000         // Linea 2
LDMDB R0!, {R4,R2}          // Linea 3
MOVT R4,# 0x01AC           // Linea 4
ADD R2, #0X0C               // Linea 5
STMIA R0, {R2}                // Linea 6
...

```

1. Complete la/s instrucción/es que faltan al final de la ISR para que el sistema continúe funcionando correctamente.
2. Complete la Tabla 3.14 determinando qué posiciones de memoria o registros se modifican cuando se ejecuta su rutina, y con qué valor, si justo antes de ejecutarse la primera instrucción de su rutina la memoria y registros se encuentran en la situación que se muestra en la Tabla 3.15.

TABLA 3.14. CRONOGRAMA DE SEÑALES.

Línea de código	Registro o Dir. memoria	Contenido (Hexadecimal)			
1					
2					
3					
4					
5					
6					

TABLA 3.15. VALORES INICIALES DE LA MEMORIA Y LOS REGISTROS.

Memoria		Registros	
Dirección	Contenido	Registro	Valor
0FFF FFF4	89 AB CD EF	R0	00 00 00 00
0FFF FFF8	AA 00 BB 11	R1	11 11 11 11
0FFF FFFC	CC 22 DD 33	R2	00 11 22 33
1000 0000	EE 44 FF 55	R3	44 55 66 77
1000 0004	66 77 88 99	R4	88 99 AA BB
1000 0008	00 11 23 45	R5	CC DD EE FF
1000 000C	67 89 AB CD	R6	CC 22 DD 33
.....		R7	EE 44 FF 55
1000 1FF4	66 77 88 99	R8	66 77 88 99
1000 1FF8	00 11 23 45	R9	00 11 23 45
1000 1FFC	67 89 AB CD	R10	AA AA AA AA
1000 2000	EF 01 23 45	R11	01 23 45 67
1000 2004	CC DD EE FF	R13	10 00 20 04
1000 2008	01 23 45 67	R14	FF FF FF F9
1000 200C	89 AB CD EF		
1000 2010	EF 01 23 45		
.....			
1000 6004	00 00 00 00		
1000 6008	00 00 00 00		
1000 600C	00 00 00 00		
1000 6010	00 00 00 00		
1000 6004	00 00 00 00		

PROBLEMA 3.13.

Un sistema digital basado en un microprocesador de la familia Cortex M-3 está configurado para trabajar con un conjunto de excepciones. Dispone de memoria de programa y datos permanentes (ROM), y memoria de datos de lectura y escritura (RAM), ocupando los rangos de direcciones indicados en la Tabla 3.16.

TABLA 3.16. DIRECCIONES DE INICIO Y FIN DE LA MEMORIA.

Memoria	Dir. inicio	Dir. Fin
ROM	0x0000_0000	0x0000_3FFF
RAM	0x1000_0000	0x1000_7FFF

Además de las excepciones de prioridad negativa, está contemplada la interrupción del Systick y las interrupciones hardware #0 y #1. Se conoce la dirección de inicio de algunas de las ISRs, según se indica en la Tabla 3.17.

TABLA 3.17. DIRECCIONES DE INICIO Y FIN DE LAS RUTINAS.

Rutina	Dirección inicio
ISR_NMI	0x0000_0800
ISR_HFAULT	0x0000_0A00
ISR_SYSTICK inicio fin	0x0000_0B00 0x0000_0B7F

Por otra parte, para realizar la asignación del inicio de las rutinas de atención, se ha considerado lo siguiente:

- Se planificará el sistema para extensiones futuras, considerando la posición y el tamaño de la tabla de vectores estándar (255 vectores de excepción).
- Las ISRs de las excepciones del sistema se ubicarán de manera consecutiva después de la tabla de vectores, ordenadas según su prioridad (de mayor a menor prioridad).

1. Con las premisas anteriores, calcule el tamaño en bytes de las ISRs indicadas en la Tabla 3.18.

TABLA 3.18. TAMAÑO DE LAS ISRs.

ISR	Tamaño (bytes)
Reset	
NMI	
HARD FAULT	
SYSTICK	

2. Complete en la Tabla 3.19 el contenido de las 12 primeras posiciones de la tabla de vectores.

TABLA 3.19. CONTENIDO DE LA TABLA DE VECTORES.

Dirección Tabla de vectores (hex)	Contenido (hex)				Descripción
	4N	4N+1	4N+2	4N+3	

Se configura el Systick para que interrumpa periódicamente cada 10ms para generar un reloj en tiempo real. Además, se desea que la interrupción hardware #1 sea más prioritaria que la #0, sin que la pueda desalojar. Si el número de bits de prioridad implementado es 3 y se ha configurado grupo de prioridad 6,

1. Indique razonadamente las prioridades que se deben asignar a cada interrupción (SYSTICK, Interrupción hardware 0 e Interrupción hardware 1) para que se cumpla la jerarquía indicada.
2. Se realiza la siguiente configuración relacionada con la gestión de prioridades de las interrupciones:
 - Priority Group=6
 - SYSTICK_priority=0x20
 - Int.hardware_0_priority=0xB0
 - Int.hardware_1_priority=0x85
 - PRIMASK=0
 - FAULTMASK=0
 - BASEPRI=0x90

El SYSTICK se ha configurado para que interrumpa cada 10ms, y la duración de la ISR es de aproximadamente 5ms. La duración de las ISRs de la interrupción hardware #0 (INT0) y de la #1(INT1) es de 20ms y 16ms respectivamente, y sus contenidos no tienen relación con el SYSTICK.

Complete el cronograma de la Figura 3.18 indicando el estado de cada rutina de atención a la interrupción, ISR: A (activa), P (pendiente), X (no se está ejecutando), teniendo en cuenta el tiempo de ejecución de las mismas. **Nota:** Considerre que las interrupciones externas son activas por flanco de bajada.

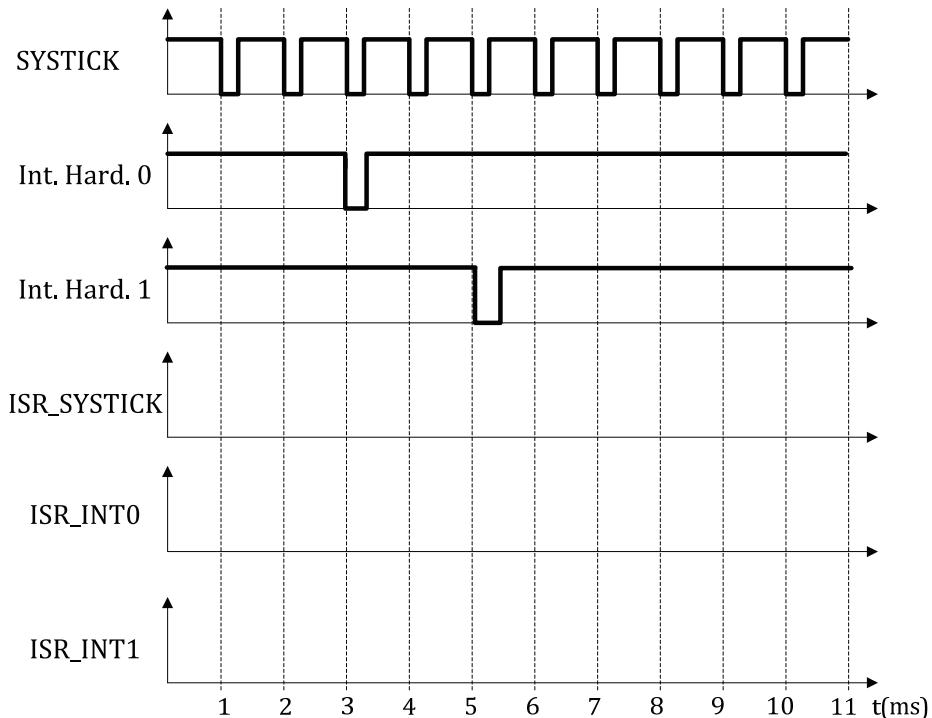


FIGURA 3.18. CRONOGRAMA DE ACTIVACIÓN DE LAS ISRs.

PROBLEMA 3.14.

Un sistema digital basado en un microprocesador de la familia Cortex M3 (LPC 1768) tiene habilitada la posibilidad de interrupción de: Systick, Timer0, EINT3, y ADC (Excepciones 15, 17, 37 y 38 respectivamente), además de las de prioridad negativa.

1. Sabiendo que no se han programado más ISR's que las de las excepciones enumeradas anteriormente, y que el compilador coloca el programa justo a continuación de la tabla de vectores necesaria, deduzca la dirección de comienzo del código.
2. Sabiendo que el puntero de pila (MSP) se desea que apunte inicialmente a la posición 0x1000_4000, que la inicialización de todo el sistema y el código de la aplicación (no incluye las ISR) ocupan 16Kbytes y el código de cada ISR 1Kbytes, complete la Tabla 3.20 con todas las entradas de la tabla

de vectores necesarias para cumplir con lo enunciado. Asuma que el orden de las ISR en memoria viene dado por su número de excepción.

TABLA 3.20. CONTENIDO DE LA TABLA DE VECTORES.

3. Sabiendo que el comportamiento de las excepciones debe ser el siguiente:

 - la interrupción EINT3 posee un valor de prioridad 0x30 (codificada en 8 bits), es la más prioritaria y debe ser capaz de desalojar a todas excepto a Systick,
 - Systick debe ser capaz de desalojar a Timer0 y ADC,
 - Timer0 debe ser capaz de desalojar a ADC y
 - el grupo de prioridad es 5,

complete la Tabla 3.21 con los valores de prioridad de todas ellas.

TABLA 3.21. PRIORIDADES DE LAS INTERRUPCIONES.

Interrupción	Prioridad binario 5 bits	Prioridad Hex
EINT3		
SysTick		
Timer0		
ADC		

4. Indique razonadamente qué ocurriría si el registro BASEPRI contiene un valor 0x38.

5. Si se produce la interrupción correspondiente a Timer 0, explique (de forma general, sin nombres ni valores concretos) qué debería hacer para saber cuál de las posibles fuentes de interrupción es la que la originó.

PROBLEMA 3.15.

Un sistema digital basado en un microcontrolador LPC1768 tiene habilitada la posibilidad de interrupción de: Systick, Timer1, EINT2 y CAN, además de las de prioridad negativa.

1. Sabiendo que en previsión de futuras ampliaciones del sistema se quiere dejar libre todo el espacio de la tabla de vectores de este microcontrolador, y que el compilador ubica el programa justo a continuación de la tabla de vectores necesaria, deduzca la dirección de comienzo del código.
 2. Sabiendo que el puntero de pila (MSP) se desea que apunte inicialmente a la posición 0x1000_4000, que la inicialización de todo el sistema y el código de la aplicación (no incluye las ISR) van ubicados a partir del primer kilobyte de memoria y ocupan 8Kbytes y el código de cada ISR ocupa 2Kbytes, complete en la Tabla 3.22 todas las entradas de la tabla de vectores necesarias para cumplir con lo enunciado. Asuma que el orden de las ISR en memoria viene dado por su número de excepción.

Tabla 3.22. Contenido de la tabla de vectores.

3. Sabiendo que el comportamiento de las excepciones debe ser indicado a continuación, complete la Tabla 3.23 con la inicialización de los valores de prioridad de todas ellas:

- la interrupción Timer1 posee un valor de prioridad 0x40 (codificada en 8 bits), es la más prioritaria y desaloja a cualquier otra,
- EINT2 debe ser capaz de desalojar a SysTick.
- EINT2 y CAN no se desalojan entre sí, pero en caso de que las dos estén pendientes primero se atiende CAN, y
- el grupo de prioridad es 4.

TABLA 3.23. CONTENIDO DE LA TABLA DE VECTORES.

Interrupción	Prioridad binario 5 bits	Prioridad Hex
EINT2		
SysTick		
Timer1		
CAN		

PROBLEMA 3.16.

Un sistema digital basado en un microprocesador de la familia Cortex M-3 está configurado para trabajar con las excepciones de prioridad negativa, la interrupción del SysTick y las excepciones externas #0, #1 y #2 (vectores 16, 17 y 18 respectivamente). Se realiza la siguiente configuración relacionada con la gestión de prioridades de las interrupciones:

- Número de bits para prioridad: 4
- Priority Group=5
- PRIMASK=0
- FAULTMASK=0
- BASEPRI=0x80
- SYSTICK_priority=0x10
- Ext_Int_0_priority=0x20
- Ext_Int_1_priority=0x80
- Ext_Int_2_priority=0x40

El SYSTICK se ha configurado para que interrumpe cada 10ms y la duración de la ISR es de aproximadamente 4ms (uno de los usos del SYSTICK es el diseño de un reloj en tiempo real). La duración aproximada de las ISRs de las excepciones externas #0, #1 y #2 (INT0, INT1 e INT2 en la gráfica) son 18ms, 20ms y 10ms respectivamente, y sus contenidos no tienen relación con el SYSTICK.

1. Complete el cronograma de la Figura 3.19 indicando el estado de cada rutina de atención a la interrupción ISR: A (activa), P (pendiente), X (no se está eje-

cutando), teniendo en cuenta el tiempo de ejecución de las mismas. Nota: Considere que todas las interrupciones son activas por flanco de bajada.

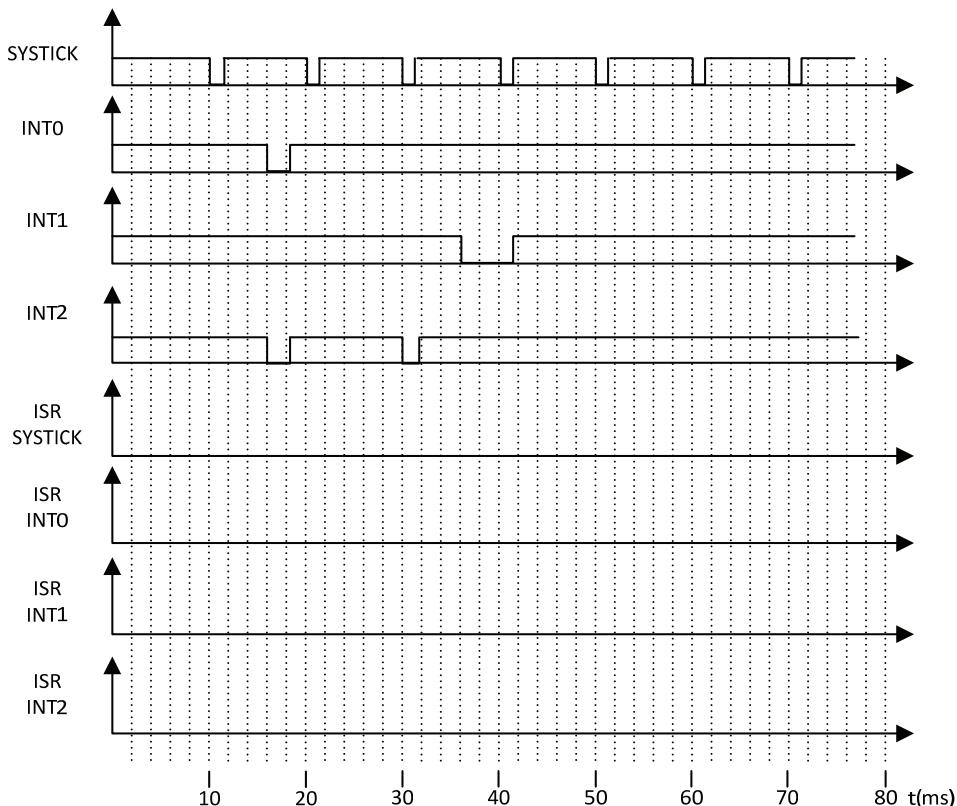


FIGURA 3.19. CRONOGRAMA DE ACTIVACIÓN DE LAS ISRs.

2. Se observa que el reloj en tiempo real diseñado a partir del SYSTICK sufre desfases. Proponga una solución para evitarlos sin modificar el valor numérico de prioridad asignado a cada fuente de interrupción.
3. Indique cómo configuraría los registros especiales del sistema para evitar que las cuatro fuentes de interrupción analizadas anteriormente puedan interrumpir (si hay varias soluciones indíquelo).
4. Si el tamaño de la tabla de vectores es el mínimo posible de acuerdo con las excepciones contempladas en el sistema, y asumiendo que la tabla se localiza a partir de la dirección 0x0, indique cuál sería la primera dirección de memoria disponible para almacenar otro tipo de información (código, datos, ...).

Se sabe que la distribución de la memoria del sistema es la indicada en la Tabla 3.24. Ubicación de la memoria.y que se ha decidido dedicar 1K bytes a la tabla de vectores, ubicada a partir de la dirección 0x0. La rutina de inicialización del sistema (ISR del *Reset*) ocupa 3kbytes, y el resto de ISRs ocupan 2kBytes cada una. Las ISRs se ubican a partir de la tabla de vectores de manera consecutiva, en orden creciente según su número de vector.

TABLA 3.24. UBICACIÓN DE LA MEMORIA.

Memoria	Dir. inicio	Dir. Fin
ROM	0x0000_0000	0x0000_1FFF
RAM	0x2000_0000	0x2000_3FFF

5. Complete en la Tabla 3.25 el contenido de las siguientes direcciones de la tabla de vectores.

TABLA 3.25. CONTENIDO DE LA TABLA DE VECTORES.

Dirección Tabla de vectores (hex)	Contenido (hex)				Descripción
	4N	4N+1	4N+2	4N+3	
0x0					
0x4					
0x8					
0x3C					

PROBLEMA 3.17.

Un sistema basado en LPC17xx maneja cuatro interrupciones. En la depuración del código se ha colocado un punto de ruptura en el inicio de la función *main*. En la Figura 3.20 se muestra el estado de parte de los recursos del microcontrolador en esa situación:

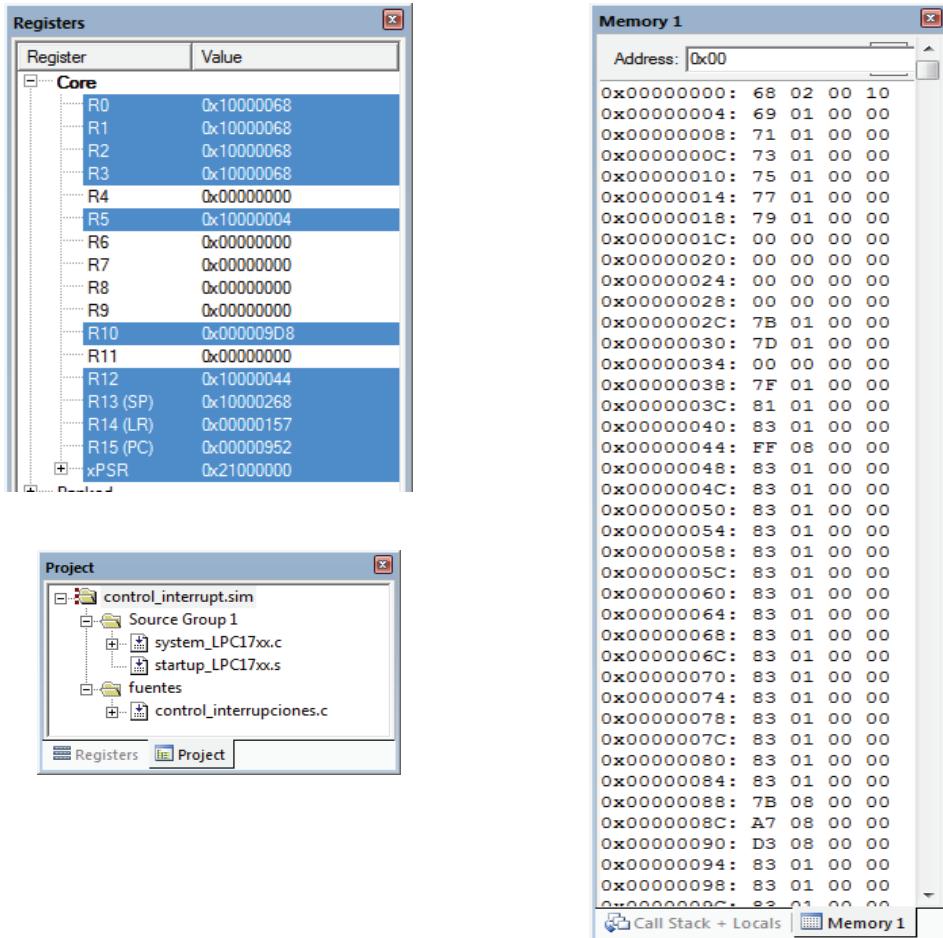


FIGURA 3.20. ESTADO DEL MICROCONTROLADOR.

1. Ayudado por la tabla de vectores, determine cuánto ocupa el código ejecutado desde el *Reset* hasta llegar a este punto de ruptura, y a qué es debido (inicio del *main*).
2. En la Figura 3.21 se muestra la situación del procesador: a) antes de ejecutar la primera sentencia de la función inicio, b) antes de ejecutar la última sentencia de la función inicio, y c) la situación antes de la primera instrucción de vuelta a la función *main*, obtenidas mediante la inserción de puntos de ruptura en dichas sentencias. Determine el tamaño en bytes de la función *inicio* y si ha guardado algo en pila al llamar a la función o dentro de ella.

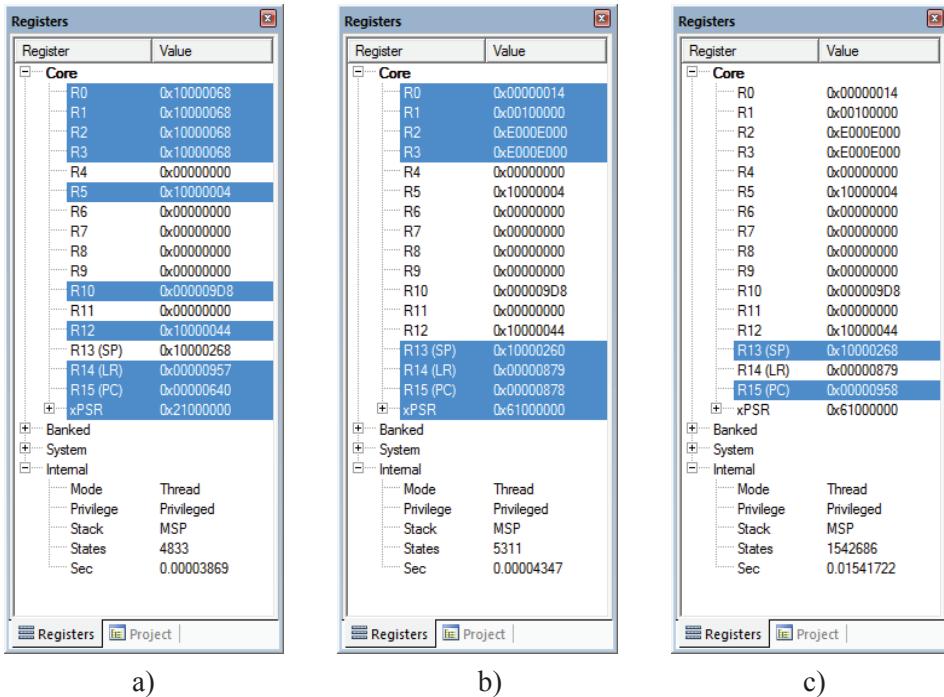


FIGURA 3.21. SITUACIÓN DEL PROCESADOR: a) ANTES DE EJECUTAR LA PRIMERA SENTENCIA DE LA FUNCIÓN INICIO, b) ANTES DE EJECUTAR LA ÚLTIMA SENTENCIA DE LA FUNCIÓN INICIO, Y c) LA SITUACIÓN ANTES DE LA PRIMERA INSTRUCCIÓN DE VUELTA A LA FUNCIÓN MAIN.

3. Si se produjeran cualquiera de las cuatro interrupciones sin haberse producido ninguna anteriormente y con un punto de ruptura en la primera instrucción de sus ISR la situación del micro sería la mostrada en la Figura 3.22.

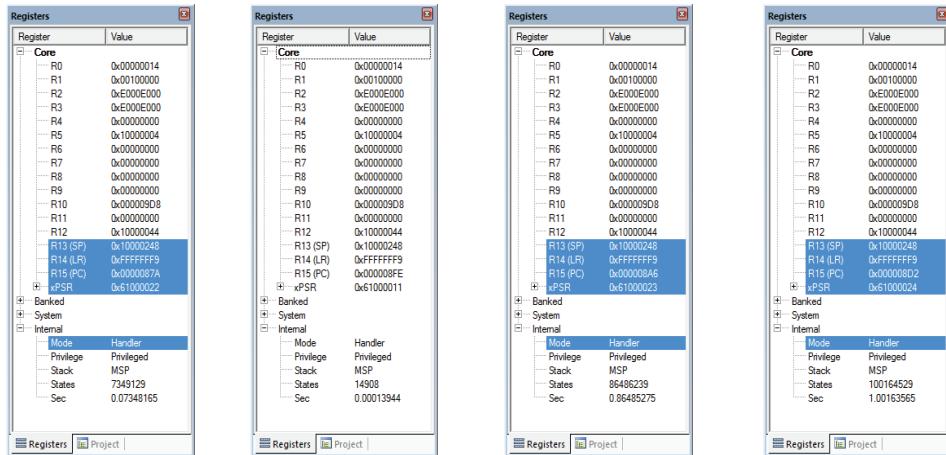


FIGURA 3.22. SITUACIÓN DESPUÉS DE PRODUCIRSE UNA INTERRUPCIÓN.

Determine de qué número de excepción se trata, si en algún caso se ha hecho uso de pila antes de llegar la interrupción, y determine lo que se ha guardado en pila por las ISR.

4. En la Figura 3.23 se muestra una captura del programa detenido cuando se ha terminado de ejecutar una de las rutinas (justo antes de recuperar el contexto anterior a la aparición de la interrupción). Determine su tamaño y el código de retorno de excepción que se cargará en el PC.

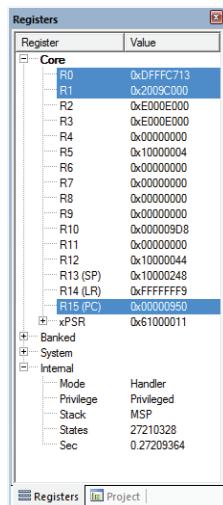


FIGURA 3.23. TRAS LA EJECUCIÓN DE UNA DE LAS RUTINAS.

5. Asumiendo que parte del código en “c” del sistema anterior es el siguiente (donde se han definido las fuentes de interrupción con nuevos nombres al principio del programa, llamándolas como: Interrupcion1, Interrupcion2 Interrupcion3 e Interrupcion4):

```
NVIC_SetPriorityGrouping(3);
NVIC_SetPriority(Interrupcion1 IRQn, 0x1);
NVIC_SetPriority(Interrupcion2 IRQn, 0x2);
NVIC_SetPriority(Interrupcion3 IRQn, 0x3);
```

- Indique si las 3 interrupciones indicadas en el cuadro anterior se expulsarían entre ellas en caso de estarse ejecutando y llegar otra.
- Indique qué nivel de prioridad le asignaría a la *Interrupcion4* para que todas pudieran desalojarla.

```
NVIC_SetPriorityGrouping(3);
NVIC_SetPriority(Interrupcion1 IRQn, 0x1);
NVIC_SetPriority(Interrupcion2 IRQn, 0x2);
NVIC_SetPriority(Interrupcion3 IRQn, 0x3);
NVIC_SetPriority(Interrupcion4 IRQn, 0x_____);
```

- Indique cómo conseguiría que la *Interrupcion4* desalojase a todas las demás, manteniéndose entre ellas el comportamiento de las restantes.

PROBLEMA 3.18.

A partir del programa y del cronograma de las señales en las entradas EINT0 y EINT1, que se muestran en la Figura 3.24, responda justificadamente a las siguientes preguntas:

Nota 1. Tenga en cuenta que la función NVIC_SetPriority() pone el valor de prioridad en los 5 bits más significativos del registro de prioridad correspondiente a la interrupción en cuestión y que las solicitudes de interrupciones externas son activas con el flanco de bajada.

Nota 2. La frecuencia del reloj del sistema es 100 Mhz.

```
#include <LPC17xx.H>
uint8_t a=0, b=0;
uint32_t n=0, m=0;
```

```
void Config() {
    LPC_PINCON->PINSEL4 |= 1 << (11*2);
    LPC_PINCON->PINSEL4 |= 1 << (10*2);

    NVIC_SetPriorityGrouping(4);
    NVIC_SetPriority(SysTick_IRQn, 0x0);
    NVIC_SetPriority(EINT0_IRQn, 0x4);
    NVIC_SetPriority(EINT1_IRQn, 0x8);

    SysTick->LOAD = 2000000;
    SysTick->VAL = 0;
    SysTick->CTRL = 0x7;

    NVIC_EnableIRQ(EINT0_IRQn);
    NVIC_EnableIRQ(EINT1_IRQn);
    NVIC_EnableIRQ(SysTick_IRQn);
}

void EINT0_IRQHandler() {
    LPC_SC->EXTINT |= (1);
    m=0;
    while(m<50)
        if(b) {a=1;}
    a=0;
}

void EINT1_IRQHandler() {
    LPC_SC->EXTINT |= (1 << 1);
    n=0;
    while(n<150){b=1;}
    b=0;
}

void SysTick_Handler (void) {
n++;
m++;
}

int main () {
    Config();
    while(1);
}
```

1. ¿Puede la interrupción EINT0 desalojar a la EINT1?
2. Calcule el periodo de interrupción el SysTick.
3. Complete el cronograma de la Figura 3.24 con los valores de las variables a y b y represente el estado de cada una de dichas interrupciones (A-Activa, P-Pendiente o X- otro estado).
4. ¿Qué hubiese ocurrido si se configura el grupo de prioridad a valor 6?

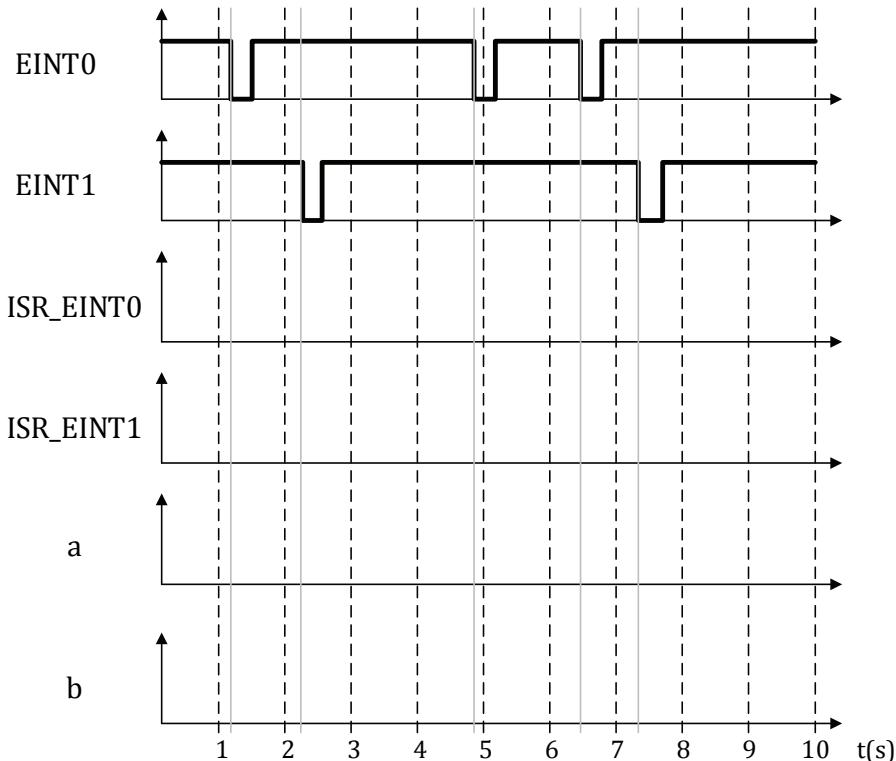
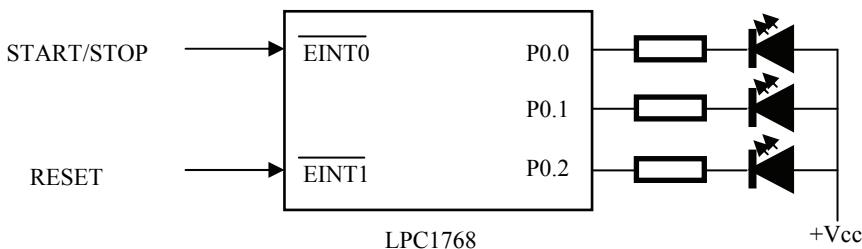


FIGURA 3.24. TRAS LA EJECUCIÓN DE UNA DE LAS RUTINAS.

PROBLEMA 3.19.

Se ha diseñado un sistema digital basado en el microcontrolador LPC1768 funcionando 100 Mhz al que se conectan dos pulsadores activos a nivel bajo y tres diodos LED's como se representa en la Figura 3.25. Los dos pulsadores deben funcionar provocando una interrupción al ser pulsados. Recuerde que las entradas de interrupción se encuentran en el puerto 2.

**FIGURA 3.25. SISTEMA DIGITAL BASADO EN EL LPC1768.**

El funcionamiento del sistema es el siguiente:

- Al iniciar el sistema, comienza en un estado de reposo en el que todos los LED's están apagados.
- Al presionar la primera vez el pulsador de **Start/Stop**, con el flanco de bajada, los LED's comienzan a cambiar de estado con una cadencia determinada para cada LED.
- El pulsador de **Reset**, con el flanco de bajada sitúa al sistema en el estado de reposo inicial.

Para implementar este sistema se hace uso del temporizador Systick, el cual se configura para que provoque una interrupción **cada milisegundo**.

Se pide:

1. Indique en la Tabla 3.26 el valor en hexadecimal al que hay que configurar el registro STRELOAD del Systick para que funcione según las especificaciones.

TABLA 3.26. VALOR DEL REGISTRO STRELOAD.

Registro	Valor (hexadecimal)
STRELOAD	

2. Complete la siguiente Tabla 3.27 con los registros que habría que configurar y con qué valores, para definir el funcionamiento de los terminales empleados y comiencen los LEDs apagados. Indique solamente los bits que sería necesario modificar, dejando en blanco los restantes. No es necesario configurar los registros *PINMODE_ODx*.

TABLA 3.27. CONFIGURACIÓN DE LOS REGISTROS.

Bit Registro \	3 1	2 8	2 4	2 0	1 6	1 2	8	4	0
Bit Registro									

3. Sabiendo que la interrupción más prioritaria es la del pulsador de **Reset**, y que debe desalojar a cualquiera de las otras dos interrupciones hardware, que la siguiente en prioridad es la interrupción del Systick, que la interrupción del pulsador **Start/Stop** no se ve desalojada por la interrupción del Systick, y que el grupo de prioridad se ha configurado a valor 6, complete la Tabla 3.28 indicando la prioridad en binario que se debería asignar a cada interrupción.

TABLA 3.28. CONFIGURACIÓN DE LAS PRIORIDADES DE LAS INTERRUPCIONES.

Interrupción	Valor 8 bits (binario)						

5. La rutina de atención a la interrupción del Systick es la mostrada a continuación. A partir de su análisis, y sabiendo que la interrupción del Systick se produce en los tiempos $t=1\text{ms}$, 2ms , 3ms , etc., represente en el cronograma de la Figura 3.26 el nivel lógico presente en los terminales de salida del sistema. Tenga en cuenta que las interrupciones son activas por flanco de bajada, que la ISR del pulsador de **Reset** deshabilita el Systick, y las ISR tardan en ejecutar se menos de 10 us.

```

volatile uint32_t cuenta;

void SysTick_Handler (void)  {

    cuenta++;
    if ((cuenta%2)==0)
        LPC_GPIO0->FIOPIN ^= (1<<0);
    if ((cuenta%4==0))
        LPC_GPIO0->FIOPIN ^= (1<<1);
    if ((cuenta%8)==0)
        LPC_GPIO0->FIOPIN ^= (1<<2);
}

```

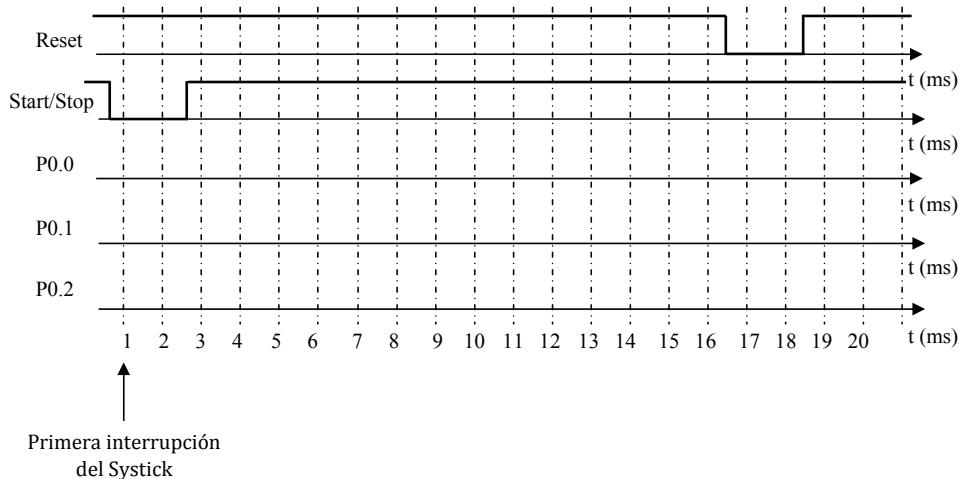


FIGURA 3.26. CRONOGRAMA DE ACTIVACIÓN DE INTERRUPCIONES.

PROBLEMA 3.20.

Un sistema digital basado en un microprocesador LPC 17XX está configurado para trabajar con las excepciones de prioridad negativa, y las excepciones externas, #1, #2 y #18 (vectores 17, 18 y 34 respectivamente). Se ha configurado el campo Priority Group con valor 2, y se ha realizado la siguiente configuración relacionada con la gestión de prioridades de las interrupciones:

- Ext_Int_1_priority =0x79
- Ext_Int_2_priority =0x82
- Ext_Int_18_priority =0x77

1. Complete la Tabla 3.29 indicando razonadamente con qué valor de prioridad quedarán programadas las interrupciones de prioridad positiva.

TABLA 3.29. PRIORIDAD DE LAS EXCEPCIONES.

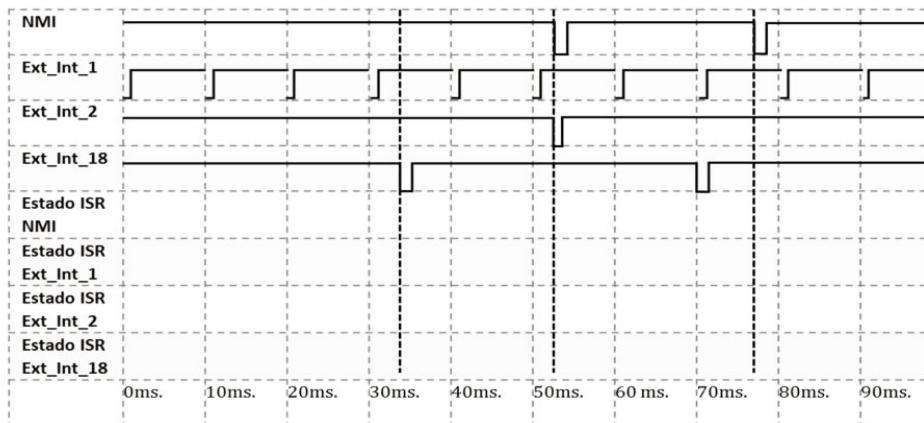
Excepción	Prioridad

2. De entre las excepciones de prioridad positiva, indique en la Tabla 3.30 cuáles pueden desalojar a las otras.

TABLA 3.30. DESALOJO ENTRE EXCEPCIONES.

Excepción	Excepción desalojada

3. Si se configura BASEPRI con valor 0x75, y con los valores de prioridad indicados anteriormente, indique, razonadamente, todas las fuentes de excepción e interrupción cuyas solicitudes serán atendidas.
4. En la situación descrita en el apartado 3 aparecen excepciones (activas en flanco descendente) en una secuencia tal y como se muestra en la Figura 3.27. Si las funciones de las ISR no están interrelacionadas y sus duraciones son 18ms, 15ms, 3 ms y 10ms, para NMI, Ext_Int_1, Ext_Int_2 y Ext_Int_18 respectivamente, represente en las cuatro últimas filas del cuadro el estado de cada rutina de atención a la interrupción ISR: A (activa), P (pendiente), X (no se está ejecutando), teniendo en cuenta el tiempo de ejecución de las mismas.

**FIGURA 3.27. CRONOGRAMA DE ACTIVACIÓN DE INTERRUPCIONES.**

5. Complete la Tabla 3.31 indicando en qué posiciones de memoria están almacenadas las direcciones de comienzo de las ISR de las excepciones de prioridad negativa y de las de prioridad positiva aludidas en el enunciado.

TABLA 3.31. COMIENZO DE LAS ISR.

Excepción	Dirección de memoria

6. Sabiendo que el compilador coloca el código después del último vector necesario, indique en qué dirección estará el comienzo del mismo.

PROBLEMA 3.21.

Un sistema digital basado en el Cortex-M3 se ha configurado para que pueda trabajar con un conjunto de excepciones e interrupciones, habiéndose dejado el registro VTOR con su configuración por defecto. El sistema dispone de memoria ROM y RAM, con la distribución indicada en la Tabla 3.32.

TABLA 3.32. DISTRIBUCIÓN DE MEMORIA.

Memoria	Dir. inicio	Dir. Fin
ROM	0x0000_0000	0x0000_3FFF
RAM	0x0000_4000	0x0000_5FFF

- El contenido de las primeras direcciones del sistema es el que se muestra en la Tabla 3.33. Indique sobre la misma tabla a qué hace referencia el contenido de cada posición de memoria.

TABLA 3.33. CONTENIDO DE LAS PRIMERAS POSICIONES DE MEMORIA.

Dirección (hex)	Contenido de memoria				Descripción
	4N	4N+1	4N+2	4N+3	
0000_0000	00	40	00	00	
0000_0004	01	02	00	00	
0000_0008	41	04	00	00	
0000_000C	01	05	00	00	
0000_003C	21	06	00	00	
0000_0050	01	0C	00	00	

Se sabe que la rutina de atención del *Reset* se ha ubicado a partir de la tabla de vectores del sistema, y el resto de rutinas de atención se han ubicado a continuación, en orden creciente según el número de vector.

- Calcule el tamaño de las rutinas correspondientes a las excepciones de *Reset* y *NMI*.
- Calcule el tamaño de memoria que se ha reservado para la tabla de vectores.
- Analice si cuando se produce una excepción se guarda adecuadamente el contexto en pila. En caso negativo proponga una solución para que se haga correctamente.

Se realiza la siguiente configuración relacionada con la gestión de prioridades de las interrupciones:

- Número de bits para prioridad: 4

- Priority Group=5
- PRIMASK=1
- FAULTMASK=0
- BASEPRI=0x80
- SYSTICK_priority=0x10
- Ext_Int_4_priority=0x80

El SYSTICK se ha configurado para que interrumpa cada 10ms y la duración de la ISR es de aproximadamente 4ms. La duración aproximada de la ISR de la excepción externa #4 es de 20ms. Las excepciones NMI y HARDFAULT duran 10ms y 12ms respectivamente, y sus contenidos no tienen relación con el SYSTICK.

5. Complete el cronograma de la Figura 3.28 indicando el estado de cada rutina de atención a la interrupción ISR: A (activa), P (pendiente), X (activa, pero no se está ejecutando), teniendo en cuenta el tiempo de ejecución de las mismas. Nota: Considere que todas las interrupciones son activas por flanco de bajada.

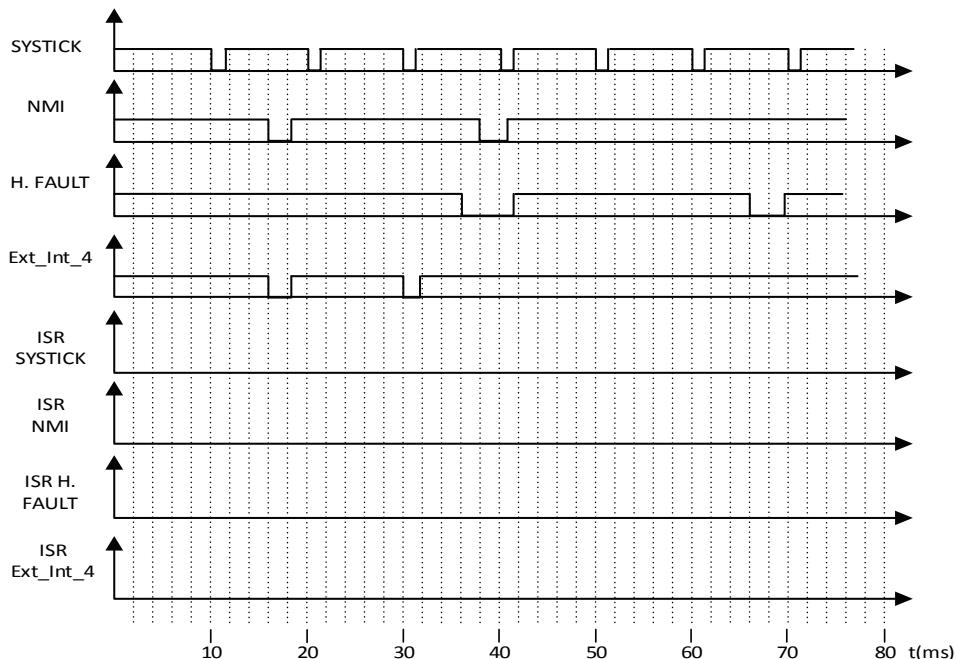


FIGURA 3.28. CRONOGRAMA DE ACTIVACIÓN DE INTERRUPCIONES.

PROBLEMA 3.22.

Un sistema digital basado en el Cortex-M3 reserva 1 Kbyte para la tabla de vectores (VT), y en ningún momento configura offset para la misma. Dispone de 2 bloques de memoria uno para ROM y otro para RAM de 512Kbytes y 32Kbytes respectivamente, correspondiendo al de RAM la dirección base 0x1000_0000. Del sistema se sabe, además, que la ISR de *RESET*, NMI y HARDFault ocupan 2Kbytes cada una y que la ISR de las otras 2 interrupciones permitidas (Timer2 y EINT0, excepciones 19 y 34) ocupan 128 bytes cada una.

1. Sabiendo que el compilador coloca el código inmediatamente a continuación de la VT sin dejar espacios vacíos, siendo colocados los códigos de las ISR por orden de número de vector, complete en la Tabla 3.34 las entradas de la VT necesarias, que puedan ser obtenidas a partir del enunciado.

TABLA 3.34. TABLA DE VECTORES

2. Si se realiza la configuración mostrada en la Tabla 3.35 relacionada con la gestión de prioridades de las interrupciones:

Tabla 3.35. Configuración de interrupciones.

<ul style="list-style-type: none"> • Número de bits para prioridad: 5 • FAULTMASK=0 • PRIMASK=0 • BASEPRI=0x90 	<ul style="list-style-type: none"> • Priority Group=6 • Timer2_priority= 0x20 • EINT0_priority= 0x10
--	---

Complete el cronograma de la Figura 3.29 indicando el estado de cada rutina de atención a la interrupción ISR: A (activa), P (pendiente), X (activa, pero no se

está ejecutando), teniendo en cuenta el tiempo de ejecución de las mismas. **Nota:** Considere que todas las interrupciones son activas por flanco de bajada, y que la duración de ejecución de las ISR es de 10 ms para HardFault, 7 ms para el Timer 2 y 4 ms para EINT0.

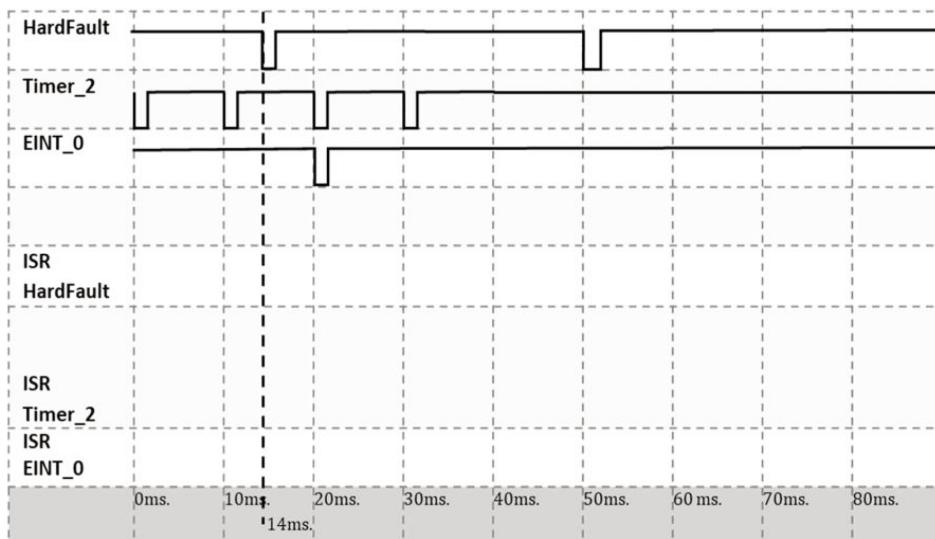


FIGURA 3.29. CRONOGRAMA DE ACTIVACIÓN DE INTERRUPCIONES.

PROBLEMA 3.23.

En un sistema digital basado en el Cortex-M3 se reserva 1 Kbyte para la tabla de vectores, y en ningún momento se modifica el contenido del registro VTOR. Dispone de 32 Kbytes de memoria ROM y 128 Kbytes de memoria RAM ubicada a partir de la dirección 0x1000_0000. Del sistema se sabe también que la rutina de interrupción del *RESET* ocupa 4 Kbytes, la de la NMI 1 Kbyte, la del Hard Fault 1Kbyte y la del SysTick 128 Bytes. Además, se han utilizado las interrupciones EINT0 y EINT1, cuyas rutinas de interrupción ocupan 1 Kbyte cada una.

1. Sabiendo que el compilador coloca el código inmediatamente a continuación de la tabla de vectores sin dejar espacios vacíos, siendo colocados los códigos de las rutinas de interrupción por orden de número de vector, complete en la Tabla 3.36 las entradas de la tabla de vectores necesarias, que puedan ser obtenidas a partir del enunciado.

TABLA 3.36. TABLA DE VECTORES.

Dirección (hex)	Contenido de memoria				Descripción
	4N	4N+	4N+	4N+3	

2. Si se realiza la configuración mostrada en la Tabla 3.37 relacionada con la gestión de prioridades de las interrupciones:

TABLA 3.37. CONFIGURACIÓN DE INTERRUPCIONES.

• Número de bits para prioridad: 5	• Priority Group=5
• FAULTMASK=0	• SysTick_priority= 0x01
• PRIMASK=0	• EINT0_priority= 0x50
• BASEPRI=0x60	• EINT1_priority= 0x40

Complete el cronograma de la Figura 3.30 indicando el estado de cada rutina de atención a la interrupción ISR: A (activa), P (pendiente), X (activa, pero no se está ejecutando), teniendo en cuenta el tiempo de ejecución de las mismas. **Nota:** Considere que todas las interrupciones son activas por flanko de bajada, y que la duración de ejecución de las rutinas de interrupción es de 10 ms para NMI, 4 ms para el Systick, 7 ms para EINT0 y 10 ms para EINT1.

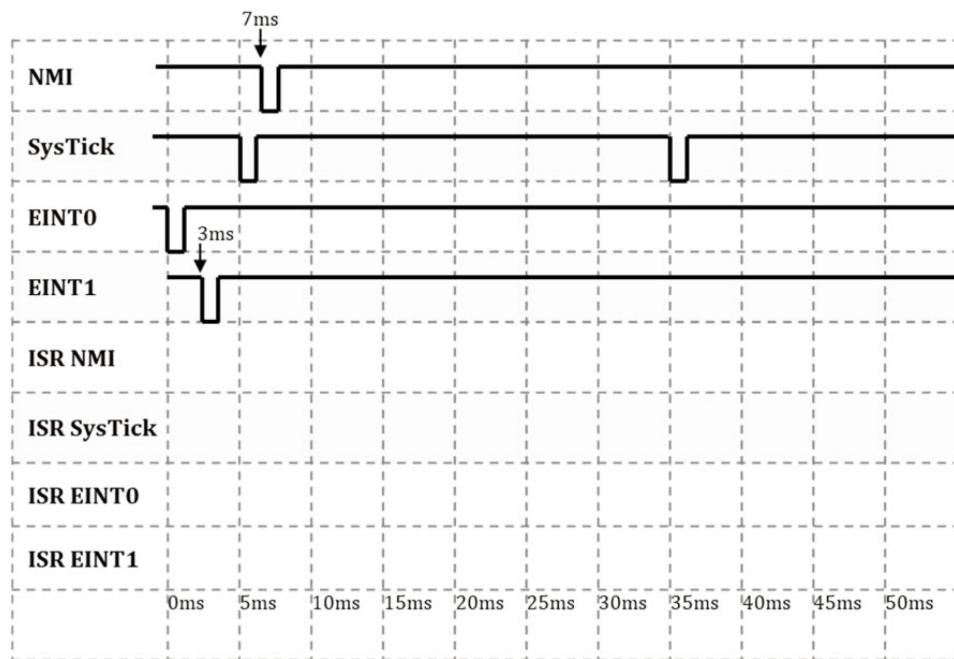


FIGURA 3.30. CRONOGRAMA DE ACTIVACIÓN DE INTERRUPCIONES.

CAPÍTULO 4

SISTEMAS ELECTRÓNICOS DIGITALES PROGRAMABLES

INTRODUCCIÓN

En este capítulo se recogen un conjunto de ejercicios relacionados con el cuarto bloque teórico, dedicado a memorias y a la organización y gestión de sistemas de memoria con microprocesadores. Con el objetivo de tener agrupados los ejercicios de la misma temática, el capítulo se ha organizado en las secciones que se indican a continuación:

- Ampliación de Memorias
- Diseño de mapas de memoria
- Diseño sistemas de memoria con el EMC
- Análisis de cronogramas de acceso
- Problemas globales

SECCIÓN 1. AMPLIACIÓN DE MEMORIAS

PROBLEMA 4.1.

Se dispone de memorias RAM de 1Gx8. Diseñe la ampliación de memoria necesaria para conseguir 2Gx16.

PROBLEMA 4.2.

Para su uso en un sistema digital que está basado en un microcontrolador de 16 bits que trabaja con ordenación de datos *little endian* se necesita una memoria con capacidad interna de **1Mbytes**, con la estructura mostrada en el esquema de la Figura 4.1, pero se dispone únicamente de chips de **512Kx2**.

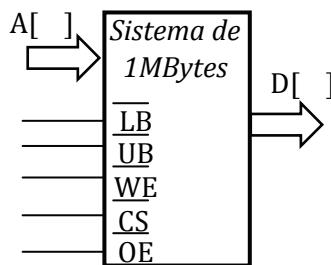


FIGURA 4.1. MEMORIA PARA EL PROBLEMA 4.2.

1. Obtenga a partir de los chips disponibles el equivalente a un chip de 512Kx8 bits.
2. A partir de chips de memoria de 512Kx8 bits, realice una ampliación para obtener el sistema de memoria equivalente de la Figura 4.1, indicando el número de líneas de datos y direcciones que tiene la memoria equivalente, así como la conexión del chip al bus de datos del microprocesador.

PROBLEMAS 4.3.

Para conectar a un sistema basado en un microcontrolador de 32 bits y 32 líneas de direcciones, se desea implementar bloques de memoria equivalente a la memoria IS62WV25616BLL, que posee:

- bus de datos de 16 bits, 18 líneas de direcciones,
- líneas de control CS# OE# WE# LB# UB#.

- Indique de qué tamaño en BYTES es la memoria IS62WV25616BLL, y represente en el esquema de la Figura 4.2 todas las entradas y salidas del bloque de memoria equivalente que se desea construir. (Dibuje en el esquema todas las líneas de control necesarias e indique a qué líneas de direcciones y datos del procesador se conectaría).

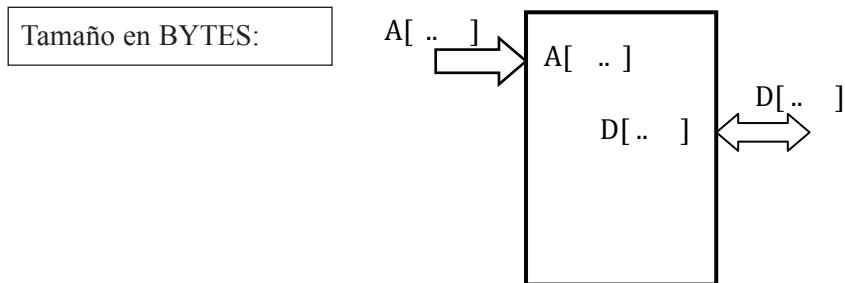


FIGURA 4.2. BLOQUE DE MEMORIA EQUIVALENTE.

- Construya el bloque de memoria equivalente a partir de chips de SRAM de 256Kx4. Tenga en cuenta que debe representar únicamente el circuito interior equivalente al bloque de la Figura 4.2, y únicamente con las líneas que entran y salen del mismo.

PROBLEMA 4.4.

Para un sistema de memoria RAM se requiere un bloque de 2Mx16, y únicamente se dispone de chips de 1Mx8 similares al de la Figura 4.3. Realice la ampliación de memoria necesaria para conseguir el bloque requerido, etiquetando correctamente los buses de datos y direcciones.

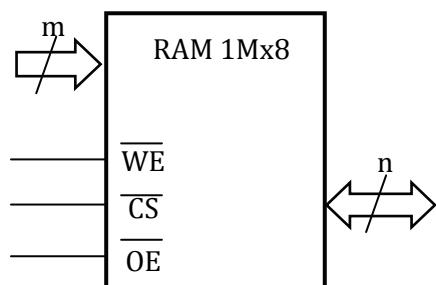


FIGURA 4.3. MEMORIA RAM DE 1Mx8.

SECCIÓN 2. DISEÑO DE MAPAS DE MEMORIA

PROBLEMA 4.5.

Un sistema digital está basado en un procesador de 32 bits que dispone, entre otras líneas, de:

- 20 líneas de direcciones A[0-19].
 - Señal de dirección válida activa a nivel bajo, AV#.
 - 3 señales activas a nivel bajo que indican el tamaño del acceso a memoria: Byte, B#; 16 bits, HW#; y 32 bits, LW#. El acceso en 16 bits sólo es posible a direcciones pares y el acceso en 32 bits a direcciones múltiplo de 4.
 - Señal indicativa del tipo de operación, R/W# (lectura/escritura#).

También se sabe que este microprocesador trabaja con ordenación de datos *big endian*.

El sistema digital necesita como **mínimo** 250 Kbytes de memoria RAM, 40 Kbytes de ROM (que deben estar colocados en las direcciones más bajas de memoria), 4 periféricos (1 chip por periférico) de 16 registros de 8 bits cada uno. ***Los chips de memorias disponibles, tanto para ROM como para RAM son de tamaño de 64 Kbytes.***

En el diseño del mapa de memoria no se desea dejar ninguna posición intermedia vacía, cumpliéndose siempre las necesidades mínimas de memoria indicadas.

- Indique en la Tabla 4.1 las direcciones de inicio y fin de cada uno de los chips que conforman el sistema.

TABLA 4.1. MAPA FUNCIONAL.

3. Diseñe la lógica de selección de los elementos del sistema (ROM, RAM, periféricos). Realice mediante circuitos electrónicos la implementación de la misma.

Nota: Genere un chip select global activo a nivel bajo por cada bloque (CS_{ROM} , CS_{RAM} , CS_{per}).

4. A partir del CS_{per} generado anteriormente, represente la conexión entre todos los chips de periféricos y el procesador incluyendo las líneas que intervienen en las operaciones de acceso (direcciones, datos y control).

PROBLEMA 4.6.

Un sistema basado en microprocesador de 8 bits y 32 líneas de direcciones, tiene las siguientes necesidades de memoria:

- 16Mbytes para tablas de datos de contenido permanente.
- 1Mbyte para pila.
- 2Mbytes para variables.
- 5Mbytes para el programa principal.
- 3Mbytes para subrutinas e ISRs.

1. Calcule las necesidades de memoria RAM y ROM para la implementación del sistema de memoria. Ajuste la capacidad final a un valor comercial.
2. Si para la implementación del sistema de memoria ROM únicamente se dispone de chips de 8Mbytes, realice la ampliación de memoria necesaria para conseguir el bloque de memoria ROM que la aplicación requiere.

PROBLEMA 4.7.

Un sistema digital que está basado en un microcontrolador de 16 bits y trabaja con ordenación de datos *big endian* dispone, entre otras líneas, de:

- 24 líneas de direcciones A[23:0].
- Señal de validación de dirección activa a nivel bajo, AV#.
- Señal SIZ, que indica el tamaño del acceso a memoria: 0 para tamaño 8 bits y 1 para tamaño 16 bits. El acceso en 16 bits sólo es posible a direcciones pares.
- Señal indicativa del tipo de operación, R/W# (lectura/escritura#).

El microcontrolador dispone de memoria ROM y RAM interna, pero además necesita conectar externamente un sistema de memoria ROM de 4 Mbytes y 2 periféricos de 4 registros de 8 bits cada uno.

Se pide:

1. A partir de chips de memoria ROM de 512Kx8 bits, realice una ampliación para obtener un sistema de memoria de 2Mx8 bits de capacidad, con los terminales mostrados en la Figura 4.4 siguiente.

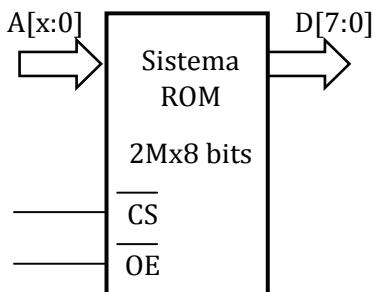


FIGURA 4.4. MEMORIA ROM DE 2Mx8.

2. A partir de los sistemas de memoria de 2Mx8 bits diseñados en el apartado anterior, y considerándolos como un chip de esa capacidad, dibuje la conexión de los sistemas de memoria al microcontrolador para obtener una capacidad total de ROM de 4Mbytes, teniendo en cuenta que debe estar mapeado a partir de la dirección 0xC0_0000. Indique la dirección inicial y final de cada uno de los sistemas de memoria de 2Mx8 bits.
3. Represente en el mapa de memoria los dos periféricos mapeados a partir de la dirección 0x80_0000. Indique las direcciones de cada uno de los 4 registros (R0 a R3) de cada periférico.

PROBLEMA 4.8.

Un sistema digital basado en un microcontrolador de 16 bits que trabaja con ordenación de datos *big endian* dispone, entre otras líneas, de:

- 26 líneas de dirección A[25:0].
- Señal de validación de dirección activa a nivel bajo, AV#.
- Señales de selección de banco, UB# y LB#.
- Señal indicativa del tipo de operación, R/W# (lectura/escritura#).

El microcontrolador dispone de memoria ROM y RAM interna, pero además necesita conectar externamente un sistema de memoria no volátil (ROM), dos sistemas de memoria volátil (RAM1 y RAM2) y una zona dedicada a periféricos

(IO). Sabiendo que para la conexión de estos elementos se ha implementado la lógica de **decodificación completa** mostrada en la Figura 4.5.

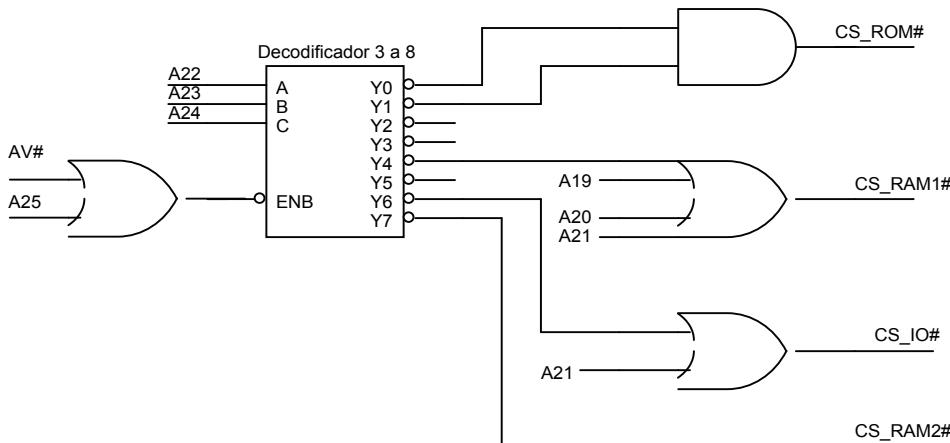


FIGURA 4.5. CIRCUITO DE DECODIFICACIÓN.

Se pide:

- Determine la capacidad en bytes de cada uno de los sistemas externos conectados al microcontrolador, y en el caso de la memoria especifique también el número de chips necesario y su capacidad. Complete la Tabla 4.2.

TABLA 4.2. DESCRIPCIÓN DE LOS CHIPS DE MEMORIA.

Sistema	Capacidad en bytes	Nº de chips
ROM		
RAM1		
RAM2		
IO		

- Complete la Tabla 4.3 con las direcciones de inicio y fin de cada uno de los sistemas externos conectados.

TABLA 4.3. DIRECCIONES DE LOS DISPOSITIVOS DE MEMORIA.

Sistema	Dirección inicio (hex)	Dirección fin (hex)
ROM		
RAM1		
RAM2		
IO		

3. Suponiendo que el sistema de memoria RAM2 tiene una capacidad de 256Kbytes, dibuje, a partir de la señal CS_RAM2#, la conexión del mismo al microcontrolador, indicando la conexión tanto de las señales de control como buses de direcciones y datos.

PROBLEMA 4.9.

Un sistema digital basado en un microprocesador de 16 bits que trabaja con ordenación de datos *big endian* dispone, entre otras líneas, de:

- 24 líneas de dirección A[23:0].
- Señal de validación de dirección activa a nivel bajo, AV#.
- Señal de tamaño de transferencia, SIZ: 0 indica tamaño byte, 1 tamaño 16 bits.
- Señal indicativa del tipo de operación, R/W# (lectura/escritura#).

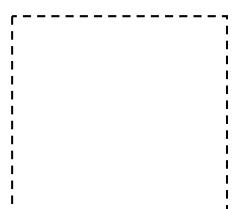
El microprocesador requiere un sistema de memoria no volátil (ROM) de 2Mbytes, un sistema de memoria volátil (RAM) de 4 Mbytes, una zona dedicada a entrada/salida (I/O) de 128 Kbytes y dos periféricos de 16 registros de 8 bits, cada uno. Sabiendo que la tabla de vectores va ubicada a partir de la posición de memoria 0, que la zona I/O y los periféricos van mapeados a continuación de la memoria ROM y que toda la zona de memoria libre debe quedar contigua, se pide:

Complete Tabla 4.4 con las direcciones de inicio y fin de cada uno de los circuitos integrados (*chips*) de memoria y periféricos con que construiría el sistema, así como de la zona de I/O. **Nota:** la implementación se debe realizar con el mínimo número de chips de memoria de capacidad estándar y bus de datos de 8 bits.

TABLA 4.4. DIRECCIONES DE LOS CIRCUITOS INTEGRADOS.

Ayúdese de una representación gráfica del mapa de memoria

Chip	Dirección inicio (hex)	Dirección fin (hex)



2. Genere las señales de selección de cada bloque, $\overline{CS_ROM}$, $\overline{CS_RAM}$, $\overline{CS_IO}$, $\overline{CS_PERS}$ a partir de las líneas de dirección

$[A_{23} \dots A_{16}]$, empleando **decodificación incompleta** usando el menor número posible de líneas de direcciones. Complete la Tabla 4.5.

TABLA 4.5

A₂₃	A₂₂	A₂₁	A₂₀	A₁₉	A₁₈	A₁₇	A₁₆	Sistema

3. A partir de la señal $\overline{CS_RAM}$ obtenida en el apartado anterior, dibuje la conexión de los chips de memoria RAM (chips de 8 bits) al microprocesador, indicando la conexión tanto de las señales de control como los buses de direcciones y datos. Asuma que se trabaja con datos alineados.

PROBLEMA 4.10.

Un sistema digital basado en un microprocesador de 32 bits que trabaja con ordenación de datos **big endian** dispone, entre otras líneas, de:

- 24 líneas de dirección $A[23:0]$.
- Señal de validación de dirección activa a nivel bajo, \overline{AS} , y señal de lectura o escritura, R/\overline{W} .
- 2 señales $\overline{SZ1}$ y $\overline{SZ0}$ para indicar el acceso a 8, 16 o 32 bits ([1,1] indica acceso a 8 bits, [0,0] indica acceso a 32 bits y el resto de códigos accesos a 16 bits).

El microprocesador requiere de un sistema de memoria de 8Mbytes para almacenar: la tabla de vectores, todo el código y tablas de datos precalculados (pueden ser de 8, 16 y 32 bits). Además, se necesita un sistema de memoria de 6 Mbytes accesible en 8, 16 y 32 bits para tablas de datos, variables y manejo de pila. También se desea mapear en memoria una zona de entrada/salida (I/O) de 16 Kbytes y 4 periféricos (PERx) de 64 registros de 8 bits cada uno. Sabiendo que la tabla de vectores va ubicada a partir de la posición de memoria 0, se pide:

1. Indique cuántos chips de memoria ROM y RAM utilizaría y de qué capacidad, teniendo en cuenta el espacio de direccionamiento total del procesador y la cantidad de memoria requerida.

2. Realice en la Tabla 4.6 un esquema con la colocación de los chips en el mapa de memoria indicando direcciones de inicio y fin de cada bloque y complete la Tabla 4.7 con las direcciones base de los periféricos, sabiendo que deben ir colocados después de la zona de I/O.

TABLA 4.6

Direcciones	Chips			

TABLA 4.7

A[23:20]	A[19:16]	A[15:12]	A[11:8]	A[7:4]	A[3:0]	Chips
						Per0
						Per1
						Per2
						Per3

3. Genere las señales de selección del bloque de ROM, $\overline{\text{CS}}_{\text{ROM}}$, empleando **decodificación completa**.
4. Considerando que existe una señal $\overline{\text{CS}}_{\text{ROM}}$, genere las señales para la selección de los bancos 0 y 2 y realice la conexión de las memorias ROM de la Figura 4.6 asumiendo que son de 2Mbyte de capacidad cada una.

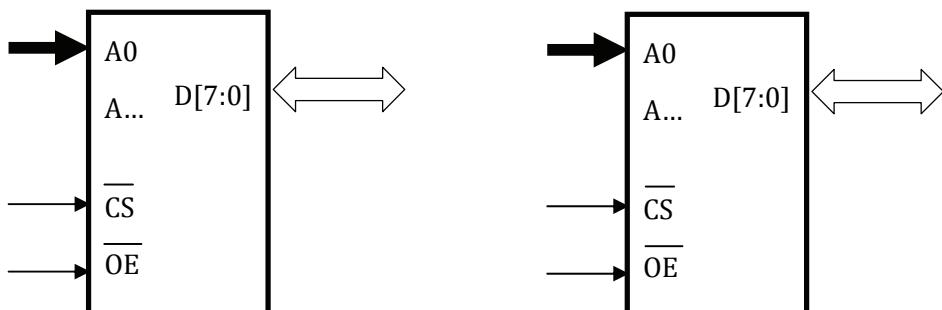


FIGURA 4.6. CONEXIÓN DE LOS BANCOS 0 Y 2.

PROBLEMA 4.11.

Un microprocesador de 16 bits que trabaja con ordenación de datos *big endian* dispone, entre otras líneas, de:

- 23 líneas de direcciones $A[23-1]$.
- 1 señal de dirección válida activa a nivel bajo, \overline{AS} .
- Señal indicativa del tipo de operación, R/W.
- 2 señales que indican el tamaño del acceso a memoria y el banco que se activa: $\overline{B0}$.

TABLA 4.8. DESCRIPCIÓN DE LAS LÍNEAS: $\overline{B0}$ Y $\overline{B1}$.

$\overline{B0}$	$\overline{B1}$	Tamaño, Banco
1	1	No hay acceso a memoria
0	1	Tamaño byte, banco par ($2N$)
1	0	Tamaño byte, banco impar ($2N+1$)
0	0	Tamaño 16bits, ambos bancos

1. Para dicho microprocesador se desea implementar un sistema de memoria RAM de 2Mbytes. Para ello se emplea la señal $\overline{CS_RAM}$, que se supone ya generada, que direcciona el último bloque de 2Mbytes del espacio de direccionamiento del microprocesador, y chips de memoria similares a los mostrados en la Figura 4.7. Realice las conexiones correspondientes entre las líneas de dirección, datos y control del microprocesador y las correspondientes de las memorias utilizadas para conseguir disponer del bloque de memoria requerido.

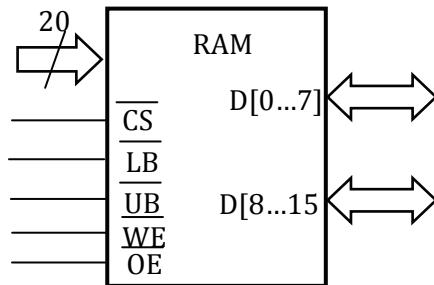


FIGURA 4.7. CHIP DE MEMORIA RAM.

2. Implemente la señal $\overline{CS_RAM}$, haciendo uso de decodificación completa.

SECCIÓN 3. DISEÑO DE SISTEMAS DE MEMORIA CON EL EMC

PROBLEMA 4.12.

En un sistema digital basado en el microcontrolador LPC1788, se desea conectar memoria externa y periféricos como se indica a continuación (la configuración en el registro CNFIG se ha realizado para que la ordenación sea big endian). En el diseño se debe realizar decodificación completa, y usar el mínimo nº de espacios de direccionamiento accesibles a través del EMC (dejar libre el máximo para futuras ampliaciones).

Se dispone de chips de SRAM de 4Mx8, dispositivos FLASH de 8Mx8. Los periféricos son de 128K registros de tamaño de datos 8 bits. El acceso a datos siempre será alineado

- La cantidad de memoria necesaria en el sistema es:
- 4MB de memoria SRAM accesible en tamaño de datos 8, 16 y 32 bits.
- 32MB de memoria FLASH que va a ser accedida siempre en tamaño 32 bits alineados (se puede suponer que nunca ocurrirán accesos en otro tamaño).
- 4 periféricos de 128K. Las direcciones de los registros internos deben ser consecutivas (estructurado en un solo banco).

Responda a las siguientes preguntas:

1. Nº mínimo de chips y tamaños de los mismos a utilizar
2. Indique en cuál o cuáles de las zonas del espacio accesible por EMC los colocará, así como la direcciones base y final de cada **uno de los chips**.
3. Valor de configuración de los bits *Memory width [1:0]* de los registros STATICCONFIG correspondiente.
4. Obtenga el circuito que genera las 3 señales de CS# generales para acceder a SRAM, FLASH y Periféricos.
5. Diseñe y conecte el sistema de memoria FLASH, indicando todas las líneas de direcciones, datos y control necesarias.

PROBLEMA 4.13.

Para implementar en el microcontrolador LPC1788 un sistema de memoria externa formado por 16Mbytes de memoria no volátil y 4 Mbytes de memoria volátil se dispone de los siguientes chips:

- Se usan dispositivos SST38VF6401, de 4Mx16, para implementar el espacio no volátil a partir de la dirección 0x9000_0000, accesible únicamente en tamaño 32 bits.

- Se usan dispositivos BS62LV1600, de 2Mx8, para implementar el espacio volátil a partir de la dirección 0x9800_0000, accesibles en tamaños 8, 16 y 32 bits.
- Se desea usar un bus de datos de 32 bits.

Realice la conexión de los chips al microcontrolador. Utilice la señal de CS necesaria para cada espacio, configurando además el registro STATICCONFIGn del CSn de forma adecuada para el funcionamiento del sistema. No tenga en cuenta la generación de periodos de espera.

PROBLEMA 4.14.

A un sistema digital basado en el microcontrolador LPC1788, se desean conectar elementos periféricos y memoria adicional, necesarios para una determinada aplicación, que serán accedidos a través del EMC (ver Figura 4.9). En todo momento se va a trabajar con ordenación Little Endian, y todos los elementos de memoria y periféricos externos deberán estar colocados en direcciones **superiores a 0x9800_0000**.

En los cálculos de diseño (mapa funcional) se ha estimado que serán necesarios **8MBytes de memoria ROM** accesibles en tamaño Word (4Bytes), **32MBytes de memoria RAM** que puedan ser accedidos en tamaños de 8, 16 y 32 bits y **8 periféricos de 32 bits** (Per0 a Per7), con capacidad cada uno **de 1Mbyte** accesibles en tamaños 16 y 32 bits que disponen de dos entradas de habilitación de parte alta, H#, y baja, L# de los registros.

Indique:

1. Nº mínimo de chips y tamaños de los mismos a utilizar, asumiendo que se dispone de todas las posibilidades comerciales. Complete la Tabla 4.9.

TABLA 4.9. CHIPS Y TAMAÑO.

	Nº chips	Tamaño de cada chip
ROM		
RAM		
Periféricos		

2. Teniendo en cuenta que se deben utilizar el **mínimo número de espacios de direccionamiento** (de los 4 disponibles a través del EMC para memo-

ria estática) y que dentro de estos **no se debe dejar ningún espacio vacío entre chips** indique las direcciones base y final de cada uno de los chips. Complete la Tabla 4.10.

TABLA 4.10. DIRECCIONES DE INICIO Y FIN.

Bloque	Dirección Base	Dirección final
ROM		
RAM		
Periféricos		

3. Obtenga las 3 señales de CS# generales para acceder a RAM, ROM y Periféricos, así como la señal de CSperx# para acceder a cada uno de los periféricos.
4. Diseñe la conexión de los periféricos Per0 y Per7 al microcontrolador mostrados en la Figura 4.8, indicando todas las líneas de direcciones, datos y control que son necesarias.

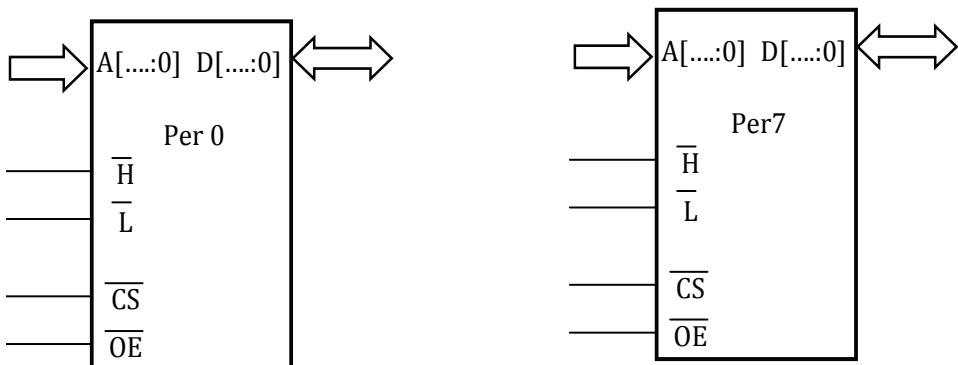


FIGURA 4.8. CONEXIÓN DE LOS PERIFÉRICOS 0 Y 7.

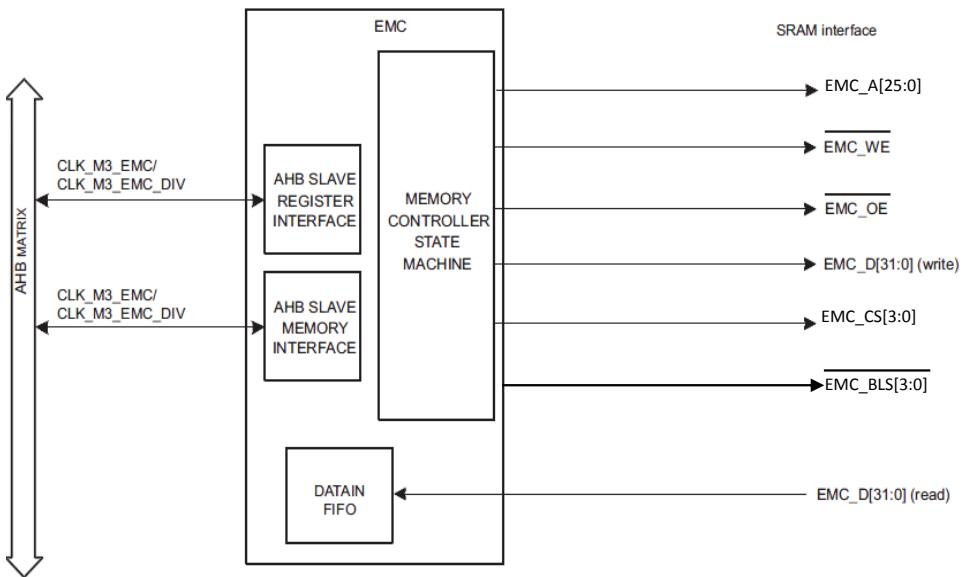


FIGURA 4.9. EMC SRAM INTERFACE.

PROBLEMA 4.15.

A un sistema digital basado en el microcontrolador LPC1788, se desean conectar elementos periféricos y memoria adicional necesarios para una determinada aplicación, que serán accedidos a través del EMC. En todo momento se va a trabajar con ordenación **Little Endian**, y todos los elementos de memoria y periféricos externos deberán estar colocados en el bloque de memoria estática para el que el EMC produce la señal de CS1#.

En los cálculos de diseño (mapa funcional) se ha estimado que serán necesarios **8 MBytes de memoria RAM** y **4MBytes de memoria ROM** que puedan ser accedidos (en ambos casos) en tamaños de 8, 16 y 32 bits, y **4 periféricos de 8 bits** con capacidad cada uno de **256Kbytes**.

1. Complete la Tabla 4.11 con el número mínimo de chips y tamaños de los mismos a utilizar, asumiendo que se dispone de todas las posibilidades comerciales.

TABLA 4.11. NÚMERO DE CHIPS Y TAMAÑO.

Tipo de dispositivo	Nº chips	Tamaño de cada chip

2. Teniendo en cuenta que si es posible **no se debe dejar ningún espacio vacío entre chips**, complete la Tabla 4.12 con las direcciones base y final de cada uno de los chips.

TABLA 4.12. DIRECCIONES DE INICIO Y FIN.

Tipo de dispositivos	Dirección Base	Dirección final

3. Obtenga las 3 señales de CS# generales para acceder a **RAM, ROM y Periféricos** con decodificación completa, así como la señal de **CSperx#** para acceder a cada uno de los periféricos.
4. Diseñe la conexión al micro-controlador de los chips de RAM del **banco 0** y **banco 3** en la Figura 4.10, indicando todas las líneas de direcciones, datos y control que son necesarias.

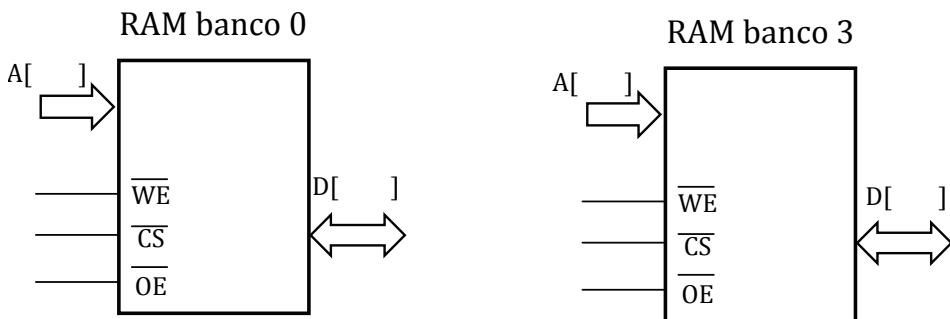


FIGURA 4.10. CONEXIÓN DE LOS BANCOS 0 Y 3.

PROBLEMA 4.16.

A un sistema digital basado en el microcontrolador LPC1788 se desean conectar chips de memoria externa SRAM (16MB), EEPROM (32MB) Y FLASH (16MB), necesarios para una determinada aplicación. Serán accedidos a través del EMC, con posibilidad de hacerlo en todos los casos en tamaños de datos **8, 16 y 32 bits**, con ordenación **Little Endian**.

Todos los chips de memoria deberán estar colocados de forma contigua (sin dejar ningún espacio vacío) a partir de la dirección 0x9800_0000.

Indique:

1. Complete la Tabla 4.13 con el número mínimo de chips y tamaños de los mismos a utilizar, asumiendo que se dispone de todas las posibilidades comerciales.

TABLA 4.13. NÚMERO DE CHIPS Y TAMAÑO.

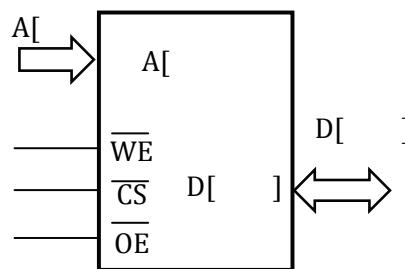
Tipo de dispositivo	Nº chips	Tamaño de cada chip

2. Indique las direcciones base y final de cada uno de los chips en la Tabla 4.14.

TABLA 4.14. DIRECCIONES DE INICIO Y FIN.

Tipo de dispositivo	Dirección Base	Dirección final

3. Obtenga las 3 señales de CS# generales para acceder a SRAM, EEPROM y FLASH con decodificación completa.
4. Dibuje la conexión al microcontrolador y **TODOS** los chips de SRAM, indicando todas las líneas de direcciones, datos y control que son necesarias (el bloque que se representa en la Figura 4.11 es un ejemplo de cómo indicar los chips de memoria).

**FIGURA 4.11. EJEMPLO DE REPRESENTACIÓN DE LA MEMORIA SRAM.****PROBLEMA 4.17.**

A un sistema digital basado en el microcontrolador LPC1788 y configurado con ordenación de datos *little endian* se desean conectar los siguientes elementos externos, que serán accedidos a través del EMC:

- 16 Mbytes de memoria RAM accesible en 8, 16 y 32 bits.
- 2 periféricos de 32 registros de 16 bits (Per16_0 y Per16_1), accesibles únicamente en 16 bits.
- 4 periféricos de 8 bits y 4 registros internos (Per8_0, Per8_1, Per8_2, Per8_3).

La memoria RAM estará mapeada en el bloque de memoria estática para el que el EMC produce la señal de CS1# y todos los periféricos en el bloque de memoria estática para el que el EMC produce la señal de CS2#, el cual se ha configurado con un ancho de bus de 16 bits.

Se pide:

1. Sabiendo que la memoria se selecciona con decodificación completa y está implementada con chips CY7C1079DV33 (véase la Figura 4.12), dibuje la conexión de los chips de memoria al microcontrolador.
2. Sabiendo que los periféricos están mapeados consecutivamente, comenzando con los de 16 bits en las direcciones más bajas y sin que queden direcciones libres entre ellos, complete la Tabla 4.15 con las direcciones de inicio y fin de cada uno de los periféricos.

TABLA 4.15. DIRECCIONES DE INICIO Y FIN.

Sistema	Dirección inicio (hex)	Dirección fin (hex)
Per16_0		
Per16_1		
Per8_0		
Per8_1		
Per8_2		
Per8_2		

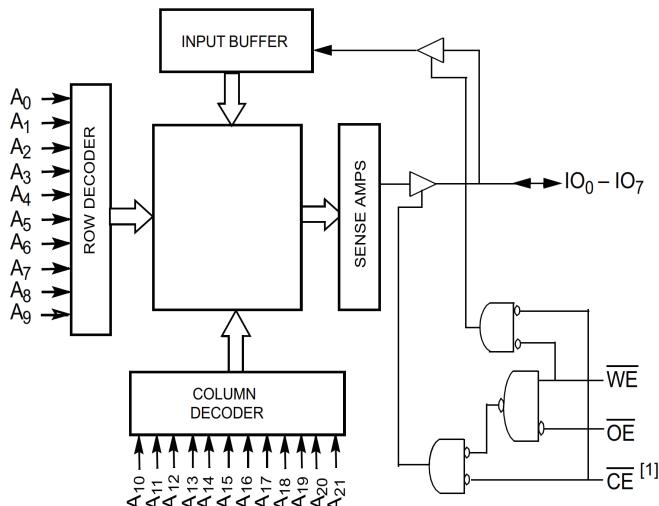


FIGURA 4.12. DIAGRAMA DE BLOQUES DE LA MEMORIA CY7C1079DV33.

PROBLEMA 4.18.

A un sistema digital basado en el microcontrolador LPC1788 se desean conectar cuatro periféricos (Per0 a Per3) de dos registros (Reg0 y Reg1) de 8 bits cada uno, necesarios para una determinada aplicación.

Estos periféricos deben estar mapeados a partir de la dirección 0x9BFF_0000, sin dejar posiciones de memoria desocupadas entre ellos, empleando para ello el módulo EMC.

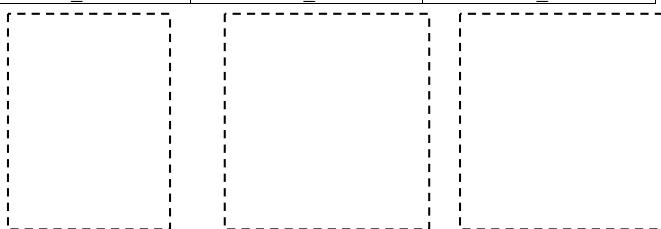
Se pide:

1. A partir de las salidas que ofrece el EMC, genere una señal de selección (CS_PER) con **decodificación completa** que se active cuando el microcontrolador accede a alguno de los cuatro periféricos. En la Tabla 4.17 se dispone de las direcciones base de cada salida de *chip select* del EMC.
2. Complete la Tabla 4.16 indicando las direcciones de cada uno de los registros, dependiendo del ancho de memoria que se hubiera configurado (*Memory Width*). En la Tabla 4.18 dispone de la configuración del campo *Memory Width*.

TABLA 4.16. DIRECCIONES DE LOS REGISTROS SEGÚN LA CONFIGURACIÓN DEL CAMPO *MEMORY WIDTH*.

Periférico	Registro	MW=00	MW=01	MW=10
Per0	Reg0	0x9BFF_000.....	0x9BFF_000.....	0x9BFF_000.....
	Reg1	0x9BFF_000.....	0x9BFF_000.....	0x9BFF_000.....
Per1	Reg0	0x9BFF_000.....	0x9BFF_000.....	0x9BFF_000.....
	Reg1	0x9BFF_000.....	0x9BFF_000.....	0x9BFF_000.....
Per2	Reg0	0x9BFF_000.....	0x9BFF_000.....	0x9BFF_000.....
	Reg1	0x9BFF_000.....	0x9BFF_000.....	0x9BFF_000.....
Per3	Reg0	0x9BFF_000.....	0x9BFF_000.....	0x9BFF_000.....
	Reg1	0x9BFF_000.....	0x9BFF_000.....	0x9BFF_000.....

Ayúdese de una representación gráfica del mapa de memoria



3. Suponiendo que se ha configurado un ancho de memoria de 8 bits, conecte los periféricos de la Figura 4.13 las señales del EMC, empleando para ello la señal CS_PER generada en el apartado 1.

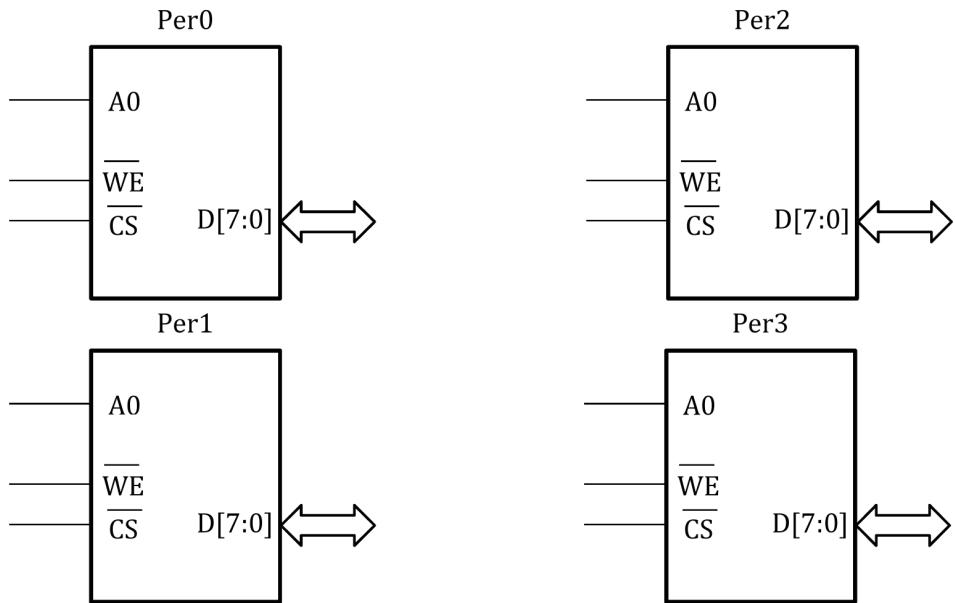


FIGURA 4.13. CONEXIÓN DE PERIFÉRICOS AL EMC.

TABLA 4.17. MEMORIA ESTÁTICA OFF-CHIP VÍA EMC.

Four static memory chip selects:	
0x8000 0000 - 0x83FF FFFF	Static memory chip select 0 (up to 64 MB)
0x9000 0000 - 0x93FF FFFF	Static memory chip select 1 (up to 64 MB)
0x9800 0000 - 0x9BFF FFFF	Static memory chip select 2 (up to 64 MB)
0x9C00 0000 - 0x9FFF FFFF	Static memory chip select 3 (up to 64 MB)

TABLA 4.18. CONFIGURACIÓN DEL CAMPO *MEMORY WIDTH*.

Bit	Symbol	Value	Description
1:0	MW		Memory width.
		0x0	8 bit (POR reset value).
		0x1	16 bit.
		0x2	32 bit.
		0x3	Reserved.

PROBLEMA 4.19.

A un sistema digital basado en el microcontrolador LPC178x, se desean conectar elementos adicionales, que serán accedidos a través del EMC. En el espacio de direccionamiento que comienza en la dirección 0x9C00_0000 se desean colocar dispositivos a los que se va a acceder únicamente en tamaño de dato de 8 bits (3 dispositivos, DISP0, DISP1 y DISP2, de 256 Kbytes cada uno) y en el espacio que comienza en la dirección 0x8000_0000 dispositivos de 16 bits que se pueden acceder en tamaño 8 y 16 bits (2 dispositivos, DISP3 y DISP4, de 1Mbyte cada uno).

En la Figura 4.14, la Tabla 4.21 y la Tabla 4.22 se proporcionan las conexiones del EMC, la configuración del campo *Memory Width* y los espacios de direccionamiento para memoria SRAM.

Sabiendo que los dispositivos DISP3 y DISP4 se deben mapear en las últimas direcciones del espacio de direccionamiento en el que se van a colocar, indique:

1. Cómo configuraría los bits MW correspondientes a los espacios de direccionamiento aludidos, y qué direcciones ocupará cada uno de los elementos a conectar. Complete la Tabla 4.19 y la Tabla 4.20.

TABLA 4.19

CSx	MW

TABLA 4.20

Elemento	Dirección Base	Dirección final

2. Realice el conexionado completo (líneas de direcciones, datos y control) entre el procesador y los dos 2 dispositivos de 1Mbyte (son dispositivos de lectura/escritura) haciendo uso de decodificación completa, sabiendo que tienen dos líneas y mediante las que se indica el acceso a los 8 bits de mayor peso y el acceso a los 8 bits de menor peso respectivamente.

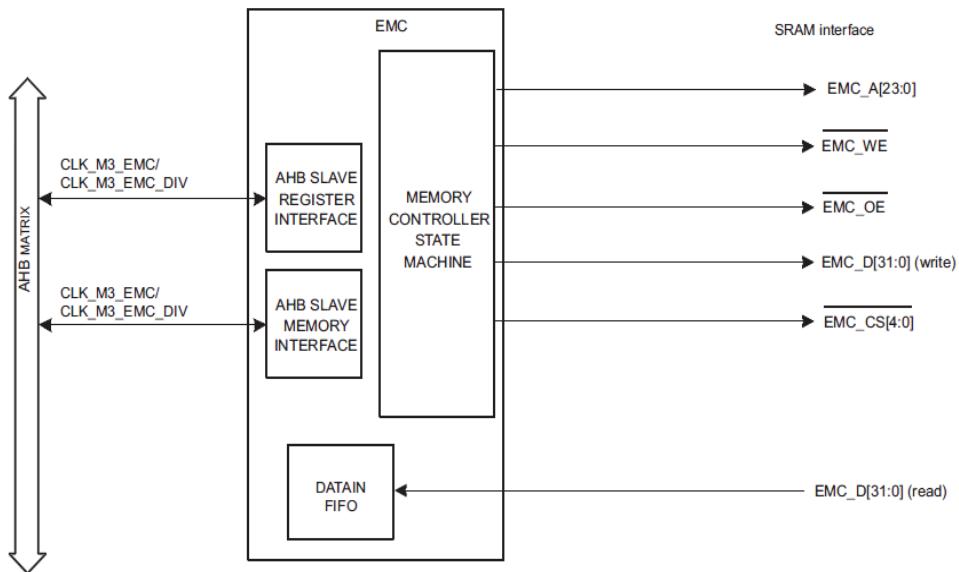


FIGURA 4.14. SRAM EMC INTERFACE.

TABLA 4.21. MEMORIA ESTÁTICA OFF-CHIP VÍA EMC.

Four static memory chip selects:	
0x8000 0000 - 0x83FF FFFF	Static memory chip select 0 (up to 64 MB)
0x9000 0000 - 0x93FF FFFF	Static memory chip select 1 (up to 64 MB)
0x9800 0000 - 0x9BFF FFFF	Static memory chip select 2 (up to 64 MB)
0x9C00 0000 - 0x9FFF FFFF	Static memory chip select 3 (up to 64 MB)

TABLA 4.22. CONFIGURACIÓN DEL CAMPO *MEMORY WIDTH*.

Bit	Symbol	Value	Description
1:0	MW		Memory width.
		0x0	8 bit (POR reset value).
		0x1	16 bit.
		0x2	32 bit.
		0x3	Reserved.

PROBLEMA 4.20.

A un sistema digital basado en el microcontrolador LPC178x, se desean conectar los cuatro chips de memoria SRAM mostrados en la figura haciendo uso del interfaz EMC (ver la Figura 4.14, la Tabla 4.21 y la Tabla 4.22). Para ello, se decide utilizar el rango de direcciones necesario del espacio de direccionamiento que comienza en la dirección 0x9C00_0000. A estas memorias se accederá exclusivamente en tamaño de dato de 8 bits.

- Realice las conexiones sobre la Figura 4.15 utilizando las líneas del EMC, incluyendo los elementos necesarios, para que se pueda acceder adecuadamente a las memorias, haciendo uso de decodificación completa. Recuerde etiquetar también los buses de datos y direcciones de cada chip.

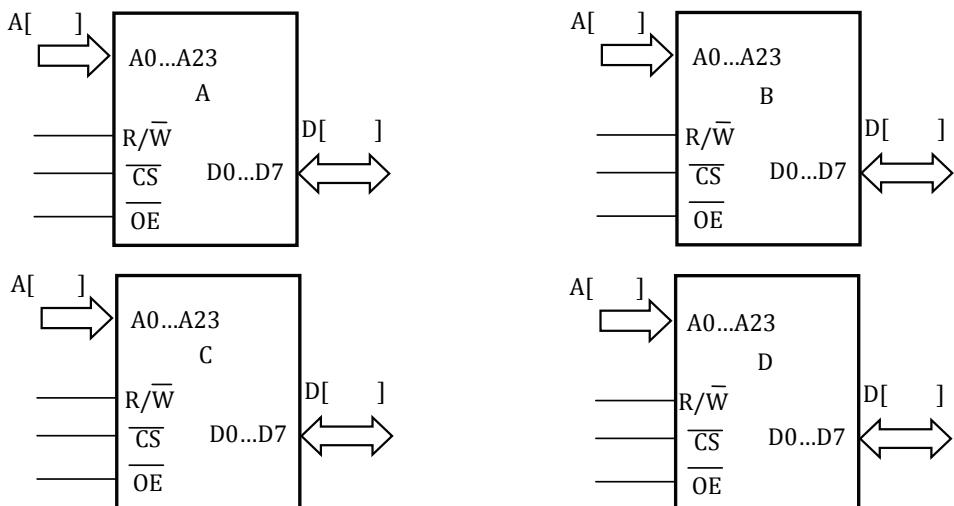


FIGURA 4.15. CONEXIÓN DE MEMORIAS A TRAVÉS DEL EMC.

- Complete la Tabla 4.23, indicando el rango de direcciones que ocupa cada chip de memoria.

TABLA 4.23. DIRECCIONES DE INICIO Y FIN DE LOS CHIPS DE MEMORIA.

Memoria	Dir. Inicio (hex)	Dir. Fin (hex)
A		
B		
C		
D		

SECCIÓN 4. ANÁLISIS DE CRONOGRAMAS DE ACCESO

PROBLEMA 4.21.

La Figura 4.16 muestra la conexión de un procesador con la memoria Flash (S29AL032D-90).

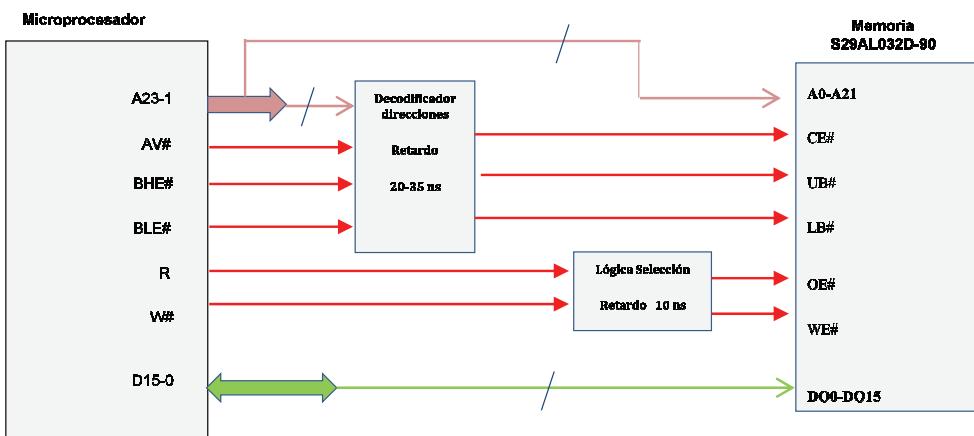


FIGURA 4.16. CONEXIÓN DE LA MEMORIA CON EL MICROPROCESADOR.

Los cronogramas de acceso a lectura del procesador son los indicados en la Figura 4.18 y los datos de los accesos a la memoria los mostrados en la Figura 4.19.

Las señales del procesador y la memoria son las siguientes:

- AV# indica que en el bus existe una dirección válida para una transacción,
- BHE#, BLE# seleccionan el banco o bancos de memoria con los que se realizará la transferencia.
- R señal de habilitación de lectura. Se activa a nivel alto en las operaciones de lectura.
- W# señal de habilitación de escritura. Se activa a nivel bajo en las operaciones de escritura.

Se pide:

1. Dibuje en el cronograma de la Figura 4.17 las señales que se utilizan para obtener las de CE# y OE# a conectar a la memoria, así como estas dos, indicando claramente los tiempos y los retardos con que se activan y el tiempo restante hasta la lectura de datos por parte del procesador.

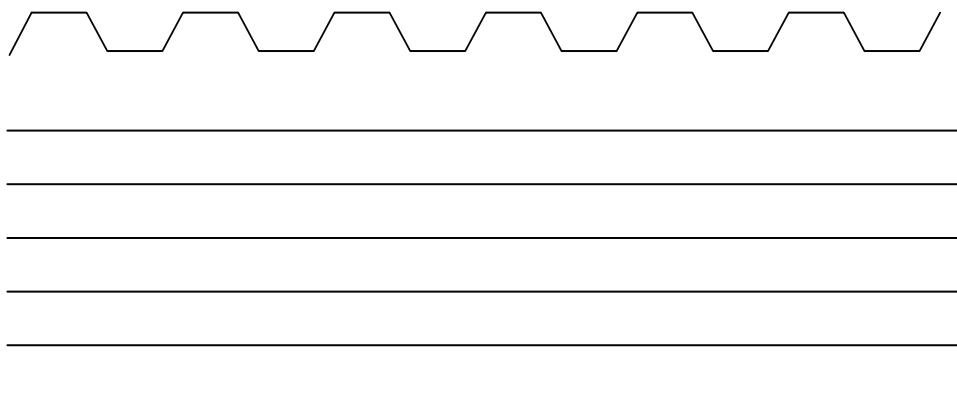


FIGURA 4.17. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

- Determine si se podría acceder sin introducir ciclos de espera. En caso de no poderse acceder, indique justificadamente cuántos ciclos de espera se deben introducir.

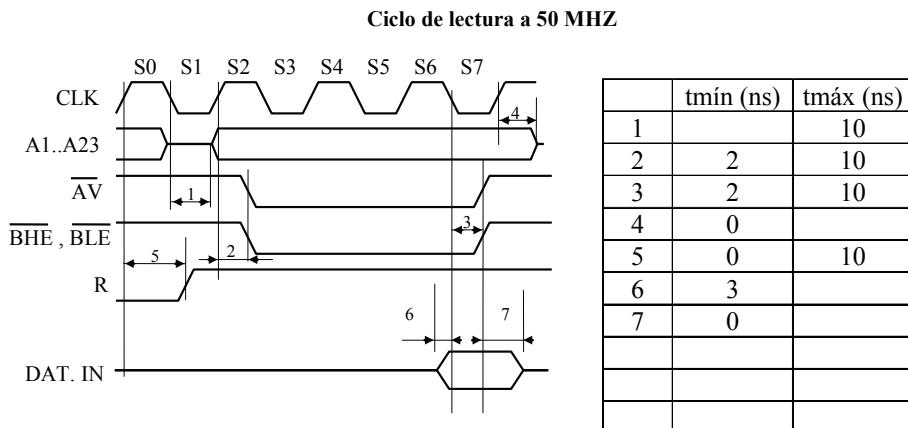
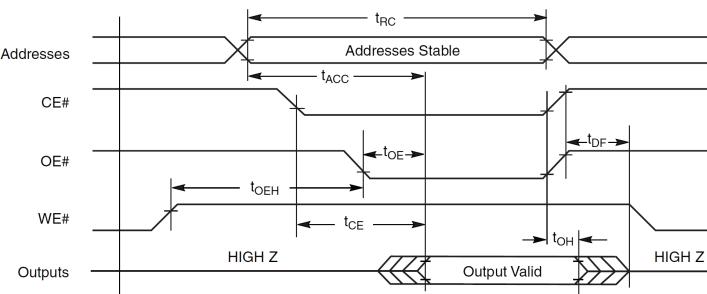


FIGURA 4.18. CRONOGRAMA DE ACCESO EN LECTURA DEL MICROPROCESADOR.

Read Operations

Parameter	JEDEC	Std	Description	Test Setup		Speed Options	Unit
				70	90		
t_{AVAV}	t_{RC}		Read Cycle Time (Note 1)		Min	70	90 ns
t_{AVQV}	t_{ACC}		Address to Output Delay	$CE\# = V_{IL}$ $OE\# = V_{IL}$	Max	70	90 ns
t_{ELQV}	t_{CE}		Chip Enable to Output Delay	$OE\# = V_{IL}$	Max	70	90 ns
t_{GLQV}	t_{QE}		Output Enable to Output Delay		Max	30	35 ns
t_{EHQZ}	t_{DF}		Chip Enable to Output High Z (Note 1)		Max	25	30 ns
t_{GHQZ}	t_{DF}		Output Enable to Output High Z (Note 1)		Max	25	30 ns
t_{OEH}			Output Enable Hold Time (Note 1)	Read	Min	0	ns
				Toggle and Data# Polling	Min	10	ns
t_{AXQX}	t_{OH}		Output Hold Time From Addresses, CE# or OE#, Whichever Occurs First (Note 1)		Min	0	ns

**FIGURA 4.19. DATOS DE LOS ACCESOS A MEMORIA.****PROBLEMA 4.22.**

La Figura 4.20 muestra la conexión de un microprocesador con la memoria Flash (S29AL032D-70).

Los cronogramas de acceso a lectura del procesador son los indicados en la Figura 4.22 y los datos de los accesos a la memoria los mostrados en la Figura 4.23.

Alguna de las señales del microprocesador son las siguientes:

- AV# indica que en el bus existe una dirección válida para una transacción,
- BHE#, BLE# seleccionan el banco o bancos de memoria con los que se realizará la transferencia
- R señal de habilitación de lectura. Se activa a nivel alto en las operaciones de lectura.
- W# señal de habilitación de escritura. Se activa a nivel bajo en las operaciones de escritura.

Y de la memoria cabe destacar las siguientes señales:

- UB#, LB# seleccionan los bancos internos de la memoria

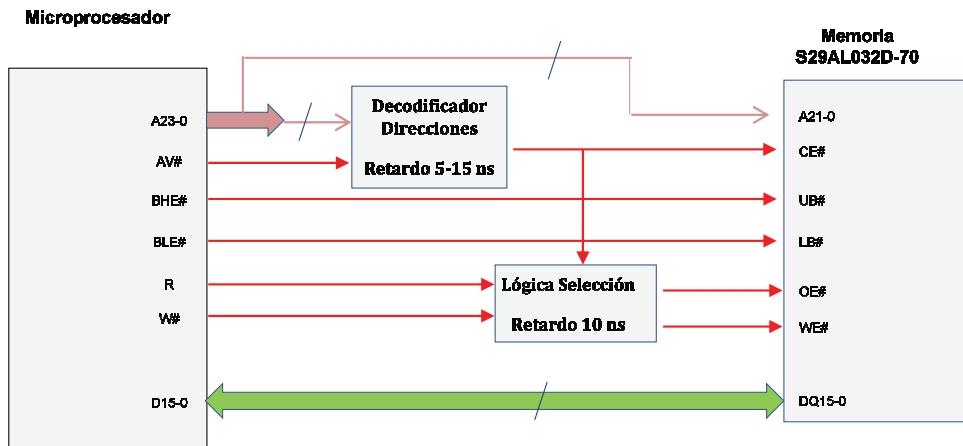


FIGURA 4.20. CONEXIÓN DE LA MEMORIA CON EL MICROPROCESADOR.

1. Dibuje el siguiente cronograma las señales que se utilizan para obtener las de CE# y OE# a conectar a la memoria, así como estas dos, indicando claramente los tiempos y los retardos con que se activan y el tiempo restante hasta la lectura de datos por parte del procesador.

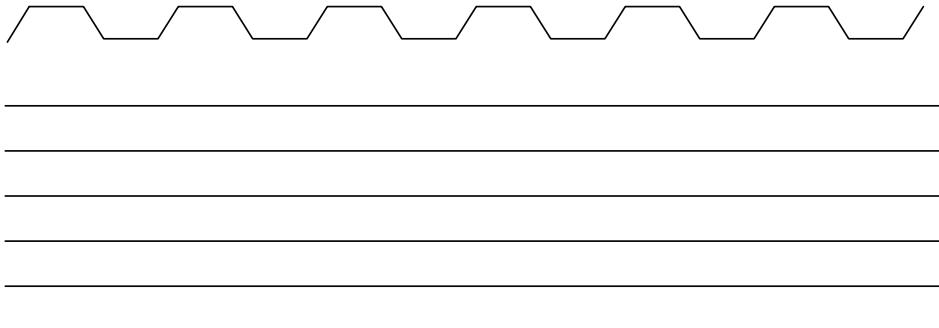


FIGURA 4.21. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

2. Determine si se podría acceder sin introducir ciclos de espera. En caso de no poderse acceder, indique justificadamente cuántos ciclos de espera se deben introducir.

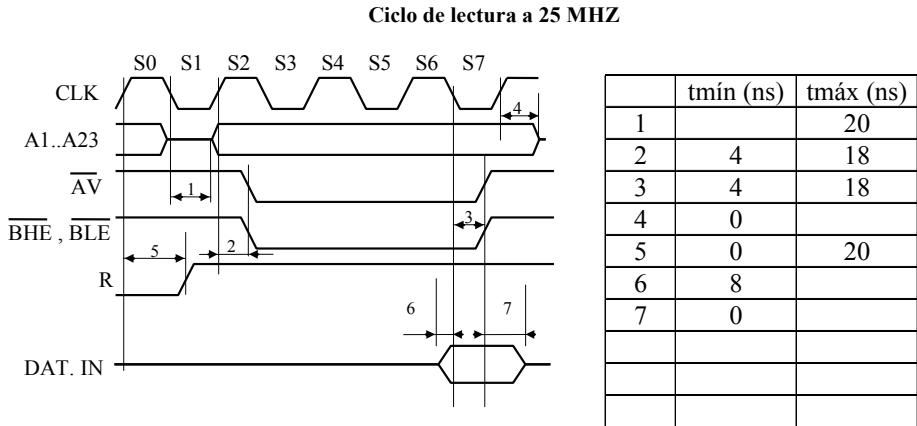


FIGURA 4.22. CRONOGRAMA DE ACCESO EN LECTURA DEL MICROPROCESADOR.

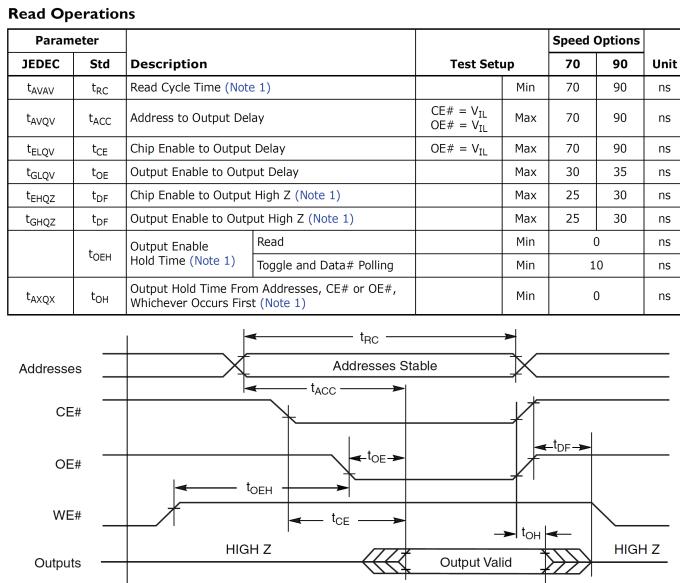


FIGURA 4.23. DATOS DE LOS ACCESOS A MEMORIA.

PROBLEMA 4.23.

La Figura 4.24 muestra la conexión de un procesador que funciona a 40 MHz con una memoria no volátil Flash (S29AL032D-70).

Los cronogramas de acceso a lectura del procesador son los indicados en la Figura 4.26 y los datos de los accesos a la memoria son los mostrados en la Figura 4.27.

Las señales del procesador y la memoria son las siguientes:

- AV indica que en el bus existe una dirección válida para una transacción,
- BHE, BLE seleccionan el banco o bancos de memoria con los que se realizará la transferencia.
- R señal de habilitación de lectura. Se activa a nivel alto en las operaciones de lectura.

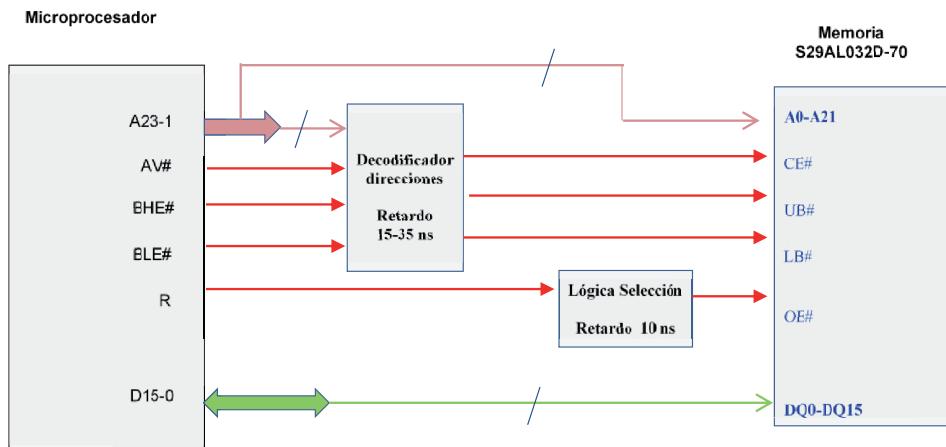


FIGURA 4.24. CONEXIÓN DEL MICROPROCESADOR CON UNA MEMORIA FLASH.

Se pide:

1. Dibuje en el cronograma de la Figura 4.25 la temporización de las señales que se utilizan para obtener las de CE# y OE# a conectar a la memoria, así como estas dos, indicando claramente los tiempos y los retardos con que se activan y el tiempo restante hasta la lectura de datos por parte del procesador.

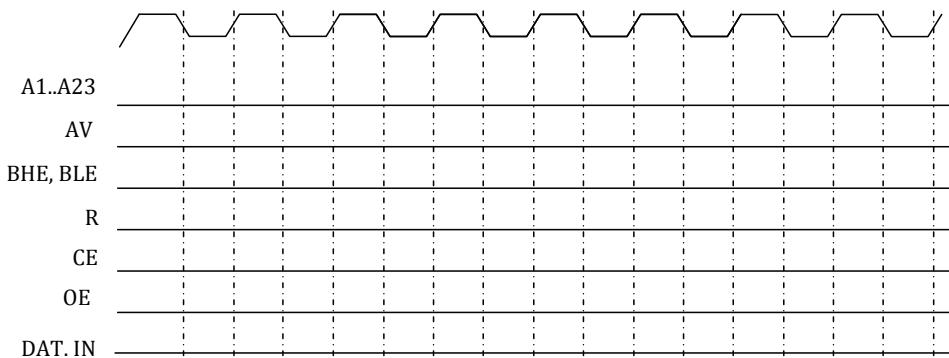


FIGURA 4.25. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

2. Determine si se podría acceder sin introducir ciclos de espera. En caso de no poderse acceder, indique justificadamente cuántos ciclos de espera se deben introducir.

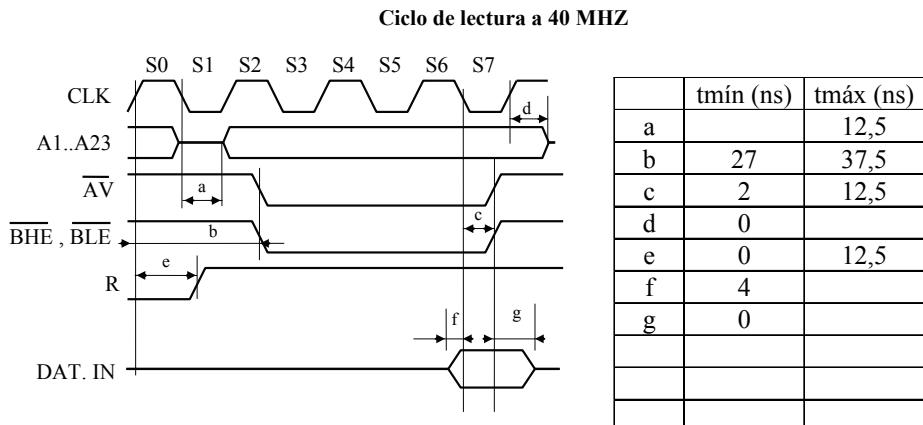


FIGURA 4.26. CRONOGRAMA DE ACCESO EN LECTURA DEL MICROPROCESADOR.

Read Operations

Parameter		Description	Test Setup		Speed Options		Unit
JEDEC	Std				70	90	
t_{AVAV}	t_{RC}	Read Cycle Time (Note 1)		Min	70	90	ns
t_{AVQV}	t_{ACC}	Address to Output Delay	$CE\# = V_{IL}$ $OE\# = V_{IL}$	Max	70	90	ns
t_{ELQV}	t_{CE}	Chip Enable to Output Delay	$OE\# = V_{IL}$	Max	70	90	ns
t_{GLQV}	t_{OE}	Output Enable to Output Delay		Max	30	35	ns
t_{EHQZ}	t_{DF}	Chip Enable to Output High Z (Note 1)		Max	25	30	ns
t_{GHQZ}	t_{DF}	Output Enable to Output High Z (Note 1)		Max	25	30	ns
	t_{OEH}	Output Enable Hold Time (Note 1)	Read	Min	0		ns
			Toggle and Data# Polling	Min	10		ns
t_{AXQX}	t_{OH}	Output Hold Time From Addresses, CE# or OE#, Whichever Occurs First (Note 1)		Min	0		ns

Read Operations

Parameter		Description	Test Setup		Speed Options		Unit
JEDEC	Std				70	90	
t_{AVAV}	t_{RC}	Read Cycle Time (Note 1)		Min	70	90	ns
t_{AVQV}	t_{ACC}	Address to Output Delay	$CE\# = V_{IL}$ $OE\# = V_{IL}$	Max	70	90	ns
t_{ELQV}	t_{CE}	Chip Enable to Output Delay	$OE\# = V_{IL}$	Max	70	90	ns
t_{GLQV}	t_{OE}	Output Enable to Output Delay		Max	30	35	ns
t_{EHQZ}	t_{DF}	Chip Enable to Output High Z (Note 1)		Max	25	30	ns
t_{GHQZ}	t_{DF}	Output Enable to Output High Z (Note 1)		Max	25	30	ns
	t_{OEH}	Output Enable Hold Time (Note 1)	Read	Min	0		ns
			Toggle and Data# Polling	Min	10		ns
t_{AXQX}	t_{OH}	Output Hold Time From Addresses, CE# or OE#, Whichever Occurs First (Note 1)		Min	0		ns

FIGURA 4.27. DATOS DE LOS ACCESOS A MEMORIA S29AL032D.**PROBLEMA 4.24.**

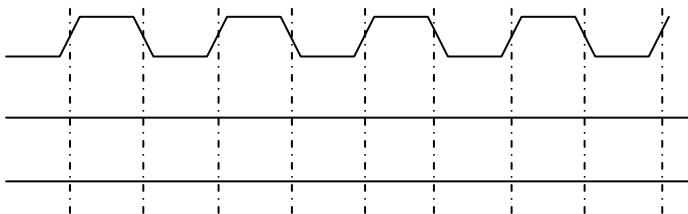
La conexión de un procesador LPC178x que funciona a 100 MHz con una memoria RAM estática (IS62WV25616-70) se realiza a través del EMC de tal manera que todas las líneas se conectan directamente entre procesador y memoria, con excepción de la línea CS# que pasa por un circuito de decodificación y se retrasa en la activación entre 2 y 5 ns; y en la desactivación entre 1 y 2 ns.

Los cronogramas de acceso a escritura del procesador, una vez configurado su funcionamiento, son los indicados en la Figura 4.28 y los datos de los accesos a la memoria los mostrados en la Figura 4.29.

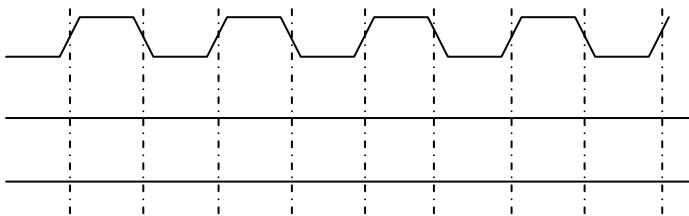
Se pide:

1. Compruebe si se cumplen las condiciones necesarias para que se pueda realizar la escritura desde la aparición de direcciones ($tAW \rightarrow Address\ Setup$

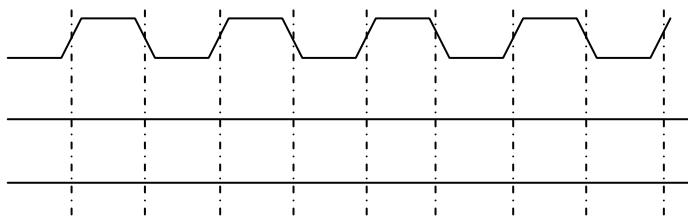
Time to Write End) sin añadir ciclos de espera, y en caso contrario indique cuánto tiempo adicional sería necesario.



2. Compruebe si se cumplen las condiciones necesarias para que se pueda realizar la escritura desde la generación de la señal de CE# ($t_{SCS1} \rightarrow CS1$ to *Write End*) sin añadir ciclos de espera, y en caso contrario indique cuánto tiempo adicional sería necesario.



3. Compruebe si se cumplen las condiciones necesarias para que se pueda realizar la escritura desde la aparición de los datos ($t_{SD} \rightarrow Data\ Setup\ to\ End\ of\ Write$) sin añadir ciclos de espera, y en caso contrario indique cuánto tiempo adicional sería necesario.



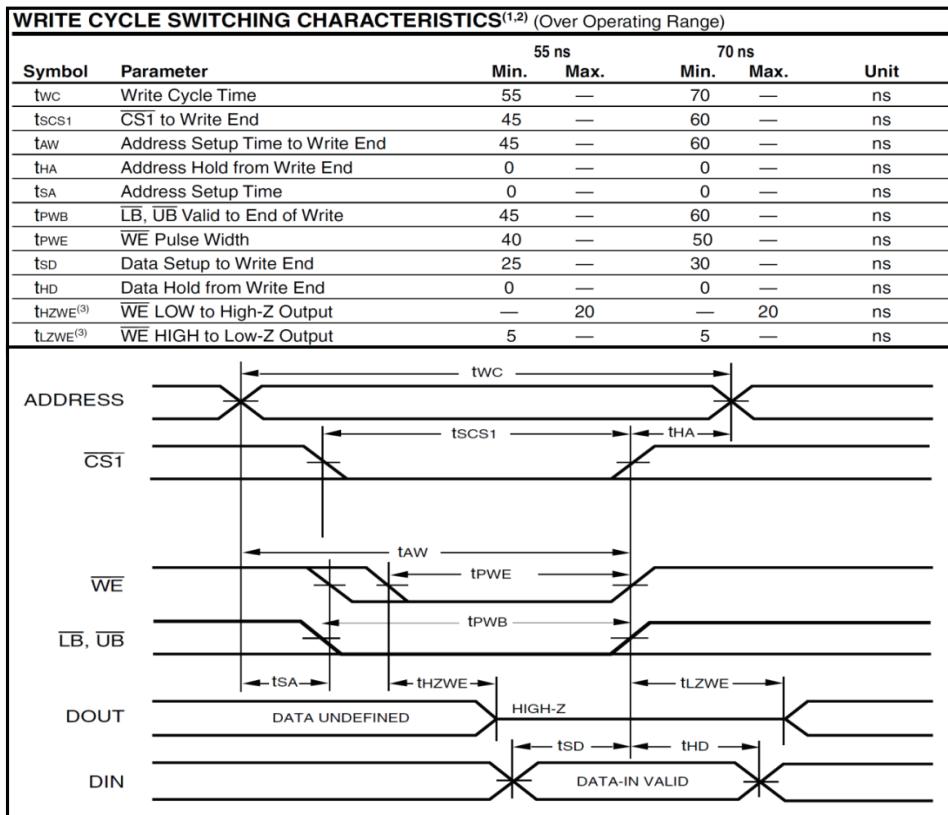


FIGURA 4.28. DATOS DE ACCESO EN ESCRITURA A LA MEMORIA IS62WV25616.

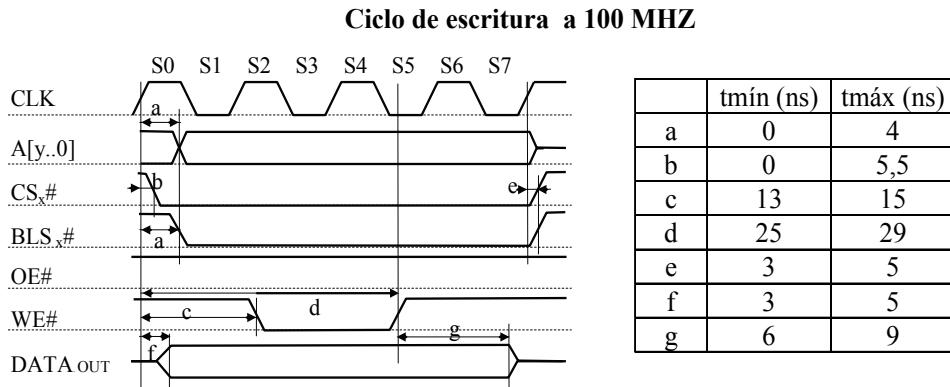


FIGURA 4.29. DATOS DE ACCESO EN ESCRITURA DEL PROCESADOR.

PROBLEMA 4.25.

Un microcontrolador de 32 bits funcionando a 25 MHz dispone, entre otros, de los siguientes terminales de control:

- TIP#: Señal de validación de dirección.
- R/W#: Señal de selección de operación, lectura o escritura.
- BS[3:0]#: Señales de selección de banco.

Este microcontrolador se conecta a una memoria según se muestra en la Figura 4.35. El ciclo de bus en lectura se muestra en la Figura 4.32. En la tabla de tiempos, algunos de ellos se dan referidos a t_{CYC} , que representa el periodo de la señal de reloj CLKOUT.

Se pide:

1. Dibuje en el cronograma de la Figura 4.30 la temporización de las señales que se utilizan para acceder a la memoria en lectura indicando claramente los tiempos y los retardos con que se activan. Calcule también el tiempo de acceso máximo que podría tener la memoria respecto de la señal CS# (t_{CE}) para que no requiera de la inserción de ciclos de espera.

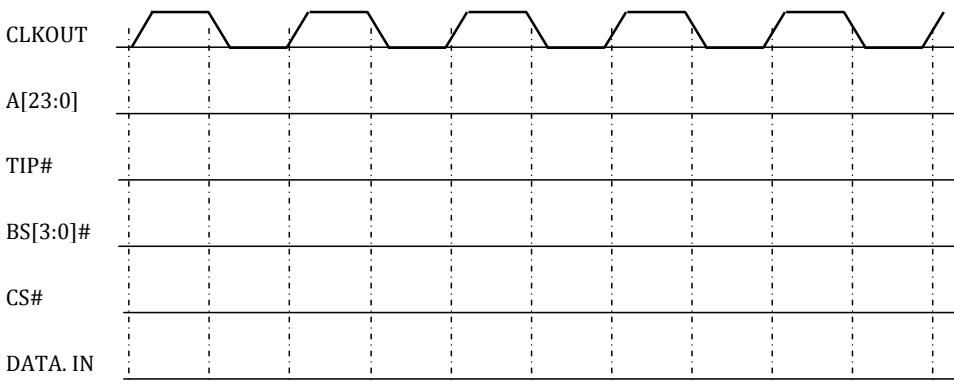


FIGURA 4.30. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

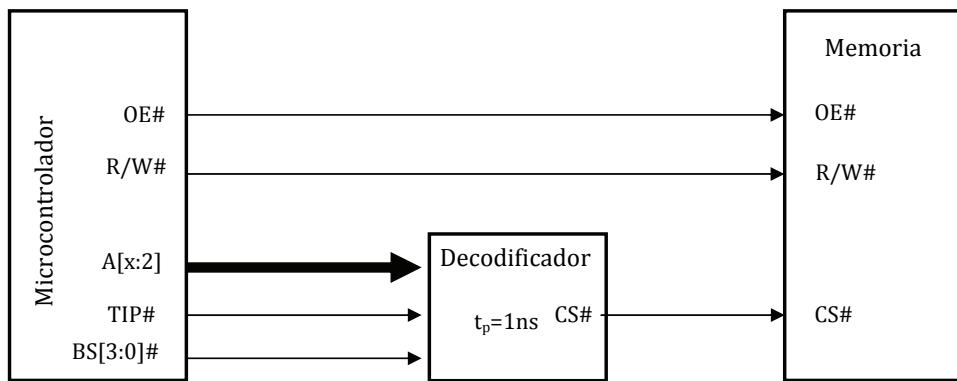
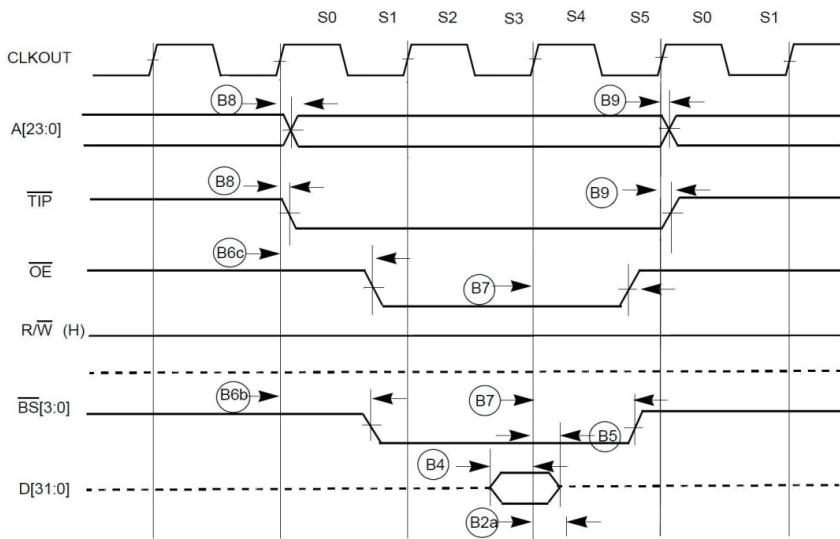


FIGURA 4.31. CONEXIÓN ENTRE MICROCONTROLADOR Y MEMORIA.



Name	Characteristic ¹	Symbol	Min	Max	Unit
Control Inputs					
B1a	Control input valid to CLKOUT high ²	t_{CVCH}	10	—	ns
B1b	\overline{BKPT} valid to CLKOUT high ³	t_{BKVCH}	10	—	ns
B2a	CLKOUT high to control inputs invalid ²	t_{CHCII}	0	—	ns
B2b	CLKOUT high to asynchronous control input \overline{BKPT} invalid ³	t_{BKNCH}	0	—	ns
Data Inputs					
B4	Data input (D[31:0]) valid to CLKOUT high	t_{DIVCH}	6	—	ns
B5	CLKOUT high to data input (D[31:0]) invalid	t_{CHDII}	0	—	ns
Name	Characteristic	Symbol	Min	Max	Unit
Control Outputs					
B6a	CLKOUT high to chip selects valid ¹	t_{CHCV}	—	$0.5t_{CYC} + 10$	ns
B6b	CLKOUT high to byte enables (BS[3:0]) valid ²	t_{CHBV}	—	$0.5t_{CYC} + 10$	ns
B6c	CLKOUT high to output enable (OE) valid ³	t_{CHOV}	—	$0.5t_{CYC} + 10$	ns
B7	CLKOUT high to control output ($\overline{BS}[3:0]$, \overline{OE}) invalid	t_{CHCOI}	$0.5t_{CYC} + 2$	—	ns
B7a	CLKOUT high to chip selects invalid	t_{CHCI}	$0.5t_{CYC} + 2$	—	ns
B8	CLKOUT high to address (A[23:0]) and control (\overline{TS} , $SIZ[1:0]$, \overline{TIP} , R/W) valid	t_{CHAV}	—	10	ns
B9	CLKOUT high to address (A[23:0]) and control (\overline{TS} , $SIZ[1:0]$, \overline{TIP} , R/W) invalid	t_{CHAI}	2	—	ns
Data Outputs					
B11	CLKOUT high to data output (D[31:0]) valid	t_{CHDOV}	—	10	ns
B12	CLKOUT high to data output (D[31:0]) invalid	t_{CHDOI}	2	—	ns
B13	CLKOUT high to data output (D[31:0]) high impedance	t_{CHDOZ}	—	6	ns

FIGURA 4.32. CICLO DE BUS DE LECTURA DEL MICROCONTROLADOR.

PROBLEMA 4.26.

Un microcontrolador de 32 bits funcionando a 50 MHz dispone, entre otros, de los siguientes terminales de control:

- TIP#: Señal de validación de dirección.
 - R/W#: Señal de selección de operación, lectura o escritura.
 - BS[3:0]#: Señales de selección de banco.

Este microcontrolador se conecta a una memoria N04L63W2A-70 alimentada a 3,3V según se muestra en la Figura 4.34. La Figura 4.35 muestra el ciclo de lectura de la memoria y la Figura 4.40 el ciclo de bus en lectura del microprocesador. En la tabla de tiempos, algunos de ellos se dan referidos a t_{CYC} , que representa el periodo de la señal de reloj CLKOUT.

Se pide:

1. Dibuje en el cronograma de la Figura 4.33 la temporización de las señales que se utilizan para acceder a la memoria en lectura indicando claramente los tiempos y los retardos con que se activan.

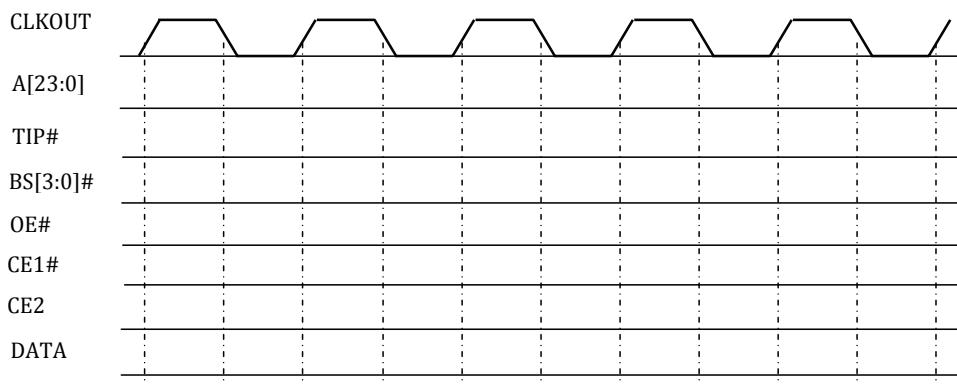


FIGURA 4.33. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

2. Complete la Tabla 4.24 indicando los tiempos que se indican, de selección por parte del microprocesador y de acceso de la memoria. Debajo de la tabla incluya el cálculo de los tiempos de selección que no se obtengan directamente del ciclo de bus en lectura del microprocesador.

TABLA 4.24. VALOR DE LOS TIEMPOS DE ACCESO.

Tiempo	Valor selección (Microprocesador)	Valor acceso (Memoria)
Dirección válida a dato válido		
<i>Chip enable</i> a dato válido		
<i>Output enable</i> a dato válido		
<i>Byte select</i> a dato válido		

Calcule cuántos ciclos de espera son necesario insertar para que la lectura se realice correctamente.

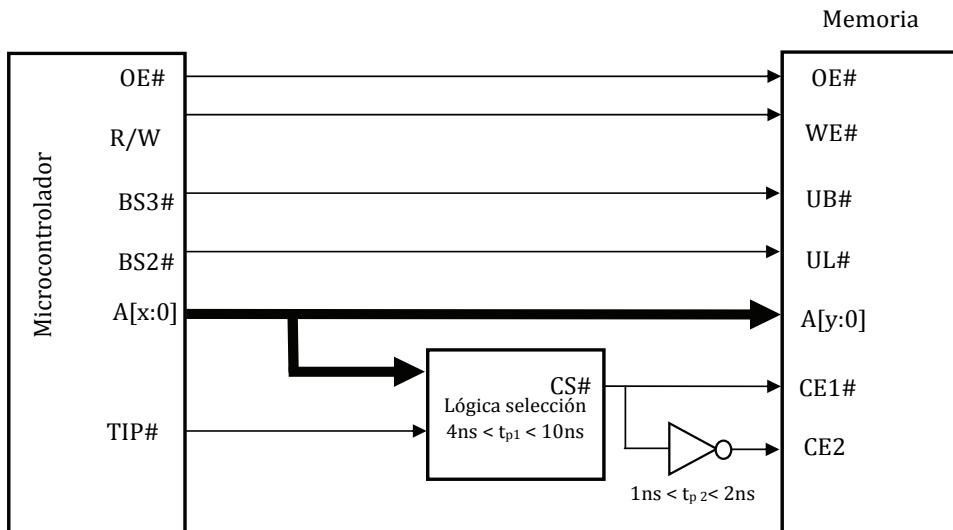
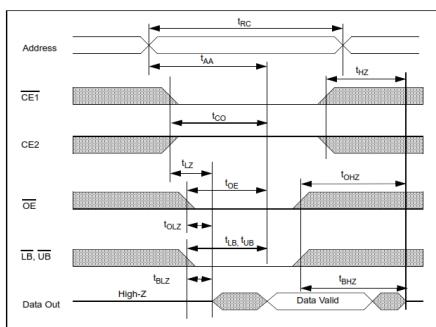
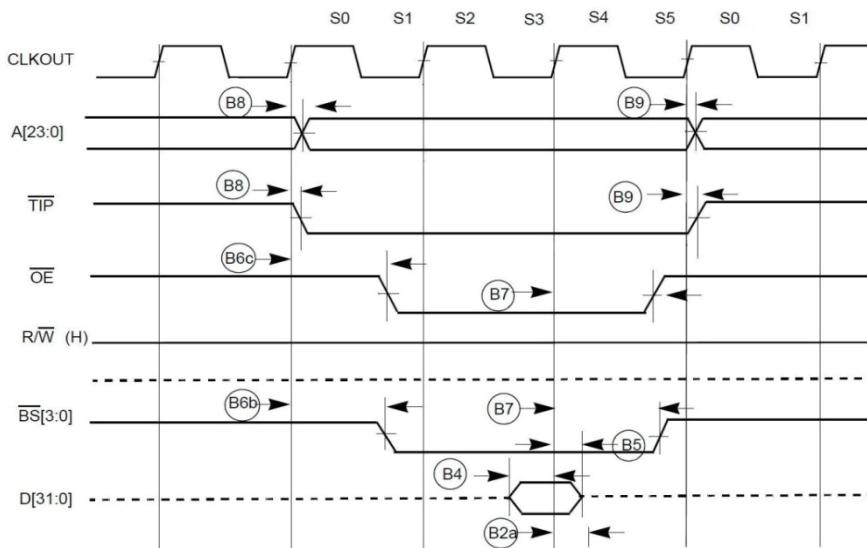


FIGURA 4.34. CONEXIÓN ENTRE MICROCONTROLADOR Y MEMORIA.



Item	Symbol	-70				Units	
		2.3 - 2.65 V		2.7 - 3.6 V			
		Min.	Max.	Min.	Max.		
Read Cycle Time	t_{RC}	85		70		ns	
Address Access Time	t_{AA}			85		ns	
Chip Enable to Valid Output	t_{CO}			85		ns	
Output Enable to Valid Output	t_{OE}			30	25	ns	
Byte Select to Valid Output	t_{LB_UB}			85		ns	
Chip Enable to Low-Z output	t_{LZ}	10		10		ns	
Output Enable to Low-Z Output	t_{OLZ}	5		5		ns	
Byte Select to Low-Z Output	t_{BZ}	10		10		ns	
Chip Disable to High-Z Output	t_{HZ}	0	20	0	20	ns	
Output Disable to High-Z Output	t_{OHZ}	0	20	0	20	ns	
Byte Select Disable to High-Z Output	t_{BHZ}	0	20	0	20	ns	
Output Hold from Address Change	t_{OH}	10		10		ns	

FIGURA 4.35. CICLO DE LECTURA DE LA MEMORIA N04L63W2A-70 Y TABLA DE TIEMPOS.



Name	Characteristic ¹	Symbol	Min	Max	Unit
Control Inputs					
B1a	Control input valid to CLKOUT high ²	t_{CVCH}	10	—	ns
B1b	\overline{BKPT} valid to CLKOUT high ³	t_{BKVCH}	10	—	ns
B2a	CLKOUT high to control inputs invalid ²	t_{CHCII}	0	—	ns
B2b	CLKOUT high to asynchronous control input \overline{BKPT} invalid ³	t_{BKNCH}	0	—	ns
Data Inputs					
B4	Data input (D[31:0]) valid to CLKOUT high	t_{DIVCH}	6	—	ns
B5	CLKOUT high to data input (D[31:0]) invalid	t_{CHDII}	0	—	ns
Name	Characteristic	Symbol	Min	Max	Unit
Control Outputs					
B6a	CLKOUT high to chip selects valid ¹	t_{CHOV}	—	$0.5t_{CYC} + 10$	ns
B6b	CLKOUT high to byte enables ($\overline{BS}[3:0]$) valid ²	t_{CHBV}	—	$0.5t_{CYC} + 10$	ns
B6c	CLKOUT high to output enable (\overline{OE}) valid ³	t_{CHOV}	—	$0.5t_{CYC} + 10$	ns
B7	CLKOUT high to control output ($\overline{BS}[3:0]$, \overline{OE}) invalid	t_{CHCOI}	$0.5t_{CYC} + 2$	—	ns
B7a	CLKOUT high to chip selects invalid	t_{CHCI}	$0.5t_{CYC} + 2$	—	ns
B8	CLKOUT high to address (A[23:0]) and control (\overline{TS} , $SIZ[1:0]$, TIP, R/W) valid	t_{CHAV}	—	10	ns
B9	CLKOUT high to address (A[23:0]) and control (\overline{TS} , $SIZ[1:0]$, TIP, R/W) invalid	t_{CHAI}	2	—	ns
Data Outputs					
B11	CLKOUT high to data output (D[31:0]) valid	t_{CHDOV}	—	10	ns
B12	CLKOUT high to data output (D[31:0]) invalid	t_{CHDOI}	2	—	ns
B13	CLKOUT high to data output (D[31:0]) high impedance	t_{CHDOZ}	—	6	ns

FIGURA 4.36. CICLO DE BUS EN LECTURA DEL MICROPROCESADOR Y TABLA DE TIEMPOS.

PROBLEMA 4.27.

La conexión de un procesador, que funciona a 50Mhz, con memorias RAM del modelo (CY62256-55) es la mostrada en la Figura 4.39. Los cronogramas de acceso a escritura del procesador son los indicados en la Figura 4.40 y los datos de los accesos a la memoria los mostrados en la Tabla 4.26.

- Deduzca razonadamente qué señal controla la operación de escritura, \overline{CE} o \overline{WE} . Apóyese en la Figura 4.47.

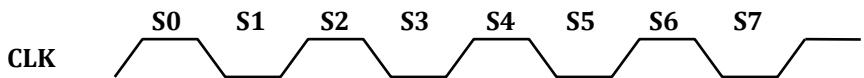


FIGURA 4.37. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

- Asumiendo que la operación de escritura está controlada \overline{WE} , por complete razonadamente la Tabla 4.25 indicando los tiempos de acceso y selección, y deduzca si es necesario insertar ciclos de espera. Dibuje las señales en la Figura 4.38.

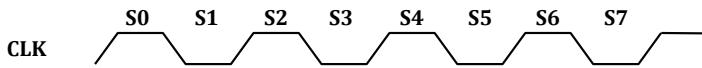


FIGURA 4.38. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

TABLA 4.25. TIEMPO DE ACCESO Y SELECCIÓN.

Tiempo	Valor selección (Microprocesador)	Valor acceso (Memoria)
Dirección válida a fin de escritura		
<i>Chip enable</i> a fin de escritura		
Dato válido a fin de escritura		
Ancho del pulso de escritura		

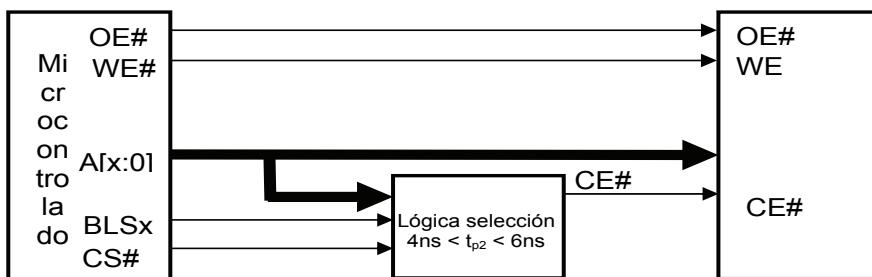
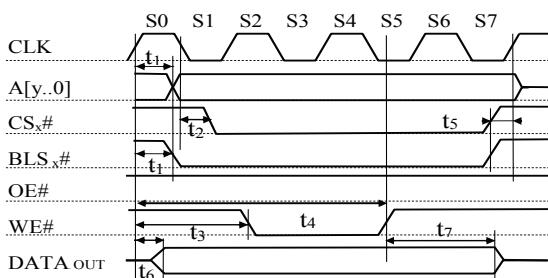


FIGURA 4.39. CONEXIÓN ENTRE MICROCONTROLADOR Y MEMORIA.

TABLA 4.26. TIEMPOS PARA EL CICLO DE ESCRITURA DE LA MEMORIA (CY62256-55)

Parameter	Description	CY62256-55		CY62256-70		Unit
		Min.	Max.	Min.	Max.	
Write Cycle^[10, 11]						
t _{WC}	Write Cycle Time	55		70		ns
t _{SCE}	CE LOW to Write End	45		60		ns
t _{AW}	Address Set-up to Write End	45		60		ns
t _{HA}	Address Hold from Write End	0		0		ns
t _{SA}	Address Set-up to Write Start	0		0		ns
t _{PWE}	WE Pulse Width	40		50		ns
t _{SD}	Data Set-up to Write End	25		30		ns
t _{HD}	Data Hold from Write End	0		0		ns
t _{HZWE}	WE LOW to High-Z ^[8, 9]		20		25	ns
t _{LZWE}	WE HIGH to Low-Z ^[8]	5		5		ns

Ciclo de escritura a 50 Mhz



	t _{mín} (ns)	t _{máx} (ns)
t ₁	0	10
t ₂	0	15
t ₃	25	30
t ₄	55	65
t ₅	2	6
t ₆	8	12
t ₇	20	

FIGURA 4.40. CRONOGRAMAS DE ESCRITURA DEL PROCESADOR.

PROBLEMA 4.28.

Una memoria RAM se conecta a un microprocesador según el esquema de la Figura 4.42. El cronograma de la operación se muestra en la Figura 4.43 y los datos de acceso de la memoria en la Figura 4.44.

1. Dibuje en el cronograma de la Figura 4.41 las señales que se utilizan para obtener \overline{CE} y \overline{OE} a conectar a la memoria (así como estas dos) en las operaciones de lectura, mostrando los retardos que se generen.



FIGURA 4.41. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

2. Deduzca cuáles podrían ser los valores de los retardos tp_1 y tp_2 de la lógica de selección, para que no se incluya más de un ciclo de espera en las operaciones de lectura sobre dicha memoria.

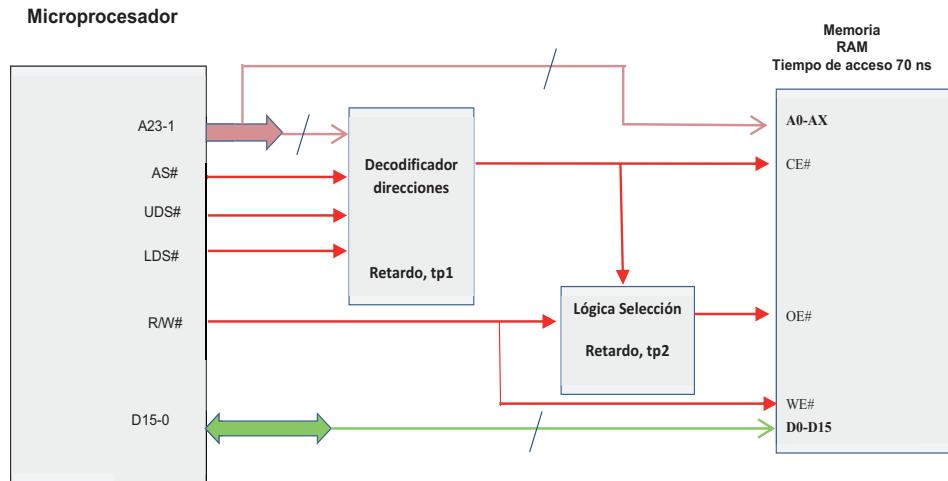


FIGURA 4.42. CONEXIÓN DEL MICROPROCESADOR CON UNA MEMORIA.

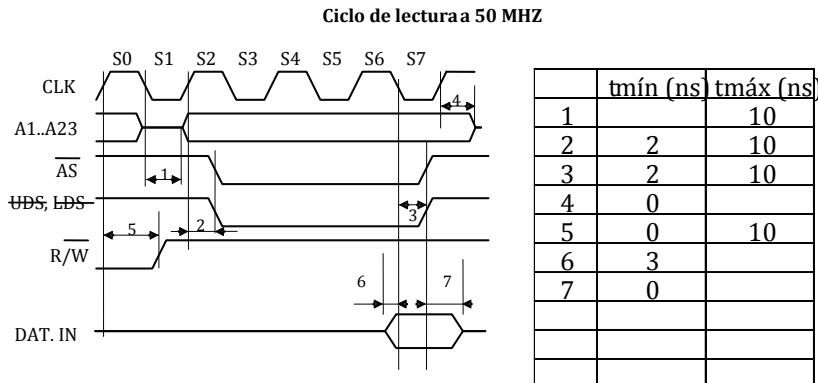


FIGURA 4.43. CRONOGRAMA DE ACCESO EN LECTURA DEL MICROPROCESADOR.

Read Operations			Read Operations		
Parameter	Std	Description	Test Setup		Speed Options
			70	90	
t_{AVAU}	t_{AC}	Read Cycle Time (Note 1)			
t_{AVQV}	t_{Acc}	Address to Output Delay $\text{CE}\# = V_{\text{IL}}$ $\text{OE}\# = V_{\text{HL}}$	Min 70	90	
t_{EQUV}	t_{C}	Chip Enable to Output Delay $\text{OE}\# = V_{\text{IL}}$	Max 70	90	
t_{GQV}	t_{GE}	Output Enable to Output Delay	Max 30	35	
t_{EHQZ}	t_{GZ}	Chip Enable to Output High Z (Note 1)	Max 25	30	
t_{GHQZ}	t_{GZ}	Output Enable to Output High Z (Note 1)	Max 25	30	ns
t_{OHQ}	t_{OH}	Output Enable Hold Time (Note 1)	Read		
t_{OHQ}	t_{OH}	Toggle and Data# Polling	Min 0	ns	
t_{AVQX}	t_{OH}	Output Hold Time From Addresses, CE# or OE#, Whichever Occurs First (Note 1)	Min 0	ns	

Read Operations			Read Operations		
Parameter	Std	Description	Test Setup		Speed Options
			70	90	
t_{AVAU}	t_{AC}	Read Cycle Time (Note 1)			
t_{AVQV}	t_{ACC}	Address to Output Delay $\text{CE}\# = V_{\text{IL}}$ $\text{OE}\# = V_{\text{IL}}$	Min 70	90	ns
t_{EQUV}	t_{C}	Chip Enable to Output Delay $\text{CE}\# = V_{\text{IL}}$ $\text{OE}\# = V_{\text{IL}}$	Max 70	90	ns
t_{GQV}	t_{GE}	Output Enable to Output Delay	Max 30	35	ns
t_{GQZ}	t_{GZ}	Chip Enable to Output High Z (Note 1)	Max 25	30	ns
t_{GHQZ}	t_{GZ}	Output Enable to Output High Z (Note 1)	Max 25	30	ns
t_{OHQ}	t_{OH}	Output Enable Hold Time (Note 1)	Read		
t_{OHQ}	t_{OH}	Toggle and Data# Polling	Min 0	ns	
t_{AVQX}	t_{OH}	Output Hold Time From Addresses, CE# or OE#, Whichever Occurs First (Note 1)	Min 0	ns	

FIGURA 4.44. DATOS DE LOS ACCESOS A MEMORIA.

SECCIÓN 5. PROBLEMAS GLOBALES

PROBLEMA 4.29.

Un microprocesador de 16 bits que trabaja con ordenación de datos *big endian*, dispone, entre otras líneas, de:

- 20 líneas de direcciones $A[19-0]$.
- 1 señal de dirección válida activa a nivel bajo, $DV\#$.
- 1 señal que indica el tamaño del acceso a memoria, SIZ (0, dato de 8 bits; 1, dato de 16 bits).
- Señal indicativa del tipo de operación, $R/W\#$.

Inicialmente se conoce la siguiente información:

- La aplicación (programa de gestión) tiene un tamaño de 60 Kbytes.
 - Se necesita hacer uso de tablas de datos con información de los usuarios, almacenadas de antemano (antes de que el sistema comience a funcionar) en memoria, con un tamaño máximo de 20Kbytes.
 - Trabaja con interrupciones vectorizadas, ocupando la tabla de vectores 512 bytes.
 - La aplicación maneja subrutinas, acepta interrupciones y pasa parámetros a través de memoria. Para el paso de parámetros por memoria se estima un uso máximo de 12Kbytes. Para el resto de operaciones, de lectura/escritura y pila, se necesitarán 20Kbytes.
 - El área que el sistema dedica a entrada/salida (interfaces con periféricos) ocupa un tamaño de 32Kbytes.
1. Indique razonadamente el **número, tipo y tamaño de chips de memoria** que utilizaría para implementar el sistema completo, según las especificaciones dadas. Tenga en cuenta que para reducir el número de chips de memoria puede implementar más capacidad de memoria que la estrictamente necesaria.

El sistema anterior se modifica según las siguientes premisas:

- Se rediseña el sistema de memoria y se decide implementar 128Kbytes de memoria ROM, 16Kbytes para entrada/salida y 256Kbytes para memoria RAM.
- La tabla de vectores debe ubicarse en las direcciones más bajas del mapa de memoria, y a continuación debe ir el código de programa.
- Se desea que la memoria RAM ocupe las últimas direcciones de mapa de memoria que es capaz de gestionar el microprocesador.
- El área dedicada a entrada/salida, se debe mapear antes de la memoria RAM.

- El espacio libre del mapa de memoria debe ser contiguo.
2. Represente gráficamente el mapa funcional del sistema (chips y funcionalidad), indicando las direcciones de comienzo y fin de cada espacio de memoria (ROM, RAM y entrada/salida), la funcionalidad y el tipo de chip utilizado. Debe explicarse claramente el criterio elegido para el diseño del mapa de memoria.
 3. Manteniendo las premisas de ubicación de la memoria indicadas, y haciendo uso de **decodificación completa**, genere las tres líneas de selección para cada bloque: CSROM#, CS_E/S#, CSRAM#.
 4. A partir de las líneas de selección generadas en el apartado anterior, realice la conexión completa de todo el bloque de memoria RAM y el microprocesador (incluya las líneas del microprocesador que sean necesarias. Para el bloque de memoria RAM, use el número de chips adecuado, similares a los de la Figura 4.45).

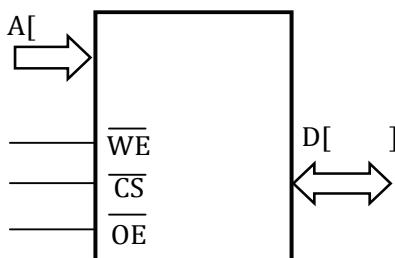


FIGURA 4.45. EJEMPLO DE CHIP DE MEMORIA RAM.

Para la implementación del sistema completo de memoria de contenido permanente (ROM, 128Kbytes), se utiliza la memoria Flash S29AL032D-70, siguiendo la conexión mostrada en la Figura 4.47. Los cronogramas de acceso a lectura del procesador son los indicados en la Figura 4.48 y los datos de los accesos a la memoria los mostrados en la Figura 4.49.

5. A partir de la información suministrada, indique cuál debe ser la correspondencia entre líneas del bus de direcciones del microprocesador y las del bus de direcciones de la memoria (cómo se debe realizar la conexión), de manera que los accesos sean correctos y no exista ambigüedad entre dirección del mapa de memoria y posición del chip de memoria (primera dirección del mapa → primera posición del chip de memoria).
6. Dibuje en el cronograma de la Figura 4.46 las señales que se utilizan para obtener las de CE# y OE# a conectar a la memoria, así como estas dos,

indicando claramente los tiempos y los retardos con que se activan y el tiempo restante hasta la lectura de datos por parte del procesador. Determine también si se podría acceder sin introducir ciclos de espera. En caso de no poderse acceder, indique justificadamente cuántos ciclos de espera se deben introducir.

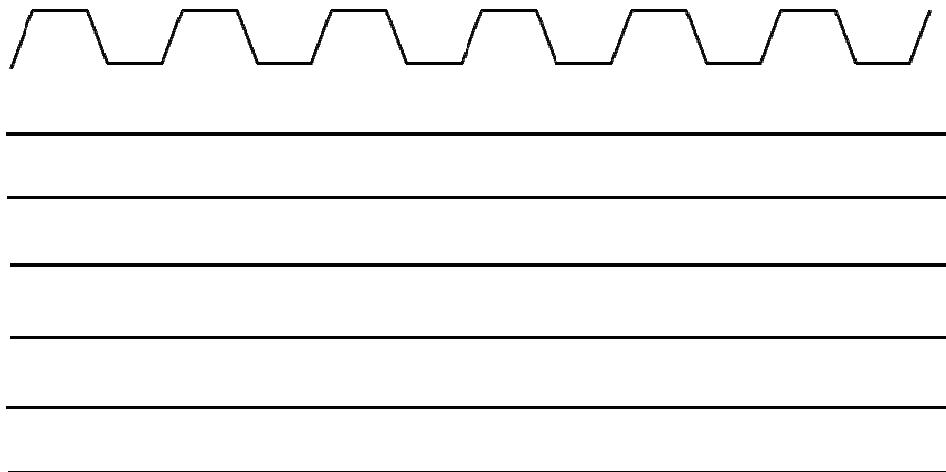


FIGURA 4.46. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

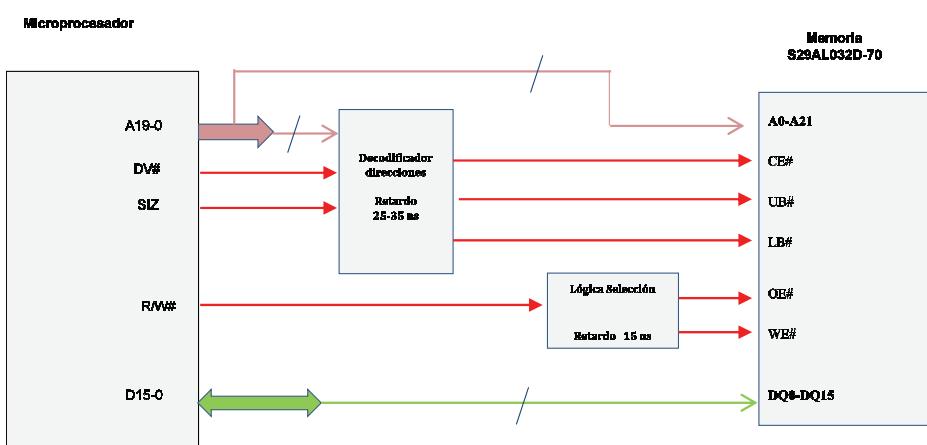


FIGURA 4.47. CONEXIÓN CON UNA MEMORIA FLASH.

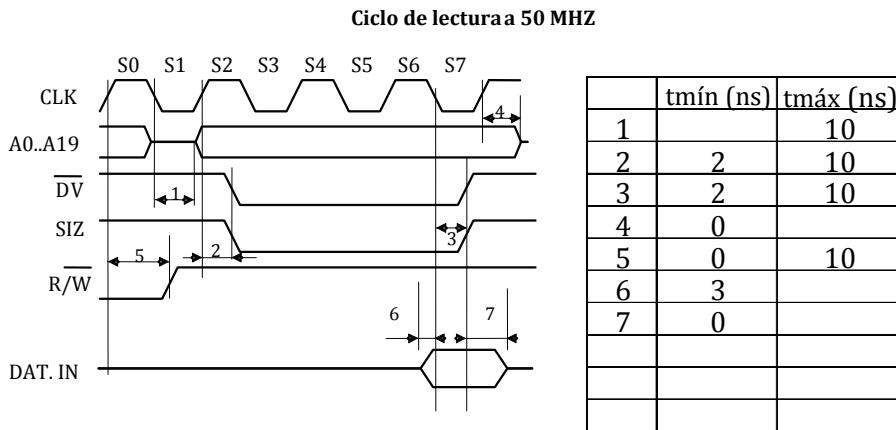


FIGURA 4.48. CRONOGRAMA DE ACCESO EN LECTURA DEL MICROPROCESADOR.

Read Operations

Parameter		Description	Test Setup		Speed Options		Unit
JEDEC	Std			70	90		
t_{AVAV}	t_{RC}	Read Cycle Time (Note 1)		Min	70	90	ns
t_{AVQV}	t_{ACC}	Address to Output Delay	$CE\# = V_{IL}$ $OE\# = V_{IL}$	Max	70	90	ns
t_{ELQV}	t_{CE}	Chip Enable to Output Delay	$OE\# = V_{IL}$	Max	70	90	ns
t_{GLQV}	t_{OE}	Output Enable to Output Delay		Max	30	35	ns
t_{EHQZ}	t_{DF}	Chip Enable to Output High Z (Note 1)		Max	25	30	ns
t_{GHQZ}	t_{DF}	Output Enable to Output High Z (Note 1)		Max	25	30	ns
t_{OEH}	t_{OEH}	Output Enable Hold Time (Note 1)	Read	Min	0		ns
			Toggle and Data# Polling	Min	10		ns
t_{AXQX}	t_{OH}	Output Hold Time From Addresses, CE# or OE#, Whichever Occurs First (Note 1)		Min	0		ns

Read Operations

Parameter		Description	Test Setup		Speed Options		Unit
JEDEC	Std			70	90		
t_{AVAV}	t_{RC}	Read Cycle Time (Note 1)		Min	70	90	ns
t_{AVQV}	t_{ACC}	Address to Output Delay	$CE\# = V_{IL}$ $OE\# = V_{IL}$	Max	70	90	ns
t_{ELQV}	t_{CE}	Chip Enable to Output Delay	$OE\# = V_{IL}$	Max	70	90	ns
t_{GLQV}	t_{OE}	Output Enable to Output Delay		Max	30	35	ns
t_{EHQZ}	t_{DF}	Chip Enable to Output High Z (Note 1)		Max	25	30	ns
t_{GHQZ}	t_{DF}	Output Enable to Output High Z (Note 1)		Max	25	30	ns
t_{OEH}	t_{OEH}	Output Enable Hold Time (Note 1)	Read	Min	0		ns
			Toggle and Data# Polling	Min	10		ns
t_{AXQX}	t_{OH}	Output Hold Time From Addresses, CE# or OE#, Whichever Occurs First (Note 1)		Min	0		ns

FIGURA 4.49. DATOS DE LOS ACCESOS A MEMORIA.

PROBLEMA 4.30.

Un sistema digital basado en un microcontrolador de 32 bits (configurado para acceder a memoria externa con ordenación ***big endian***) dispone, entre otras líneas, de:

- 32 líneas de dirección A[31:0].
 - Señal de validación de dirección activa a nivel bajo, AV#.
 - 2 señales de tamaño de transferencia, SIZ [1:0]: 00 indica tamaño byte, 01 tamaño 16 bits y 10 tamaño 32 bits.
 - 2 señales indicativas del tipo de operación, R# y W# para lectura y escritura respectivamente.
1. A partir de las señales del microcontrolador, se desea generar otra señal ($CS_{ext}\#$) que se active cuando el procesador acceda a cualquier dirección en un rango de 64Mbytes, a partir de la dirección 0x9C00_0000. Realice el diseño del circuito que la genera, con decodificación completa.

2. Para seleccionar los diferentes bytes que componen un “Word” se desean generar las señales $S_3B\#$, $S_2B\#$, $S_1B\#$ y $S_0B\#$, que se corresponden con los pesos de los bytes, de mayor a menor. Diseñe el circuito para generar la señal que seleccione los chips que contengan el byte menos significativo, sabiendo que se trabaja con dato alineados. (Complete la tabla de verdad para la generación de la señal requerida sobre la Tabla 4.27).

TABLA 4.27. TABLA DE VERDAD.

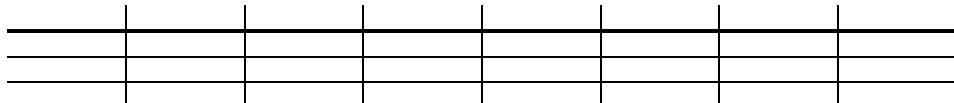
3. A partir de la dirección 0x9C00_0000 se quiere mapear 4 periféricos de 8 bits de 32Kbytes cada uno, y 1 Mbyte de memoria RAM, para lo que se dispone de chips IS62WV1288-55 (ver Figura 4.51 y Figura 4.52).

a) Realice un diagrama de la colocación de los chips necesarios de RAM y Periféricos e indique las direcciones de comienzo y final de cada uno.

Chip	Dirección inicio (hex)	Dirección fin (hex)

TABLA 4.28. DIRECCIONES DE INICIO Y FIN.

- b) Genere las señales de selección globales de RAM y periféricos empleando **decodificación incompleta**. Asuma que además de las señales del micro se dispone de la señal CS_{ext}#.



- c) Realice la conexión entre el chip de memoria con dirección base más baja y el procesador y la conexión entre un periférico con dirección base más alta y el procesador. Asuma que además de las señales del microprocesador también se dispone de las señales CS_{ext}#, S₃B#, S₂B#, S₁B# y S₀B#.
4. Si los datos temporales de acceso en escritura del procesador son los indicados en la Figura 4.53, y el retardo acumulado en los circuitos de generación del CS# de las memorias es de $7 < t_d < 11$ ns (ver esquema de conexión de la Figura 4.54), determine si se puede acceder a las memorias sin introducir estados de espera, comprobando que se cumplen los tiempos necesarios de t_{AW} , t_{SCSI} , t_{PWE} , t_{SD} . Se sugiere que calcule el tiempo desde la activación de la señal AV# hasta que se realiza la escritura, sabiendo que la operación de escritura está controlada por W#. Ayúdese del cronograma de la Figura 4.54 y complete la Tabla 4.29.

(Asuma que el reloj del sistema es de 100MHz y que WAITWR y WAITWEN tienen un valor 0)

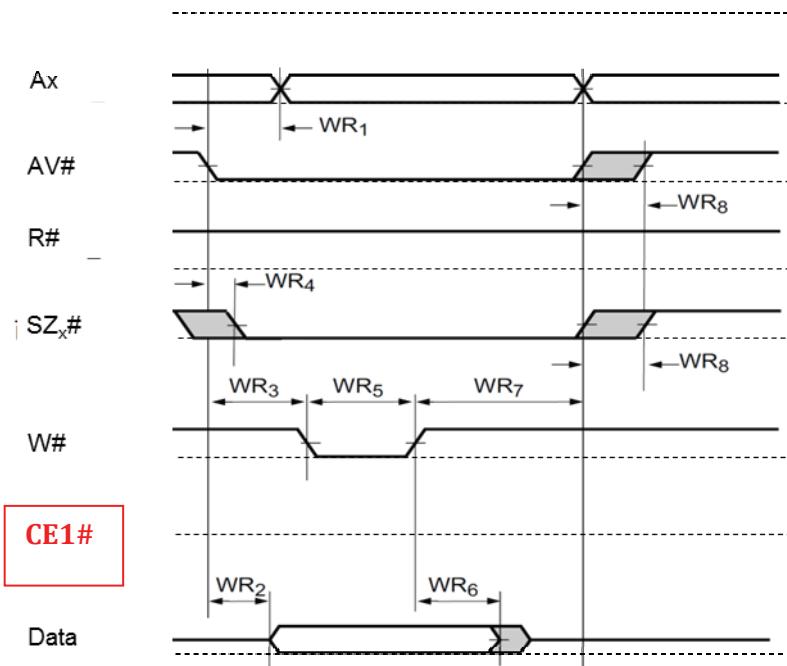


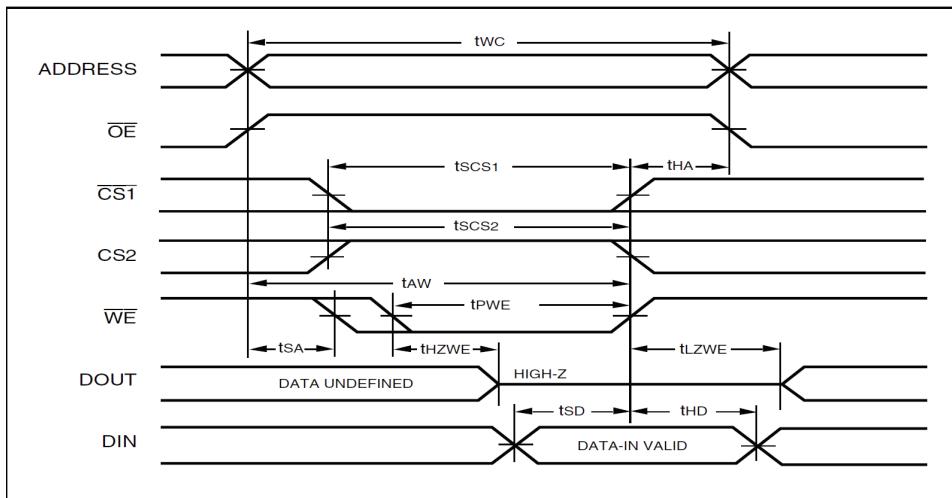
FIGURA 4.50. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

TABLA 4.29. TIEMPOS DE ACCESO Y DE SELECCIÓN.

tiempo	t_{ACC}	t_{Sel}	¿Se cumple? (Si / No)
t_{AW}			
t_{SCSI}			
t_{PWE}			
t_{sp}			

PIN DESCRIPTIONS	
A0-A16	Address Inputs
CS1	Chip Enable 1 Input
CS2	Chip Enable 2 Input
OE	Output Enable Input
WE	Write Enable Input
I/O0-I/O7	Input/Output
NC	No Connection
V _{DD}	Power
GND	Ground

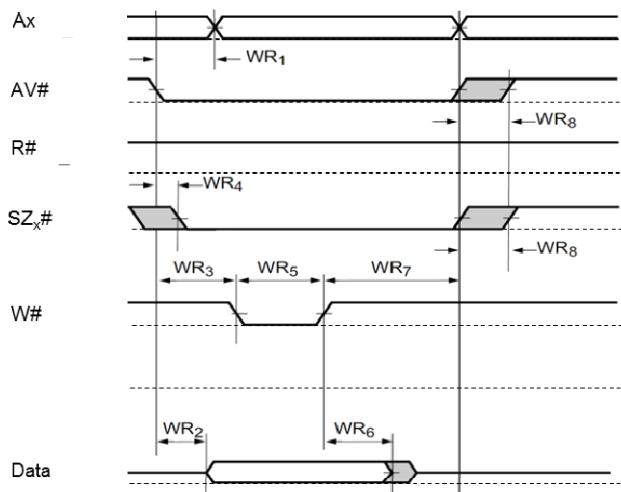
FIGURA 4.51. PINES DE LA MEMORIA IS62WV1288-55

WRITE CYCLE SWITCHING CHARACTERISTICS^(1,2) (Over Operating Range)

Symbol	Parameter	35ns		45ns		55 ns		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
t _{WC}	Write Cycle Time	35	—	45	—	55	—	ns
t _{SCS1/t_{SCS2}}	CS1/CS2 to Write End	25	—	35	—	45	—	ns
t _{AW}	Address Setup Time to Write End	25	—	35	—	45	—	ns
t _{HA}	Address Hold from Write End	0	—	0	—	0	—	ns
t _{SA}	Address Setup Time	0	—	0	—	0	—	ns
t _{PWE}	WE Pulse Width	25	—	35	—	40	—	ns
t _{SD}	Data Setup to Write End	20	—	20	—	25	—	ns
t _{HD}	Data Hold from Write End	0	—	0	—	0	—	ns
t _{HzWE⁽³⁾}	WE LOW to High-Z Output	—	10	—	20	—	20	ns
t _{LZWE⁽³⁾}	WE HIGH to Low-Z Output	3	—	5	—	5	—	ns

FIGURA 4.52. CICLO DE ESCRITURA EN LA MEMORIA IS62WV1288-55

Y TABLA DE TIEMPOS.



Write cycle parameters

Conditions	Min	Typ	Max	Unit
WR1	2.7	3.5	4.7	ns
WR2	2.8	3.9	5.1	ns
WR3	$2.7 + T_{cy}(\text{clk}) \times (1 + \text{WAITWEN})$	$3.5 + T_{cy}(\text{clk}) \times (1 + \text{WAITWEN})$	$4.6 + T_{cy}(\text{clk}) \times (1 + \text{WAITWEN})$	ns
WR4	2.8	3.9	5.1	ns
WR5	$\text{WAITWR} - \text{WAITWEN} + 1) \times T_{cy}(\text{clk}) - 2.3$	$(\text{WAITWR} - \text{WAITWEN} + 1) \times T_{cy}(\text{clk}) - 2.8$	$(\text{WAITWR} - \text{WAITWEN} + 1) \times T_{cy}(\text{clk}) - 3.8$	ns
	$(\text{WAITWR} - \text{WAITWEN} + 3) \times T_{cy}(\text{clk}) - 2.6$	$(\text{WAITWR} - \text{WAITWEN} + 3) \times T_{cy}(\text{clk}) - 3.4$	$(\text{WAITWR} - \text{WAITWEN} + 3) \times T_{cy}(\text{clk}) - 4.9$	ns
WR6	$2.5 + T_{cy}(\text{clk})$	$3.3 + T_{cy}(\text{clk})$	$4.3 + T_{cy}(\text{clk})$	ns
WR7	$T_{cy}(\text{clk}) - 2.7$	$T_{cy}(\text{clk}) - 3.4$	$T_{cy}(\text{clk}) - 4.6$	ns
	2.7	3.6	4.8	ns
	$2.4 + T_{cy}(\text{clk})$	$3.0 + T_{cy}(\text{clk})$	$3.9 + T_{cy}(\text{clk})$	ns
WR8	-2.7	-3.4	-4.7	ns

FIGURA 4.53. CRONOGRAMAS Y TEMPORIZACIÓN DEL CICLO DE ESCRITURA DEL PROCESADOR.

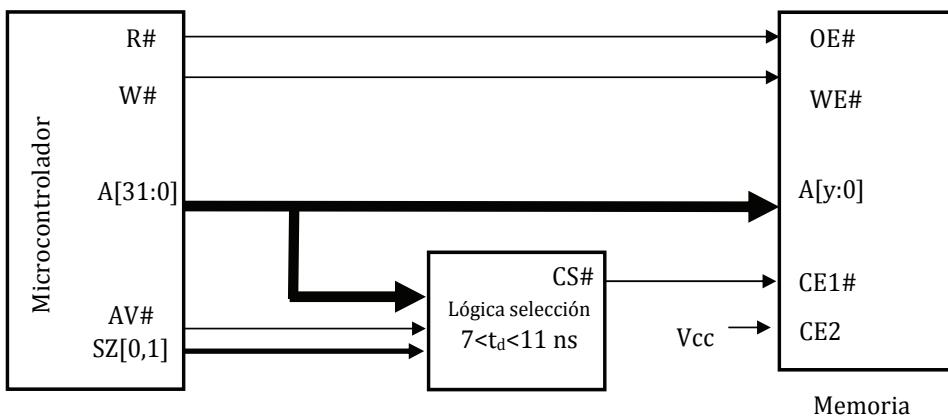


FIGURA 4.54. CONEXIÓN ENTRE MICROCONTROLADOR Y MEMORIA.

PROBLEMA 4.31.

Un microprocesador de 16 bits que trabaja con ordenación de datos *big endian*, dispone, entre otras líneas, de:

- 23 líneas de direcciones $A[23-1]$.
- 1 señal de dirección válida activa a nivel bajo, $AS\#$.
- Señal indicativa del tipo de operación, $R/W\#$.
- 2 señales que indican el tamaño del acceso a memoria y el banco que se activa: UDS# y LDS#; con el funcionamiento indicado en Tabla 4.30.

TABLA 4.30. DESCRIPCIÓN DE LAS SEÑALES UDS# Y LDS#.

#UDS	#LDS	Tamaño, Banco
1	1	No hay acceso a memoria
0	1	Tamaño byte, banco par ($2N$)
1	0	Tamaño byte, banco impar ($2N+1$)
0	0	Tamaño 16bits, ambos bancos

1. Observe que no existe la línea A0 en el bus de direcciones. Describa razonadamente cómo se realiza la funcionalidad de dicha línea. Calcule el tamaño del espacio de direccionamiento del microprocesador e indique también su organización.

Inicialmente se conoce la siguiente información:

- La aplicación (programa de gestión) tiene un tamaño de 30 Kbytes.
 - Se necesita hacer uso de tablas de datos con información de los usuarios, almacenadas de antemano (antes de que el sistema comience a funcionar) en memoria, con un tamaño máximo de 20Kbytes.
 - Trabaja con interrupciones vectorizadas, ocupando la tabla de vectores 1Kbyte.
2. La aplicación maneja subrutinas, acepta interrupciones y pasa parámetros a través de memoria. Para el paso de parámetros por memoria se estima un uso máximo de 12Kbytes. Para el resto de operaciones, de lectura/escritura y pila, se necesitarán 20Kbytes.

Indique razonadamente el **número, tipo y tamaño de chips de memoria** que utilizaría para implementar el sistema completo, según las especificaciones dadas. Tenga en cuenta que para reducir el número de chips de memoria puede implementar más capacidad de memoria que la estrictamente necesaria.

El sistema se modifica siguiendo las siguientes premisas:

- Se rediseña el sistema de memoria y se decide usar únicamente dos áreas, de 64Kbytes cada una, al inicio y al final del espacio de memoria del microprocesador (64kbytes en las direcciones más bajas, y 64kbytes en las más altas). Únicamente se van a utilizar chips para implementar los siguientes bloques:
 - 64Kbytes de memoria ROM.
 - 16Kbytes para entrada/salida.
 - 32Kbytes para memoria RAM.
 - El resto se reserva para ampliación de RAM.
- La tabla de vectores debe ubicarse en las direcciones más bajas del mapa de memoria, y a continuación debe ir el código de programa.
- Se desea que la memoria RAM implementada ocupe las últimas direcciones de mapa de memoria que es capaz de gestionar el microprocesador.

- El área dedicada a entrada/salida, se debe mapear antes de la zona de RAM (la que se implementa y la que se reserva).
3. Represente gráficamente el mapa funcional del sistema (chips y funcionalidad), indicando las direcciones de comienzo y fin de cada espacio de memoria (ROM, RAM y entrada/salida), la funcionalidad y el tipo de chip utilizado.
 4. Manteniendo las premisas de ubicación de la memoria indicadas, y haciendo uso de **decodificación incompleta**, genere las cuatro líneas de selección para cada bloque: CSROM#, CS_E/S#, CSRAM#, CSRAM_AMP#.
 5. A partir de las líneas de selección generadas en el apartado anterior, realice la conexión completa de todo el bloque de memoria RAM implementada y el microprocesador (incluya las líneas del microprocesador que sean necesarias). Para el bloque de memoria RAM, use el número de chips adecuado, similares a los de la Figura 4.55, teniendo en cuenta que únicamente se dispone de chips de 8Kbytes. Etiquete los buses de datos y direcciones de las memorias.

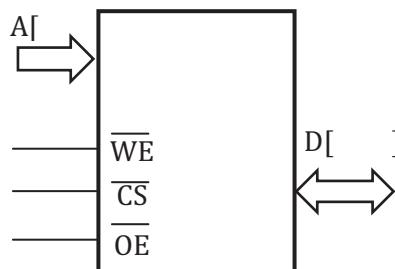


FIGURA 4.55. EJEMPLO DE CHIP DE MEMORIA RAM.

Para la implementación del sistema completo de memoria de contenido permanente (ROM, 64Kbytes), se utiliza la memoria Flash S29AL032D-70, siguiendo la conexión mostrada en la Figura 4.57. Los cronogramas de acceso a lectura del procesador son los indicados en la Figura 4.58 y los datos de los accesos a la memoria los mostrados en la Figura 4.59.

6. A partir de la información suministrada, indique cuál debe ser la correspondencia entre líneas del bus de direcciones del microprocesador y las del bus de direcciones de la memoria (cómo se debe realizar la conexión), de manera que los accesos sean correctos y no exista ambigüedad entre direc-

ción del mapa de memoria y posición del chip de memoria (primera dirección del mapa® primera posición del chip de memoria).

7. Dibuje en el cronograma de la Figura 4.56 las señales que se utilizan para obtener las de CE# y OE# a conectar a la memoria, así como estas dos, indicando claramente los tiempos y los retardos con que se activan y el tiempo restante hasta la lectura de datos por parte del procesador. Determine también si se podría acceder sin introducir ciclos de espera. En caso de no poderse acceder, indique justificadamente cuántos ciclos de espera se deben introducir.

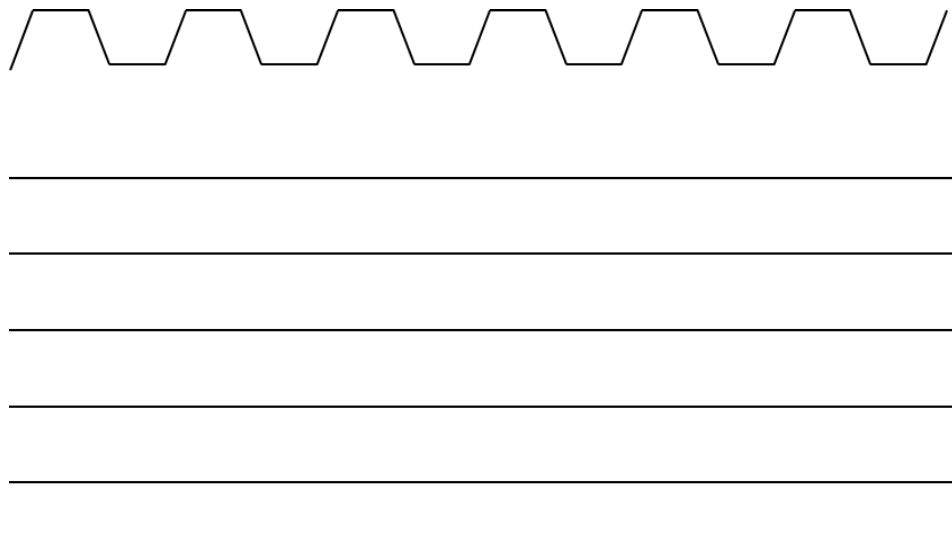


FIGURA 4.56. CRONOGRAMA PARA LA RESOLUCIÓN DEL EJERCICIO.

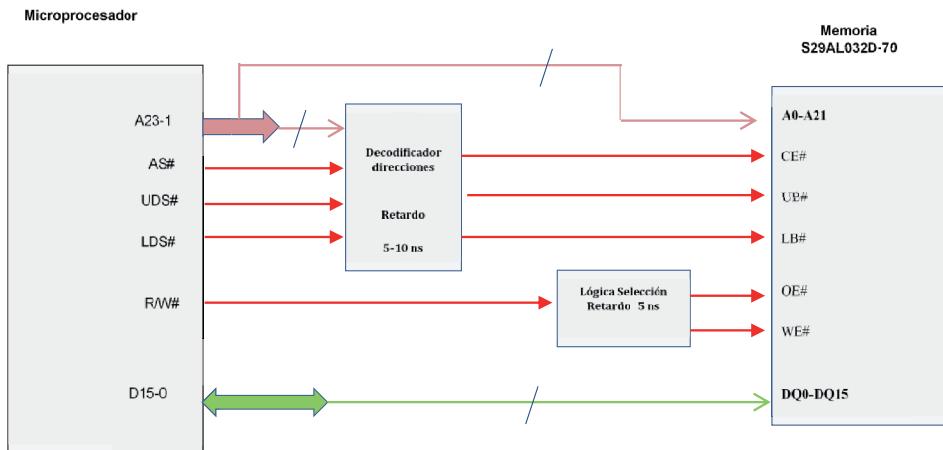


FIGURA 4.57. CONEXIÓN CON UNA MEMORIA FLASH.

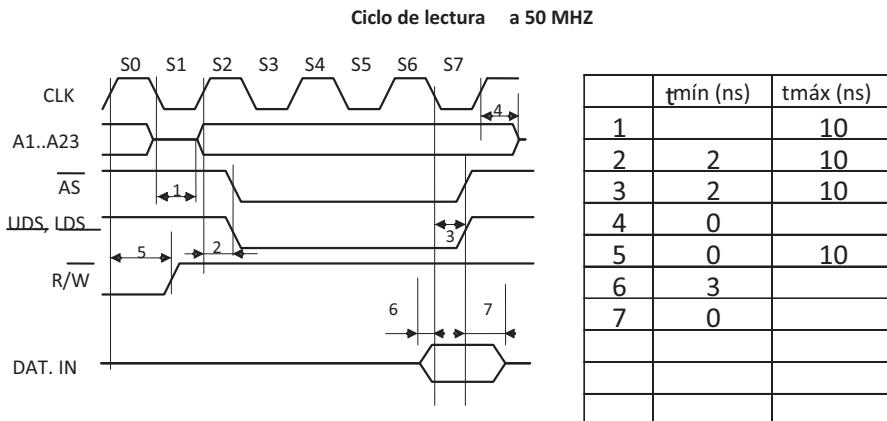
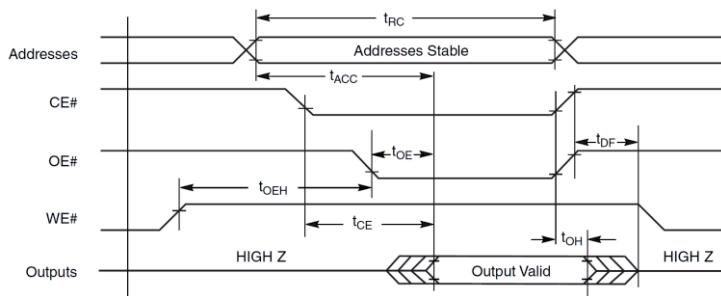


FIGURA 4.58. CRONOGRAMA DE ACCESO EN LECTURA DEL MICROPROCESADOR.



Read Operations

Parameter		Description	Test Setup		Speed Options		Unit
JEDEC	Std			Min	70	90	
t_{AVAV}	t_{RC}	Read Cycle Time (Note 1)			70	90	ns
t_{AVQV}	t_{ACC}	Address to Output Delay	$CE\# = V_{IL}$ $OE\# = V_{IL}$	Max	70	90	ns
t_{ELQV}	t_{CE}	Chip Enable to Output Delay	$OE\# = V_{IL}$	Max	70	90	ns
t_{GLQV}	t_{OE}	Output Enable to Output Delay		Max	30	35	ns
t_{EHQZ}	t_{DF}	Chip Enable to Output High Z (Note 1)		Max	25	30	ns
t_{GHQZ}	t_{DF}	Output Enable to Output High Z (Note 1)		Max	25	30	ns
t_{AXQX}	t_{OH}	Output Enable Hold Time From Addresses, CE# or OE#, Whichever Occurs First (Note 1)	Read	Min	0		ns
			Toggle and Data# Polling	Min	10		
t_{AXQX}	t_{OH}			Min	0		ns

FIGURA 4.59. DATOS DE LOS ACCESOS A MEMORIA.

APÉNDICE A. SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 1

SOLUCIÓN AL PROBLEMA 1.1.

En la Tabla A.1, se presenta la solución al Problema 1.1.

TABLA A.1. CUESTIONES.

	Preguntas	V
1	La depuración sobre el dispositivo: <ul style="list-style-type: none">• Permite visualizar contenidos de memoria y registros	X
2	Los microprocesadores RISC <ul style="list-style-type: none">• Las instrucciones son sencillas	X
3	La segmentación: <ul style="list-style-type: none">• Posibilita que no sea necesario terminar una instrucción para comenzar otra	X
4	Los sistemas empotrados: <ul style="list-style-type: none">• Realizan varias tareas de forma concurrente• Se diseñan para que estén continuamente funcionando	X

SOLUCIÓN AL PROBLEMA 1.2.

En la Tabla A.2, se muestran las soluciones al Problema 1.2.

TABLA A.2. PREGUNTAS DE TEST.

Preguntas		V
1.	Un sistema empotrado para un juguete básico debe cumplir las características de:	
	• Bajo coste	X
	• Bajo consumo	X
2	Los microprocesadores:	
	• Requieren de memoria externa	X
3	Las instrucciones que ejecuta un microprocesador:	
	• Se pueden ampliar con combinaciones de operaciones básicas	X
4	Un microcontrolador generalmente contiene en su encapsulado:	
	• Una CPU	X
	• Memoria de datos	X
	• Periféricos	X
5	Los sistemas digitales:	
	• Leen información del exterior por medio de interfaces de entrada	X
	• Procesan datos de entrada para generar señales de salida	X

SOLUCIÓN AL PROBLEMA 1.3.

En la Tabla A.3 se muestran las soluciones al Problema 1.3.

TABLA A.3. PREGUNTAS DE TEST.

	Preguntas	V
1.	La capacidad de direccionamiento de un microprocesador depende de:	
	• El número de líneas del bus de direcciones	X
2	Algunas ventajas de la utilización de subrutinas en la programación son:	
	• Facilitar la depuración	X
	• Estructurar el programa	X
	• Evitar que el programa sea demasiado largo	X
3	La pila:	
	• Todas las respuestas son incorrectas	X
4	En un procesador segmentado de N etapas el rendimiento:	
	• Se multiplica por N	X

SOLUCIÓN AL PROBLEMA 1.4.

En la Tabla A.4, se muestran las soluciones al Problema 1.4.

TABLA A.4. PREGUNTAS DE TEST.

	Preguntas	V
1.	En un microprocesador de 24 líneas de direcciones:	
	• El número de posiciones a las que se puede acceder es 16M	X
2	El registro contador de programa	
	• Indica dónde se encuentra la siguiente instrucción a ejecutar	X
3	La comunicación entre una unidad periférica y el microprocesador, cuando es iniciada a petición del primero se denomina:	
	• Por interrupción	X
4	Las subrutinas:	
	• Permiten estructurar el programa	X

• Evitan tener que codificar repetidas veces una tarea	X
• Equivalen a las funciones en un lenguaje de alto nivel	X

Solución al Problema 1.5.

En la Tabla A.5 se muestran las soluciones al Problema 1.5.

TABLA A.5. PREGUNTAS DE TEST.

Preguntas		V
1.	La pila:	
	• Todas las respuestas son incorrectas	X
4	Las instrucciones se almacenan en la memoria de programa en formato:	
	• Binario	X

SOLUCIÓN AL PROBLEMA 1.6.

1. En la Tabla A.6 se muestra la solución al Problema 1.6.1. Se ha asumido que los switches cuando están activos ponen en su salida un nivel alto (V_{CC}) y cuando no lo están su salida está “al aire”.

TABLA A.6. CONFIGURACIÓN DE PINES DEL PUERTO P0.

PIN	P0.15	P0.14	P0.13	P0.12	P0.11	P0.10	P0.9	P0.8	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
Entrada	X	X	X	X	X	X	X	X	X							
Salida										X	X	X	X	X	X	X
Pull-up/down	Pull-down	-	-	-	-	-	-	-								

2. En la Tabla A.7 se muestra la solución al Problema 1.6.2.

TABLA A.7. CONFIGURACIÓN DE PINES DEL PUERTO P0.

PIN	P0.15	P0.14	P0.13	P0.12	P0.11	P0.10	P0.9	P0.8	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
Registro de salida	-	-	-	-	-	-	-	-	-	1	1	0	0	1	1	0

3. A continuación, se muestra la solución al Problema 1.6.3.

Generar una Tabla que comience en $DIR+1$ (*hasta DIR+9*) con los bits a activar para presentar los números del 1 al 9. En $DIR+x$ está el código para presentar el nº x

Leer el puerto P0

Desplazar el puerto P0.7 posiciones a la derecha

Detectar el nº del bit Y que está a nivel alto

Escribir en P0 el valor de la posición de la Tabla DIR+Y

SOLUCIÓN AL PROBLEMA 1.7.

- En la Tabla A.8 se muestran las soluciones al Problema 1.7.1.

TABLA A.7. CONFIGURACIÓN DE PINES Px.y.

PIN	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	P0.0
Entrada									X
Salida		X	X	X	X	X	X	X	
Pull-up	-		-	-	-	-	-	-	X

No sería necesario haber puesto la resistencia R_S

- En la Figura A.1. se muestra la solución al Problema 1.7.2.

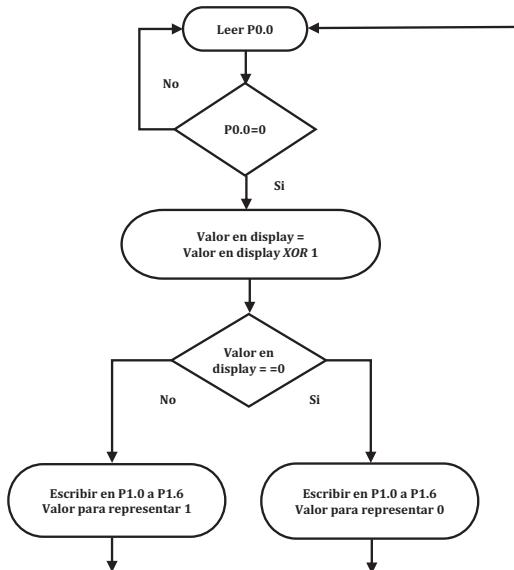


FIGURA A.1. DIAGRAMA DE FLUJO.

SOLUCIÓN AL PROBLEMA 1.8.

1. En la Tabla A.9 se muestran las soluciones al Problema 1.8.1.

TABLA A.9. CONFIGURACIÓN DE PINES DEL PUERTO P.

PIN	P7	P6	P5	P4	P3	P2	P1	P0
Entrada	-	-	-	-	X	X	X	X
Salida	X	X	X	X	-	-	-	-
Pull-up	-	-	-	-	X	X	X	X

2. A continuación se muestra la solución al Problema 1.8.2.

LDR R0,=Puerto_P

LDR R1, [R0]

AND R1,#0x0F

3. A continuación se muestra la solución al Problema 1.8.3.

ADD R1, #3

4. A continuación se muestra la solución al Problema 1.8.4.

LSL R1, #4

LDR R0,=Puerto_P

LDR R2, [R0]

AND R2, #0xFF00

ADD R2, R1

STR R2, [R0]

SOLUCIÓN AL PROBLEMA 1.9.

1. En la Figura A.2 se muestran la solución al Problema 1.9.

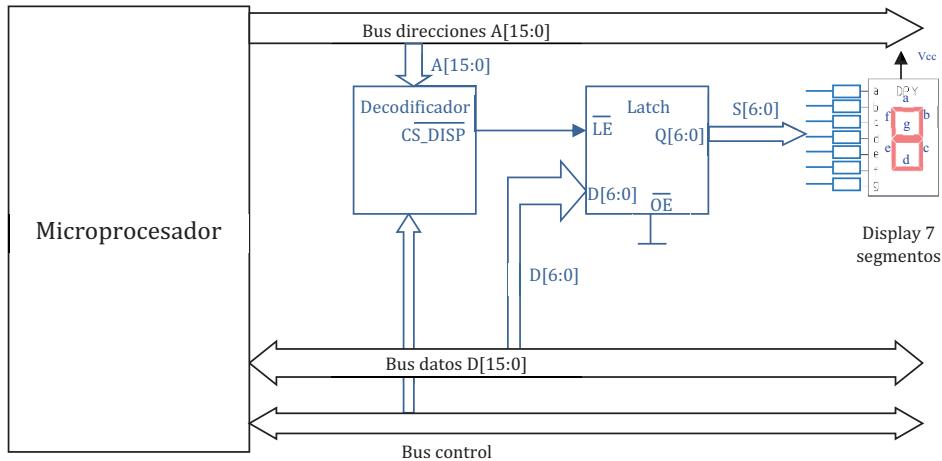


FIGURA A.2. DIAGRAMA DE BLOQUES DE CONEXIONES. SOLUCIÓN AL PROBLEMA 1.9.1

Se ha asumido un display de 7 segmentos de ánodo común. Es necesario incluir un latch para conectar las entradas del display de 7 segmentos al bus de datos. Ese latch almacenará el dato que se desea visualizar en el display. Se debe mapear en memoria, como si de una posición de esta se tratase, para lo cual hay que decodificar el bus de direcciones y generar la señal de carga del latch.

SOLUCIÓN AL PROBLEMA 1.10.

1. En la Figura A.3 se muestran la solución al Problema 1.10.1.

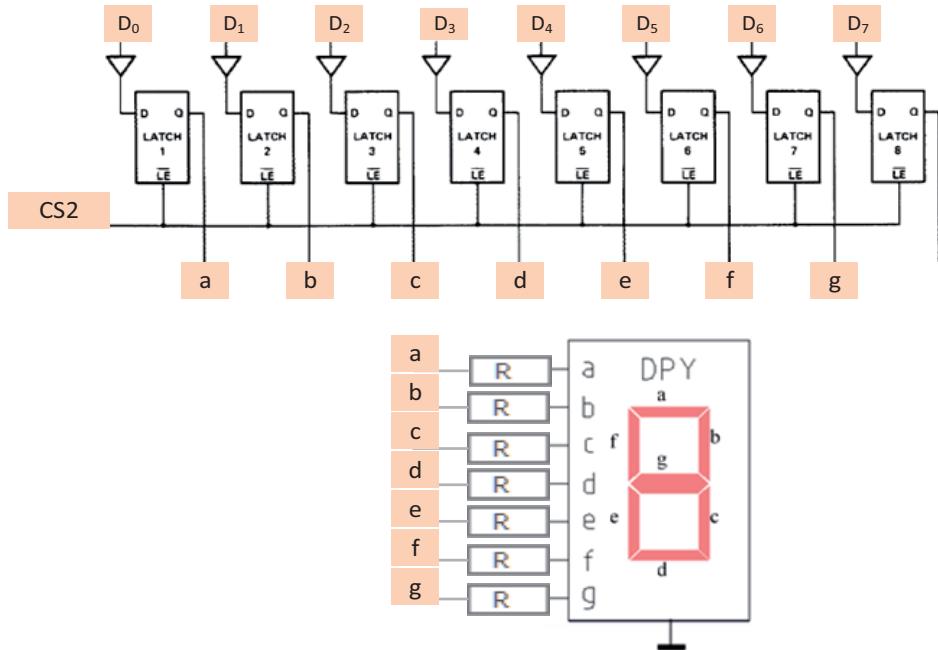


FIGURA A.3. DIAGRAMA DE BLOQUES DE CONEXIONES DE LA SOLUCIÓN AL PROBLEMA 1.10.1

2. En la Figura A.4 se muestran la solución al Problema 1.10.2.

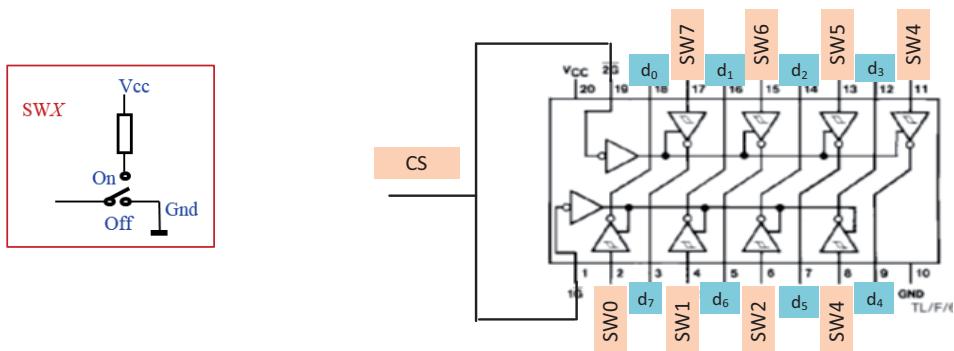


FIGURA A.4. DIAGRAMA DE BLOQUES DE CONEXIONES DE LA SOLUCIÓN AL PROBLEMA 1.10.2

Téngase en cuenta que el dato en dx será el dato en SWx# cuando CS sea un nivel bajo (L)

3. Solución al Problema 1.10.3.

LEER [0x1000], R0
ESCRIBIR R0, [0x2000]

4. Solución al Problema 1.10.4.

Número 2 → segmentos *a, b, d, e, g* iluminados
 segmentos *c, f* apagados

Nótese que los *buffers* son inversores y que los leds del *display* se iluminan con un nivel alto en la entrada correspondiente (cátodo común).

Por tanto *SW0, SW1, SW3, SW4, SW6* en OFF
 SW2, SW5, en ON

SOLUCIÓN AL PROBLEMA 1.11.

1. Solución al Problema 1.11.1.

LOAD, STORE → Instrucciones de transferencia
ADD, SUB, DIV → Instrucciones aritméticas
OR → Instrucción lógica
Modos de direccionamiento: Directo a registro (p.e. R1), directo a memoria (p.e. 0x2004) e inmediato (#0x03)

2. En la Tabla A.10 se muestran las soluciones al Problema 1.11.2.

TABLA A.10. SOLUCIÓN AL PROBLEMA 1.11.2.

1^a Fase	<i>El valor del PC se coloca en el bus de direcciones y se selecciona la dirección 400. La memoria de programa se selecciona y pone la primera instrucción (parte de ella) en el bus de datos. La instrucción llega al registro de instrucciones. Se detecta que esa instrucción aún no ha llegado completamente (tiene 4 Bytes)</i>	<i>1^a transf. (2bytes)</i>
2^a Fase	Se coloca el valor del PC en el bus de direcciones, 0x402, y se accede a memoria de programa a por el resto de la instrucción, el valor 0x2004.	<i>2^a transf. (2bytes)</i>
3^a Fase	Se ejecuta la instrucción, colocándose en el bus de direcciones la posición donde se encuentra el operando, 0x2004. Se accede a la memoria de datos y se lee por el bus de datos el valor almacenado en esa dirección, cargándose en el registro R1. Finaliza la instrucción y comienza la ejecución de la siguiente	<i>3^a transf. (2bytes)</i>

3. En la Tabla A.11 se muestran las soluciones al Problema 1.11.3.

TABLA A.11. SOLUCIÓN AL PROBLEMA 1.11.3.

1^a Fase	Por ejemplo, para la instrucción ADD, se coloca en el bus de direcciones el valor del PC, 0x40C. Se lee la instrucción completa que solo ocupa 2 bytes.	<i>1^a transf. (2bytes)</i>
2^a Fase	Se decodifica la instrucción y se ejecuta. No hay transferencias por el bus de datos porque los operandos y el resultado se almacenan en registros internos de la CPU	No hay transferencias

4. En la Tabla A.12 se muestran las soluciones al Problema 1.11.4.

TABLA A.12. SOLUCIÓN AL PROBLEMA 1.11.4.

1^a Fase	El valor del PC se coloca en el bus de direcciones y se selecciona la dirección 0x414. La memoria de programa se selecciona y pone la primera instrucción (parte de ella) en el bus de datos. La instrucción llega al registro de instrucciones. Se detecta que esa instrucción aún no ha llegado completamente (tiene 4 Bytes)	<i>1^a transf. (2bytes)</i>
2^a Fase	Se coloca el valor del PC en el bus de direcciones, 0x416, y se accede a memoria de programa a por el resto de la instrucción, el valor 0x2010.	<i>2^a transf. (2bytes)</i>
3^a Fase	Se ejecuta la instrucción, colocándose en el bus de direcciones la posición donde se debe almacenar el operando almacenado en R5, 0x2010. Se accede a la memoria de datos y se escribe el valor almacenado en R5 en la dirección 0x2010. Finaliza la instrucción y comienza la ejecución de la siguiente	<i>3^a transf. (2bytes)</i>
	Para ejecutar la instrucción se han realizado 3 transferencias	

5. En la Tabla A.13 se muestran las soluciones al Problema 1.11.5.

TABLA A.13. SOLUCIÓN AL PROBLEMA 1.11.5.

1^a Fase	El valor del PC se coloca en el bus de direcciones y se accede a la memoria de programa para leer la instrucción (parte de ella) en el bus de datos. La instrucción llega al registro de instrucciones. Se detecta que esa instrucción aún no ha llegado completamente (tiene 8 Bytes)	1 ^a transf. (2bytes)
2^a Fase	Se realizan tres transferencias más para leer el resto de la instrucción, que son las direcciones donde se encuentran los operandos, 0x2004 y 0x2008, y la dirección donde almacenar el resultado, 0x2014	2 ^a , 3 ^a y 4 ^a transf. (6bytes)
3^a Fase	Se accede a la posición de memoria 0x2004 a por el primer operando	5 ^a transf. (2bytes)
4^a Fase	Se accede a la posición de memoria 0x2008 a por el segundo operando	6 ^a transf. (2bytes)
5^a Fase	Se ejecuta la instrucción, realizándose la suma	No hay transf. (0bytes)
6^a Fase	Se accede a la posición de memoria 0x2014 y se almacena el resultado de la operación	7 ^a transf. (2bytes)

6. Solución al Problema 1.11.6

Para las instrucciones 1, 2 y 4:

Número de transferencias: 4 para leer instrucción + 4 leer operandos + 2 almacenar resultado en memoria = 10

Para la instrucción 3: 4 para leer instrucción + 2 leer operandos + 2 almacenar resultado en memoria = 8

Total transferencias = $3 \times 10 + 8 = 38$ transferencias

SOLUCIÓN AL PROBLEMA 1.12.

En la Figura A.5. se muestra la solución al Problema 1.12.1

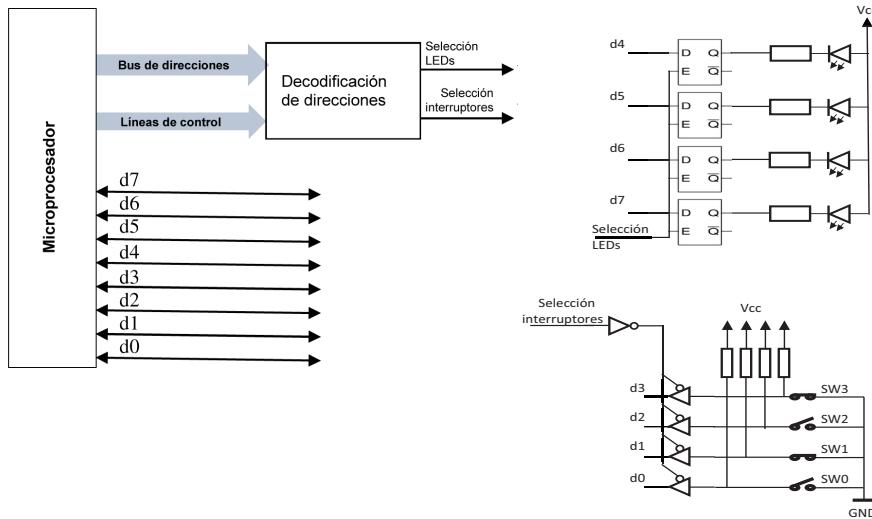


FIGURA A.5. DIAGRAMA DE BLOQUES DE CONEXIONES DE LA SOLUCIÓN AL PROBLEMA 1.12.1.

En la Figura A.6. se muestra la solución al Problema 1.12.2.

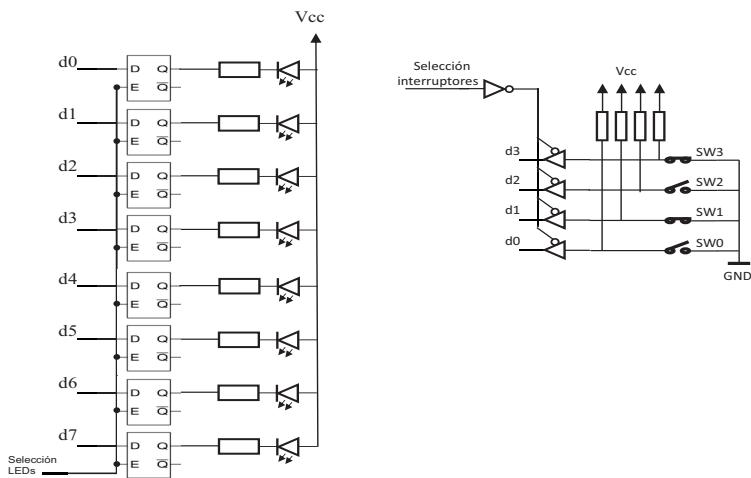


FIGURA A.6. DIAGRAMA DE BLOQUES DE CONEXIONES DE LA SOLUCIÓN AL PROBLEMA 1.12.2.

3. Solución al Problema 1.12.3.

El ejercicio se resuelve asumiendo el circuito del apartado 1 o 2. En la Figura A.7. se muestra la solución al Problema 1.12.3.

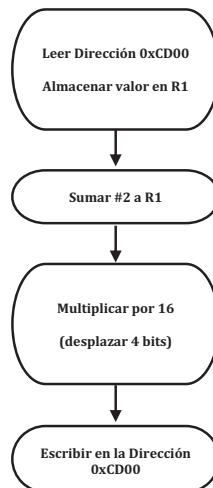


FIGURA A.7. DIAGRAMA DE FLUJO DE LAS OPERACIONES A REALIZAR.

5. Solución al Problema 1.12.4.

LEER [0xCD00], R1
SUMAR #2, R1
MULTIPLICAR #0x10, R1
ESCRIBIR R1, [0xCD00]

6. Solución al Problema 1.12.5.

Cambiaría la última instrucción

ESCRIBIR R1, [0xCC00]

APÉNDICE B. SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 2

SOLUCIÓN AL PROBLEMA 2.1.

En la Tabla B.1, se muestran las soluciones al Problema 2.1.

TABLA B.1 SOLUCIÓN AL PROBLEMA 2.1

<i>MOV r0, #0x20 MSR BASEPRI, R0</i>	Configura la máscara de interrupciones a valor 0x20. (Solamente se permiten interrupciones con prioridad mayor del 0x20 -prioridad de -2 a 0x1F)
<i>MOV r0, #0x01 MSR FAULTMASK, R0</i>	Se enmascaran todas las excepciones excepto la NMI
<i>MOV r0, #0x02 MSR CONTROL, R0</i>	En modo thread se utilizará la pila PSP. (Se configura el nivel a “privilegiado”)
<i>MOV r0, #0x01 MSR CONTROL, R0</i>	En modo thread se utilizará la pila MSP. Se configura el nivel a “usuario”
<i>MOV r0, #0x02 MSR CONTROL, R0</i>	Se intenta cambiar de nuevo a nivel “privilegiado” pero no se realiza, ya que estamos en modo usuario

SOLUCIÓN AL PROBLEMA 2.2.

En la Tabla B.1 se muestra la solución al Problema 2.2.

TABLA B.1. SOLUCIÓN AL PROBLEMA 2.2

Línea	Registro/Dirección de memoria y valor
8	R0 = 0x1000:0000
9	R1 = 0x1000:2000
10	R1 = 0x0000:0000
11	LR = 0x0000:0018
12	R0 = 0x1000:0018
16	[0x1000:3FFC]=0x1800:0000 [0x1000:3FF8]=0x0000:0000 [0x1000:3FF4]=0x0000:0010 SP=0x1000:3FF4
17	R1 = 0x0000:0001
18	[0x1000:0000]=0x0100:0000 R0 = 0x1000:0004
19	R2 = 0x0020:0000
20	R3 = 0x0000:3344
21	R4 = 0x0000:5566
24	[0x1000:0004]=0x11EE:0000 [0x1000:0008]=0x0300:0000 R0=0X1000:000C
25	[0x1000:000C]=0x0000:2000 [0x1000:0010]=0X4433:0000 [0x1000:0014]=0x6655:0000
26	R0 = 0x1000:0000 R1 = 0x0000:0000 PC = 0x0000:0018 SP = 0x1000:4000

SOLUCIÓN AL PROBLEMA 2.3.

En la Tabla B.3, se muestra la solución al Problema 2.3.

TABLA B.3. SITUACIÓN INICIAL DE MEMORIA Y REGISTROS PARA EL PROBLEMA 2.3

Dirección	<i>Little endian</i>					<i>Registros</i>	
	Memoria antes de la ejecución		Mem. después de la ejecución			R0	R1
0x000010000		FFFFAAAA	0005 A000
	A2	03	00	01		EEEE1111	0000 FODE
0x000010004	FF	00	88	11		DDDD2222	
	00	00	12	34		CCCC4444	
	56	78	9A	BC		BBBB5555	0000 3412
	DE	F0	12	34		AAAA6666	0000 XXXX
		000FF000	0005 A000
	00	01	22	33		5555AAAA	
	44	55	66	77		00000000	0001 0012
	FF	00	AA	11		00000001	0000 0002
	76	54	32	10		00010000	0001 000C
	11	11	11	11		00000004	0000 0016
	00	01	22	33		00000000	
	44	55	66	77		SP	30001000
0x300010000	FF	00	AA	11		LR	000003FC
	76	54	32	10		PC	

Dirección	<i>Big endian</i>					<i>Registros</i>	
	Memoria antes de la ejecución		Mem. después de la ejecución			R0	R1
0x000010000		FFFFAAAA	0005 A000
0x000010004	A2	03	00	01		EEEE1111	0000 DEF0
	FF	00	88	11		DDDD2222	
	00	00	12	34		CCCC4444	
	56	78	9A	BC		BBBB5555	0000 1234
	DE	F0	12	34		AAAA6666	0000 XXXX
		000FF000	0005 A000
	00	01	22	33		5555AAAA	
	44	55	66	77		00000000	0001 0012
	FF	00	AA	11		00000001	0000 0002
	76	54	32	10		00010000	0001 000C
	11	11	11	11		00000004	0000 0016
	00	01	22	33		00000000	
	44	55	66	77		SP	30001000
0x300010000	FF	00	AA	11		LR	000003FC
	76	54	32	10		PC	

SOLUCIÓN AL PROBLEMA 2.4.

1. En la Tabla B.4 se muestra la solución al Problema 2.4.1

TABLA B.4. SOLUCIÓN AL PROBLEMA 2.4.1

Fila	Contenido
1	PUSH {R4, R2, LR}
2	LDR R0,=0x10000000
3	LDMDB R0!, {R4,R2}
4	MOVT R4,# 0x01AC
5	ADD R2, #0X0C
6	STMIA R0, {R2}
7	POP {R4,R2,PC}
8	

2. En la Tabla B.5 se muestra la solución al Problema 2.4.2

TABLA B.5. SOLUCIÓN AL PROBLEMA 2.4.2.

Línea de código	Registro o Dir. memoria	Contenido (Hexadecimal)			
1	0X10002000	F9	FF	FF	FF
	0X10001FFC	BB	AA	99	88
	0X10001FF8	33	22	11	00
	R13	10	00	1F	F8
2	R0	10	00	00	00
3	R4	33	DD	22	CC
	R2	11	BB	00	AA
	R0	0F	FF	FF	F8
4	R4	01	AC	22	CC
5	R2	11	BB	00	B6
6	0X0FFFFFF8	B6	00	BB	11

SOLUCIÓN AL PROBLEMA 2.5.

1. En la Tabla B.6 se muestra la solución al Problema 2.5.1

TABLA B.6. SOLUCIÓN AL PROBLEMA 2.5.1.

	Registros	Valor (hex)
MOV R2,#5	R2	0X05
MOV R3,#0x04	R3	0X04
SUB R4,R3,R2	R4	0xFFFFFFFF
ADDLE R3,R4	R3	0X03
NOP		

2. En la Tabla B.7 se muestra la solución al Problema 2.5.2

TABLA B.7. SOLUCIÓN AL PROBLEMA 2.5.2.

		Registro/Dir memoria	Valores
1	STACK_TOP EQU 0x10000000		
2	AREA RESET, CODE	DDC	
3			
4	STACK_TOP	DDC	
5	DCD Reset_Handler		
6			
7	ENTRY		
8			
9			
10	Reset_Handler		
11	LDR R0,=0x10004000	R0	0X10004000
12	LDR R1,=0x10002000	R1	0X10002000
13			
14	MOV R2,#0x22		
15	MOVW R3,#0x3344	R3	0X00003344
16	MOVT R3,#0x55AA	R3	0X55AA3344
17	MOV R4,#0x5566	R4	0X00005566
18	MOV R5,#0xAABB		
19	MOV R6,#0xCCDD		
20	MOV R7,#0xEE11		
21	MOV R8,#0X03		
22	LDR R9,=0X10001000		
23			
24	MSR PSP, R9	PSP	0X10001000
25	MSR CONTROL, R8	CONTROL	03 (% 11 en binario)
26			TODOS LOS DATOS EN HEXADECIMAL
27			
28	STMIA R1, {R2-R4}	0x10002000- x1000200B	22000000 / 4433AA55 / 66550000
29	STMIA R0!, {R5-R7}	0X10004000- 0X1000400B //R0	BBA0000 / DDCC0000 / 11EE0000 // 1000400C
30	LDMDB R0, {R2-R4}	R2/R3/R4	0000AABB / 0000CCDD / 0000EE11
31	LDMDB R1!, {R5-R7}	R5/R6/R7 //R1	00000000 / 00000000 / 00000000 // 10001FF4
32			
33	PUSH {R5,R3,R1,R4,R2}	0X10000FEC- 0X10000FFF PSP	F41F0010 / BBA0000 / DDCC0000 / 11EE0000 / 00000000 0x10000FEC
34	POP {R1,R5,R3,R2,R4}	PSP Y REGISTROS	Quedan igual que antes de ejecutarse (33)
35	NOP		
36			
37	AREA Datos,DATA		
38	ALIGN 4		
39			
40	END		

3. En la Tabla B.8 se muestra la solución al Problema 2.5.3

TABLA B.8. SOLUCIÓN AL PROBLEMA 2.5.3.

		Registro/Dir memoria	Valores
1	STACK_TOP EQU 0x10000000		
2	AREA RESET, CODE		
3		DDC	
4	STACK_TOP		
5	DCD Reset_Handler		
6			
7			
8	ENTRY		
9			
10	Reset_Handler		
11	LDR R0,=0x10003000	R0	0X10004000
12	LDR R1,=0x10001000	R1	0X10002000
13			
14	MOV R2,#0x5566		
15	MOVW R3,#0x3344	R3	0X00003344
16	MOVT R3,#0x55AA	R3	0X55AA3344
17	MOV R4,#0x22	R4	0X00005566
18	MOV R5,#0x1122		
19	MOV R6,#0x3344		
20	MOV R7,#0xEE11		
21	MOV R8,#0X03		
22	LDR R9,=0X10002000		
23			
24	MSR PSP, R9	PSP	0X10001000
25	MSR CONTROL, R8	CONTROL	03 (% 11 en binario)
26			TODOS LOS DATOS EN HEXADECIMAL
27			
28	STMIA R1, {R2-R4}	0x10002000- x1000200B	22000000 / 4433AA55 / 66550000
29	STMIA R0!, {R5-R7}	0X10004000- 0X1000400B //R0	BAA0000 / DDCC0000 / 11EE0000 // 1000400C
30	LDMDB R0, {R2-R4}	R2/R3/R4	0000AABB / 0000CCDD / 0000EE11
31	LDMDB R1!, {R5-R7}	R5/R6/R7 //R1	00000000 / 00000000 / 00000000 // 10001FF4
32			
33	PUSH {R5,R3,R1,R4,R2}	0X10000FEC- 0X10000FFF PSP	F41F0010 / BAA0000 / DDCC0000 / 11EE0000 / 00000000 0x10000FEC
34	POP {R1,R5,R3,R2,R4}	PSP Y REGISTROS	Quedan igual que antes de ejecutarse (33)
35	NOP		
36			
37	AREA Datos,DATA		
38	ALIGN 4		
39	END		
40			

SOLUCIÓN AL PROBLEMA 2.6.

1. En la Tabla B.9 se muestra la solución al Problema 2.6.1

TABLA B.9 SOLUCIÓN AL PROBLEMA 2.6.1.

Línea del código	Registro o Dir. memoria	Contenido (Hexadecimal)			
244	R0	10	00	60	00
245	PSP	10	00	60	00
246	R1	10	00	00	00

2. En la Tabla B.10 se muestra la solución al Problema 2.6.2

TABLA B.10. SOLUCIÓN AL PROBLEMA 2.6.2.

Línea del código	Registro o Dir. de memoria	Contenido (Hexadecimal)			
247	R4	33	DD	22	CC
	R3	11	BB	00	AA
	R2	EF	CD	AB	89
	R1	0F	FF	FF	F4
248	R1	10	00	00	00
249	1000:0000 h	33	DD	22	CC
	1000:0004 h	55	FF	44	EE
250	1000:1FFC h	33	DD	22	CC
	1000:1FF8 h	AA	00	BB	11
	R13	10	00	1F	F8

3 .En la Tabla B.11 se muestra la solución al Problema 2.6.3

TABLA B.11. SOLUCIÓN AL PROBLEMA 2.6.3.

Línea del código	Nueva situación
251	Se enmascaran todas las interrupciones excepto NMI
252	
253	Se enmascaran todas las interrupciones con prioridad igual o inferior a 4 (binario 100). Pero están enmascaradas todas excepto NMI
254	
255	Se pasa a modo no privilegiado y con pila PSP
256	

4. Solución al Problema 2.6.4.

PUSH {R6-R9, R11, LR}

POP {R6-R9, R11, PC}

SOLUCIÓN AL PROBLEMA 2.7.

1. Solución al Problema 2.7.1.

Utilizar STMDB R13!, como PUSH es correcto,

Pero POP no recupera R4 de la pila, por lo que en PC se cargará una dirección de vuelta de la rutina errónea

2. Solución al Problema 2.7.2.

LDMIA R1!, {R2-R5}

por

LDMIA R1, {R2-R5}

ADD R1, #16

3. En la Tabla B.12 se muestra la solución al Problema 2.7.3

TABLA B.12. SOLUCIÓN AL PROBLEMA 2.7.3.

Línea de código	Registro o Dir. memoria	Contenido (Hexadecimal)			
3	R1	10	00	20	10
7	0x1000:2015	89			
	R1	10	00	20	16

SOLUCIÓN AL PROBLEMA 2.8.

1. Solución al Problema 2.8.1.

calcular = 0x00000138, ya que fin = 0x00000134 y todas las instrucciones son de 32 bits

Del mismo modo, el tamaño de la rutina calcular es 8 instrucciones x 4 bytes/instrucción = 32 bytes

2. En la Tabla B.13 se muestra la solución al Problema 2.8.2

TABLA B.13. SOLUCIÓN AL PROBLEMA 2.8.2.

Tabla B.2. Instrucciones, registros y valores modificados para el Problema 2.8.2.		
Fila	Instrucción	Registros modificados y valores
1	LDR R0,= esperando	R0=0x10000008
2	LDR R1, [R0]	R1=0x00000001
11	PUSH {R1}	SP(R13)=0x10001FFC
12	BL calcular	LR=0x00000130 / PC=0x00000138

3. En la Tabla B.14 se muestra la solución al Problema 2.8.3

TABLA B.14. SOLUCIÓN AL PROBLEMA 2.8.3.

Dirección	Valor	Notas
0x10002000	XX XX XX XX	
0x10001FFC	0x00000000	R2=0x00000000
0x10001FF8	0xA5005A00	R1=0xA5005A00
0x10001FF4	XXXXXXXX	
SP termina apuntando a 0x10001FF8		

5. En la Tabla B.15 se muestra la solución al Problema 2.8.4

TABLA B.15. SOLUCIÓN AL PROBLEMA 2.8.4.

Dirección	Valor
0x10000000	0xA5005A00
0x10000004	0x5AFFA600
0x10000008	0x00000000

5. Solución al Problema 2.8.5.

Hace el complemento a 2 del dato recibido en entrada y que se le pasa como parámetro por la pila. El resultado del complemento a 2 lo almacena en salida

SOLUCIÓN AL PROBLEMA 2.9.

1. En la Tabla B.16 se muestra la solución al Problema 2.9.1

TABLA B.16. SOLUCIÓN AL PROBLEMA 2.9.1.

Dirección	Dirección	Contenido			
		4N	4N+1	4N+2	4N+3
	0x1000_3FE4	07	00	00	00
	0x1000_3FE8	90	00	00	00
	0x1000_3FEC	00	00	00	10
	0x1000_3FF0	77	66	55	44
	0x1000_3FF4	41	00	00	00
	0x1000_3FF8	90	00	00	00
	0x1000_3FFC	39	00	00	00
	0x1000_4000	---	---	---	---

R0
R1
R2
R3
LR
R1
LR

2. En la Tabla B.17 se muestra la solución al Problema 2.9.2

TABLA B.17. SOLUCIÓN AL PROBLEMA 2.9.2.

Dirección	Contenido			
	4N	4N+1	4N+2	4N+3
0x1000_0000	31	24	19	09
0x1000_0004	51	04	01	---

SOLUCIÓN AL PROBLEMA 2.10.

6. Solución al Problema 2.10.1

La pila empleada es la apuntada por el registro PSP, y la última dirección es la 0x1000_3FFF

7. Solución al Problema 2.10.2.

Hay 1 llamada a rutina; la llamada está en la dirección 0x412 y la rutina empieza en la dirección 0x422.

- ## 8. Solución al Problema 2.10.3.

Se pasan por la pila. Se introducen los parámetros con la instrucción de la dirección 0x410, y se devuelve el resultado por registro, en el registro R2.

9. En la Tabla B.18 se muestra la solución al Problema 2.10.4.

TABLA B.18. SOLUCIÓN AL PROBLEMA 2.10.4.

Dirección	Contenido			
	4N	4N+1	4N+2	4N+3
	0x1000_3FF4	00	00	00
	0x1000_3FF8	00	20	00
	0x1000_3FFC	17	04	00

10. En la Tabla B.19 se muestra la solución al Problema 2.10.5.

TABLA B.19. SOLUCIÓN AL PROBLEMA 2.10.5.

Dirección	$4N$	$4N+1$	$4N+2$	$4N+3$
0x1000_0000	08	CC	BB	AA

SOLUCIÓN AL PROBLEMA 2.11.

1. En la Tabla B.20 se muestra la solución al Problema 2.11.1.

TABLA B.20. SOLUCIÓN AL PROBLEMA 2.11.1.

2. En la Tabla B.21 se muestra la solución al Problema 2.11.2.

TABLA B.21. SOLUCIÓN AL PROBLEMA 2.11.2.

3. En la Tabla B.22 se muestra la solución al Problema 2.11.3.

TABLA B.22. SOLUCIÓN AL PROBLEMA 2.11.3.

Registro	b 31	b 19	b 13	b 0
FIOOPIN				1 0 0 1 1 1 1

Los LEDs se activan con nivel alto (se trata un display de cátodo común), por lo que hay que sacar por el puerto el valor: P0.6...P0.0 (g f e d c b a) = 1001111, bien con FIO0PIN o empleando los registros FIO0SET para poner a nivel alto los bits 0, 1, 2, 3 y 6; y FIO0CLR para poner a nivel bajo los bits 4 y 5.

SOLUCIÓN AL PROBLEMA 2.12.

1. En la Tabla B.23 se muestra la solución al Problema 2.12.1.

TABLA B.23. SOLUCIÓN AL PROBLEMA 2.12.1.

2. Escriba las instrucciones necesarias para leer la información de los switches y almacenar en R7 únicamente el valor del código de los switches.

LDR R0,=FIO2PIN

LD R7,[R0]

AND R7,#0x0F

3. Escriba las instrucciones necesarias para adecuar la información para presentarla en el display.

```
LDR R1,=FIO2SET  
MOV R2, #0xF0  
STR R2, [R1] ;Apago todos los LEDs  
LDR R1,=FIO2CLR  
STR R7, [R1]; Mando información complementada a los LEDs (ánodo común)
```

SOLUCIÓN AL PROBLEMA 2.13.

1. En la Tabla B.24 se muestra la solución al Problema 2.13.1.

TABLA B.24. SOLUCIÓN AL PROBLEMA 2.13.1.

2. En la Figura B.1 se muestra la solución al Problema 2.13.2.

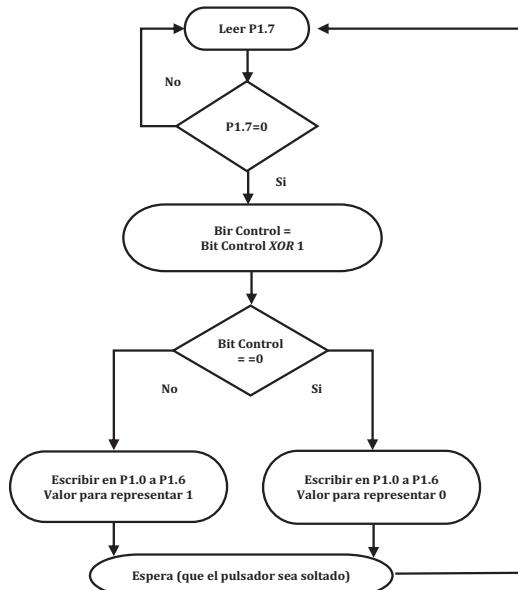


FIGURA B.1. SOLUCIÓN PROBLEMA 2.13.2.

SOLUCIÓN AL PROBLEMA 2.14.

- En la Tabla B.25 se muestra la solución al Problema 2.14.1.

TABLA B.25. SOLUCIÓN AL PROBLEMA 2.14.1.

Filas conjunto instrucciones	Registro configurado
1 a 4	PINSEL4
5 a 9	PINMODE4
10 a 13	FIO2DIR
14 a 16	FIO2SET
17 a 19	FIO2CLEAR
31 a 34	PINSEL4
35 a 38	PINMODE4
39 a 42	FIO2DIR

- Solución al Problema 2.14.2.1.

Se configuran 3 pines como GPIO sin resistencias internas y como salidas (P2.0, P2.1 y P2.2) El único periférico posible son los 3 LED (Líneas de código 1 a 13)

- Solución al Problema 2.14.2.2.

En las líneas 14 a 19, se envían primero 3 niveles “H” a (P2.0, P2.1 y P2.2) y luego 3niveles “L”.

Dada la conexión de los LED al final quedan apagados

- Solución al Problema 2.14.3.

Se configuran 2 pines como GPIO con resistencias de pull-up y como salidas (P2.10, y P2.11) El único periférico posible son los pulsadores

- Solución al Problema 2.14.4.

Sería necesario quitar el puente existente entre los dos pulsadores y añadir otra resistencia. Pero como se han configurado YA con resistencia de pull-up externas se podría quitar la existente y funcionaría de forma correcta

SOLUCIÓN AL PROBLEMA 2.15.

1. En la Figura B.2 se muestra la solución al Problema 2.15.1.

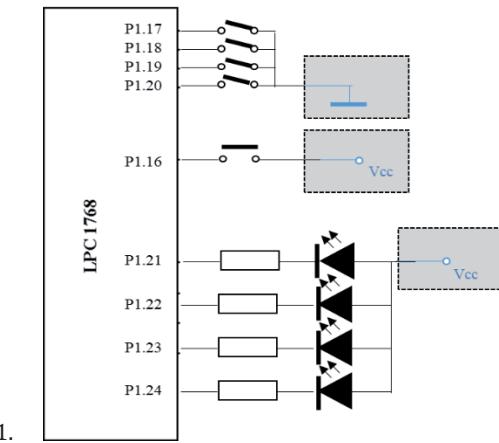


FIGURA B.2. CONEXIÓN DEL CIRCUITO PARA EL PROBLEMA 2.15.1.

2. En la Tabla B.26 se muestra la solución al Problema 2.15.2.

TABLA B.26. SOLUCIÓN AL PROBLEMA 2.15.2.

Dirección del registro				0x4002_C00C								PINSEL3					
Bit	3 1	2 8	2 4	2 0				1 6			1 2		8		4		0
Valor								0	0	0	0	0	0	0	0	0	0

Los terminales P1.16 a P1.24 deben tener la función GPIO. Están en la parte alta de P1, por lo que se configuran en PINSEL3, a valor 00 para función GPIO

Dirección del registro				0x4002_C04C								PINMODE3					
Bit	3 1	2 8	2 4	2 0				1 6			1 2		8		4		0
Valor								1	0	1	0	1	0	1	0	0	0

El terminal P1.16 debe tener pull-down (11), y los terminales P1.17 a P1.20 pull-up (00). Los terminales de salida los configuraremos sin pull-up ni pull-down (10).

Dirección del registro				0x2009_C200								FIO1DIR					
Bit	3 1	2 8	2 4	2 0				1 6			1 2		8		4		0
Valor				1	1	1	1	0	0	0	0						

Se configuran los terminales P1.16 a P1.20 como entrada y los terminales P1.21 a P1.24 como salida en FIO1DIR.

Dirección del registro				0x2009_C038								FIO1SET					
Bit	3 1	2 8	2 4	2 0				1 6			1 2		8		4		0
Valor				1	1	1	1										

Los LEDs se activan con niveles bajos, ponemos inicialmente los terminales P1.21 a P1.24 a nivel alto con FIO1SET.

SOLUCIÓN AL PROBLEMA 2.16.

1. En la Figura B.3 se muestra la solución al Problema 2.16.1.

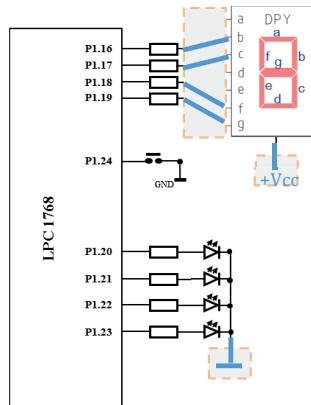


FIGURA B.3. CONEXIÓN DEL CIRCUITO PARA EL PROBLEMA 2.16.1.

2. En la Tabla B.27 se muestra la solución al Problema 2.16.2.

TABLA B.27. SOLUCIÓN AL PROBLEMA 2.16.2.

Fila	Instrucción
1	LDR R0, = 0x4002C00C
2	LDR R1,[R0]
3	MOV R2, # 0x0000
4	MOVT R2, # 0xFFFF
4	AND R1, R2
6	STR R1,[R0]
	LDR R0, = 0x4002C04C ; PINMODE3
	LDR R1,[R0]
	ORR R1, # 0xAAAA ; Bits 16-23 ni pull-up ni down
	MOV R2, #0xAAAA
	MOVT R2, #0xFFFFC ; Bit 24 pull-up
	AND R1, R2
	STR R1,[R0]
	LDR R0, = 0x2009C020 ; FIO1DIR
	LDR R1,[R0]
	MOV R2, # 0x0000
	MOVT R2, # 0x00FF
	ORR R1, R2 ; Bits 16 a 23 como salidas
	MOV R2, # 0xFFFF
	MOVT R2, # 0xFEFF
	AND R1, R2 ; Bit 24 como entrada
	STR R1,[R0]

3. En la Tabla B.28 se muestra la solución al Problema 2.16.3.

TABLA B.28. SOLUCIÓN AL PROBLEMA 2.16.3.

Fila	Instrucción
	LDR R0, = 0x2009 C03C
	MOV R1, #0X0000
	MOVT R1, #0X0OFF
	STR R1,[R0]

SOLUCIÓN AL PROBLEMA 2.17.

1. En la Tabla B.29 se muestra la solución al Problema 2.17.1.

TABLA B.29. SOLUCIÓN AL PROBLEMA 2.17.1

Puerto:	Pines:	conexión
1 (PINSEL 2 PINMODE 2 FIO1DIR)	0 ,1	GPIO / sin resistencia / entrada /Entradas del cto. 1
1 (PINSEL 2 PINMODE 2 FIO1DIR)	2, 3, 4	GPIO / sin resistencia / salida /Salidas al circuito 2

2. Solución al Problema 2.17.2.

Función como GPIO // Entrada // Con resistencia de Pull_up
 (PINSEL4) (FIO1DIR) (PINMODE 4)

3. En la Figura B.4 y la Figura B.5 se muestra la solución al Problema 2.17.3.

Situación 1

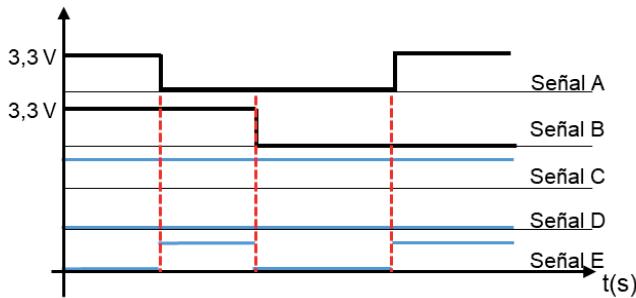


FIGURA B.4. SEÑALES GENERADAS POR EL CIRCUITO. SITUACIÓN 1.

Situación 2

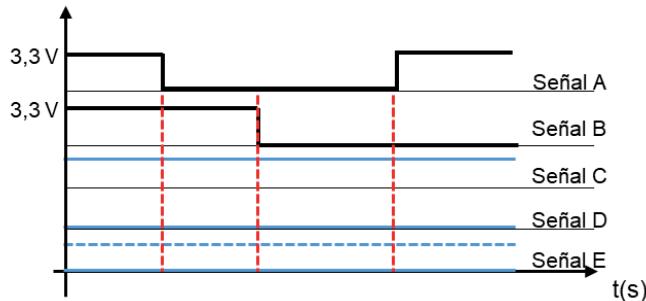


FIGURA B.5. SEÑALES GENERADAS POR EL CIRCUITO. SITUACIÓN 2.

SOLUCIÓN AL PROBLEMA 2.18

1. En la Tabla B.30 se muestra la solución al Problema 2.18.1.

TABLA B.30. SOLUCIÓN AL PROBLEMA 2.18.1.

Registro	Valor (hexadecimal)
STRELOAD	0x1869F

Si para 10ms el valor es 999999d, para 1ms es 99999d=0x1869F

2. En la Tabla B.31 se muestra la solución al Problema 2.18.2.

TABLA B.31. SOLUCIÓN AL PROBLEMA 2.18.2.

Bit Registro \	3 1	...	2 ...			2 0				1 6			1 2			8			4			0		
PINSEL0																		0	0	0	0	0	0	
PINMODE0																		1	0	1	0	1	0	
PINSEL4				0	1	0	1																	
PINMODE4				0	0	0	0																	
FIOODIR																				1	1	1		
FIOOSET																				1	1	1		

3. En la Tabla B.32 se muestra la solución al Problema 2.18.3.

TABLA B.32. SOLUCIÓN AL PROBLEMA 2.18.3.

Interrupción	Valor 8 bits (binario)							
EINT1 (Reset)	0	0	0	0	0	X	X	X
Systick	1	1	1	1	0	X	X	X
EINT0 (Start/Stop)	1	1	1	1	1	X	X	X

Esta es una posible solución, pero no es la única

4. Suponiendo que las prioridades asignadas han sido 0x60, 0x70 y 0x71, configure todos los registros necesarios, tanto del *NVIC* como del *core*, para que esas interrupciones puedan ser atendidas.

PRIMASK \leftarrow 0

FAULTMASK \leftarrow 0

BASEPRI \leftarrow >71

INTERRUPCIÓN SYSTICK, EINT0 Y EINT 1 \leftarrow ENABLE

APÉNDICE C. SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 3

SOLUCIÓN AL PROBLEMA 3.1.

En la Tabla C.1, se muestran las soluciones al Problema 3.1.

TABLA C.1. SOLUCIÓN AL PROBLEMA 3.1.

DIR	Contenido	Significado
0x00000000	0x10008000	Contenido del puntero de pila inicial tras un reset, MSP=0x1000_8000
0x00000004	0x000001A8	Contenido del contador de programa inicial tras un reset, PC=0x0000_01A8
0x00000008	0x000002B4	Dirección de la ISR de la excepción no enmascarable, NMI

SOLUCIÓN AL PROBLEMA 3.2.

- La solicitud de atención de la interrupción EINT1 se queda pendiente, aunque su valor de prioridad es menor (0x40) que el de la interrupción EINT3 (0x60), esto indica que ambas tienen mismo nivel de prioridad con desalojo (*Preempt Priority*).
- La solicitud de atención de la interrupción EINT2 interrumpe a la EINT3, esto indica que la EINT2 tiene un nivel mayor de prioridad con desalojo (*Preempt Priority*).
- Las condiciones anteriores solo se cumplen si los bits 7 y 6 de los *Interrupt Priority Registers* se corresponden con bits de prioridad con desalojo y los bits 5, 4 y 3 con la subprioridad, por tanto, el grupo de prioridad es el 5.

SOLUCIÓN AL PROBLEMA 3.3.

1. Mayor prioridad: NMI, EINT3 y TIMER1 menor. Mientras mayor es el valor del registro de prioridades, menor es la prioridad.
2. Solamente el NMI, la EINT3 tiene la misma prioridad preferente (Preempt Priority) y por tanto no desaloja al Timer1.
3. El SP se encuentra almacenado en la dirección 0x00000000 de la tabla de vectores, por tanto, su valor es 0x10000268.
4. La dirección de inicio de la rutina de atención de la interrupción del NMI se encuentra almacenada en la dirección 0x0000 0008, por tanto, su valor es 0x000001CB.
- 5.1. Es la interrupción del TIMER1 ya su dirección de inicio (0x000001E6) se corresponde con el contenido almacenado (0x000001E7 el PC siempre coge el bit menos significativo como 0) en la dirección 0x00000048 ($18 \times 4 = 72 = 0x48$) de la tabla de vectores.
- 5.2. Cuando se pasa a ejecutar el servicio de atención a una interrupción se almacenan en la pila 8 registros (PSP, PC, LR, R12, R3-R0), por tanto, si al valor inicial del SP es 0x10000268 le restamos 0x20 (32), su valor será 0x10000248.

SOLUCIÓN AL PROBLEMA 3.4.

1. Los valores de los registros son:

$SP = R13 = 0x1000_4000$

$PC = R15 = 0x0000_1008$ (pero en memoria realmente se almacena el bit de menor peso a ‘1’)

EINT1 comienza en 0x0000_100E (pero en memoria realmente se almacena el bit de menor peso a ‘1’) y su vector está en $35 \times 4 = 140d = 0x0000_008C$. En la Tabla C.2 se presenta la solución al Problema 3.4.1.

TABLA C.2. SOLUCIÓN AL PROBLEMA 3.4.1.

Dirección	4N	4N+1	4N+2	4N+3	Comentario
0x0000_0000	00	40	00	10	SP inicial
0x0000_0004	09	10	00	00	PC inicial
...
0x0000_008C	0F	10	00	00	Vector de EINT1

2. En la Tabla C.3 se presenta la solución al Problema 3.4.2.

TABLA C.3. SOLUCIÓN AL PROBLEMA 3.4.2.

Dirección	Registro almacenado	
0x10000_4000	...	Al atender a una interrupción automáticamente se almacenan xPSR, PC, LR, R12, R3-R0. Luego en la rutina ISR_EINT1 se almacena LR: PUSH (LR) Y en la rutina Absoluto se almacena R1: PUSH (R1)
0x10000_3FFC	xPSR (atención a EINT1)	
0x10000_3FF8	PC (atención a EINT1)	
0x10000_3FF4	LR (atención a EINT1)	
0x10000_3FF0	R12 (atención a EINT1)	
0x10000_3FEC	R3 (atención a EINT1)	
0x10000_3FE8	R2 (atención a EINT1)	
0x10000_3FE4	R1 (atención a EINT1)	
0x10000_3FE0	R0 (atención a EINT1)	
0x10000_3FDC	LR (desde ISR_EINT1)	
0x10000_3FD8	R1 (desde Absoluto)	

3. En la rutina ISR_EINT1 se lee el dato del periférico de entrada, almacenándolo en R0.

A continuación, se llama a la rutina Absoluto que lo compara con 0. Como en este caso es menor (es el -2 decimal) no se ejecuta el salto a Positivo y calcula la operación 0-R0 almacenando el resultado en R0. Es decir, calcula su opuesto, quedando en R0 el valor 2 decimal.

Al retornar de Absoluto y volver a la rutina ISR_EINT1 ésta envía el dato a la dirección donde esta mapeado el periférico de salida.

Por tanto, el valor que se escribe en el periférico de salida es el 0x0000_0002

4. Los tamaños de las rutinas son:

Tamaño rutina ISR_EINT1: $0x0000_1028 + 2 - 0x0000_100E = 0x1C = 28d$

Tamaño rutina Absoluto: $0x0000_1034 + 2 - 0x0000_1024 = 0x12 = 18d$

SOLUCIÓN AL PROBLEMA 3.5.

1. En la Tabla C.4 se presenta la solución al Problema 3.5.1.

TABLA C.4. SOLUCIÓN AL PROBLEMA 3.5.1.

t, (s)	Estado EINT0	Estado EINT1	Estado EINT2
1	X	X	A
2	X	P	A
3	A	P	A
4	A	P	A
5	X	P	A
6	X	P	A
7	X	P	A
8	X	A	X
9	X	X	X
10	X	X	X

- 2.1. La dirección de comienzo de una subrutina de atención a interrupción se encuentra almacenada en la dirección de la tabla de vectores correspondiente al Número de vector x 4.
- 2.2. El valor inicial que se carga en el contador de programa tras un reset se encuentra almacenado en la dirección 0x0000 0004. Por tanto, su valor es 0x0000 016C (el bit menos significativo del contador de programa siempre es 0).
- 2.3. El valor inicial que se carga en el Puntero de Pila tras un reset se encuentra almacenado en la dirección 0x0000 0000. Por tanto, su valor es 0x1000 0268.

SOLUCIÓN AL PROBLEMA 3.6.

1. Cuando se está ejecutando la rutina de atención a la EINT0, el SYSTICK no cuenta. Por tanto, se mantiene en un lazo infinito con n=0. En la Tabla C.5 se presenta la solución al Problema 3.6.1.

TABLA C.5. SOLUCIÓN AL PROBLEMA 3.6.1.

Estado EINT0	Estado EINT1	Estado EINT2	
1	X	A	X
2	X	A	P
3	A	A	P
4	A	A	P
5	A	A	P
6	A	A	P
7	A	A	P
8	A	A	P
9	A	A	P
10	A	A	P

- 2.1. La dirección de comienzo de una subrutina de atención a interrupción se encuentra almacenada en la dirección de la tabla de vectores correspondiente al Número de vector x 4.
- 2.2. El valor inicial que coge el contador de programa tras un reset se encuentra almacenado en la dirección 0x0000 0004. Por tanto, su valor es 0x0000 016C (el bit menos significativo del contador de programa siempre es 0).
- 2.3. El valor inicial que coge el Puntero de Pila tras un reset se encuentra almacenado en la dirección 0x0000 0000. Por tanto, su valor es 0x1000 0200

SOLUCIÓN AL PROBLEMA 3.7.

1. La interrupción EINT0 desaloja a las otras 2, y las interrupciones EINT1 y EINT2 no se desalojan entre sí. Con esto la solución al Problema 3.7.1 es la que se presenta en la Tabla C.6.

TABLA C.6. SOLUCIÓN AL PROBLEMA 3.7.1.

EINT0	EINT1	EINT2	
1	X	A	X
2	X	X	A
3	X	X	A
4	A	X	A*
5	A	X	A*
6	X	X	A
7	X	X	A
8	X	X	A
9	X	X	X
10	X	X	X

- 2.a La dirección de inicio de atención a la interrupción se obtiene de la tabla de vectores, multiplicando el número de vector por 4, y en esa entrada de la tabla de vectores se encuentra la dirección de la ISR, cambiando el bit de menor peso a 0:
- Systick: $15 \times 4 = 60 \rightarrow 0x3C \rightarrow 0x0000032A$
 EINT0: $34 \times 4 = 136 \rightarrow 0x88 \rightarrow 0x0000003CC$
 EINT1: $35 \times 4 = 140 \rightarrow 0x8C \rightarrow 0x0000040A$
 EINT2: $36 \times 4 = 144 \rightarrow 0x90 \rightarrow 0x00000448$
- 2.b La dirección de la rutina de reset está almacenada en la tabla de vectores, en la dirección 0x00000004. Según los datos proporcionados esta dirección es 0x0000016C.
- 2.c La dirección del puntero de pila inicial está almacenada en la tabla de vectores, en la dirección 0x00000000. Según los datos proporcionados esta dirección es 0x10000268.

SOLUCIÓN AL PROBLEMA 3.8.

- En la Figura C.1 se muestra la solución al Problema 3.8.1.

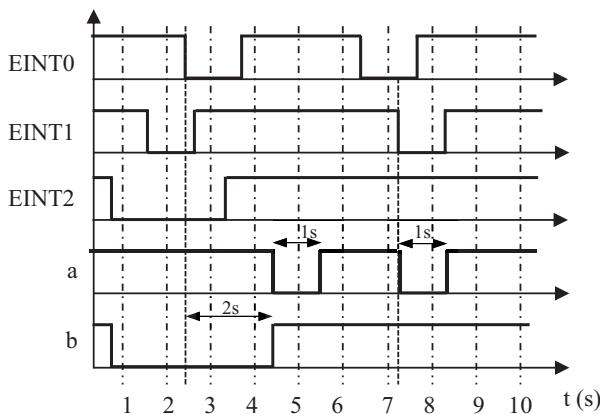


FIGURA C.1. SOLUCIÓN AL PROBLEMA 3.8.1.

En la Tabla C.7 se presentan las prioridades del Problema 3.8.1.

TABLA C.7. PRIORIDADES DEL PROBLEMA 3.8.1.

Interrupción	Prioridad	Subprioridad
EINT0	0 0 0 1	0
EINT1	0 0 1 0	0
EINT2	0 0 1 0	1
SysTick	0 0 0 0	0

En la Tabla C.8 se presenta la solución al Problema 3.8.1.

TABLA C.8. SOLUCIÓN AL PROBLEMA 3.8.1.

t (s)	Estado EINT0	Estado EINT1	Estado EINT2
0	X	X	X
1	X	X	A
2	X	P	A
3	X	P	A
4	X	P	A
5	X	A	X
6	X	X	X
7	P	X	X
9	P	X	X
10	P	X	X

2. En la Tabla C.9 se presenta la solución al Problema 3.8.2.

TABLA C.9. SOLUCIÓN AL PROBLEMA 3.8.2.

Dirección	Valor	Justificación
0x0000_0000	0x1000_0200	SP inicial
0x0000_0004	0x0000_016C	PC inicial
0x0000_003C	0x0000_032A	$15 \times 4 = 60 = 0x3C$ se encuentra el vector del SysTick
0x0000_0088	0x0000_03B4	$34 \times 4 = 136 = 0x88$ se encuentra el vector de EINT0
0x0000_008C	0x0000_03F0	$35 \times 4 = 140 = 0x8C$ se encuentra el vector de EINT1
0x0000_0090	0x0000_016C422	$36 \times 4 = 144 = 0x90$ se encuentra el vector de EINT2

SOLUCIÓN AL PROBLEMA 3.9.

1. En la tabla de vectores se debe contemplar espacio para: MSP, RESET, NMI, HARD FAULT, [hueco vectores 4-14], SYSTICK, [hueco vectores 16 y 17], EXTERNAL INTERRUPT #2, EXTERNAL INTERRUPT #3. En definitiva, hay que reservar espacio para MSP+19 vectores=20 posiciones. Redondeando a la potencia de 2 siguiente, la tabla será de 32 posiciones, y multiplicado por 4, resultará una tabla de 128 bytes.

La tabla se puede ubicar en direcciones múltiplo de su tamaño, siendo la dirección por defecto la 0. Por tanto, la tabla se podría ubicar en las direcciones: 0x0000_0000, 0x0000_0080, 0x0000_0100, 0x0000_0180, ... siempre que en esas direcciones exista un elemento de memoria que pueda albergar la tabla de vectores.

Lógicamente, la tabla de vectores se ubicará en la dirección 0 para que se pueda realizar el arranque (boot) del sistema (al menos MSP+RESET), y después configurando el registro VTOR se podría reubicar.

2. En la Tabla C.10 se presentan la solución al Problema 3.9.2.

TABLA C.10. SOLUCIÓN AL PROBLEMA 3.9.2.

Dirección Tabla de vectores (hex)	Contenido (hex)				Descripción
	Dir	Dir+1	Dir+2	Dir+3	
0	FE	FF	01	00	MSP. Se ubica al final de la memoria RAM estática (también es válida la dirección 0x2000)
4	01	010	00	00	RESET (la ISR se ubica a continuación de la tabla de vectores, esto es, dirección 0x100)
8	XX	XX	XX	XX	NMI
C	XX	XX	XX	XX	HARD FAULT
...	XX	XX	XX	XX	
3C	01	14	00	00	SYSTICK (ISR 0x1400)
...	XX	XX	XX	XX	
48	01	2C	00	00	Interrupción hardware #2 (ISR 0x2C00)
4C	01	38	00	00	Interrupción hardware #3 (ISR 0x3800)
...	XX	XX	XX	XX	

3. Con 3 bits, y Priority Group=6, esto implica que la prioridad se establece con 1 bit de preempt (bit 7) y 2 bits de subpriority, (bits 6 y 5). Esto es: P . S S x x x x x

Opción 1 Systick (0.00); Int_2 (0.01); Int_3 (0.10): No es válida ya que todas las excepciones tienen el mismo preempt, y por tanto, el Systick no sería más prioritario, y no podría desalojar a ninguna de las otras excepciones.

Opción 2 Systick (0.00); Int_2 (1.10); Int_3 (1.00): Es válida ya que en este caso el Systick tiene más prioridad que las otras dos interrupciones (preempt menor).

Opción 3 Systick (1.00); Int_2 (0.00); Int_3 (0.01): No es válida ya que el Systick tiene menor prioridad que las otras dos interrupciones (preempt superior), y nunca podría desalojarlas.

4. No se deben atender a las interrupciones 2 y 3, pero sí el resto (NMI, HARDFAULT y Systick). En primer lugar, PRIMASK y FAULTMASK deben estar desactivadas, esto es: PRIMASK=0; FAULTMASK=0.

Para que Int_2 e Int_3 no interrumpa, el valor del BASEPRI debe ser inferior al nivel de prioridad de las dos interrupciones, pero siempre, mayor que 0 (en caso contrario el BASEPRI estaría desactivado). Por tanto: $0 < \text{BASEPRI} \leq 0X20$.

SOLUCIÓN AL PROBLEMA 3.10.

1. ADC Excepción 38 -> $38 \times 4 = 152$ (0x98)

Justo antes de ejecutar la primera instrucción □ El PC debe tener el contenido del comienzo de esa instrucción R15 (PC) = 0x240

2. Al admitirse una interrupción se guardan automáticamente 8 registros = 32 bytes (0x20)

Actualmente la pila apunta a 0x1000 0248 como es Full Descending, se encontraría en 0x1000 0268

3. El orden en guardarse los 8 registros en pila automáticamente es PSR, PC, LR, R12, R3-R0

Por tanto, PC se habrá guardado en la posición 0x1000 0260 = 0x02AC

4. Priority Group= 5. Una de las posibles soluciones sería la mostrada en la Tabla C.11:

TABLA C.11. SOLUCIÓN AL PROBLEMA 3.10.4.

Interrupción	b.7	b.6	b.5	b.4	b.3			Prioridad (hex)
SysTick	0	0	x	x	x			0x00
ADC	0	1	x	x	x			0x40
EINT2	1	0	0	0	0			0x80
Timer0	1	0	0	0	1			0x88

SOLUCIÓN AL PROBLEMA 3.11.

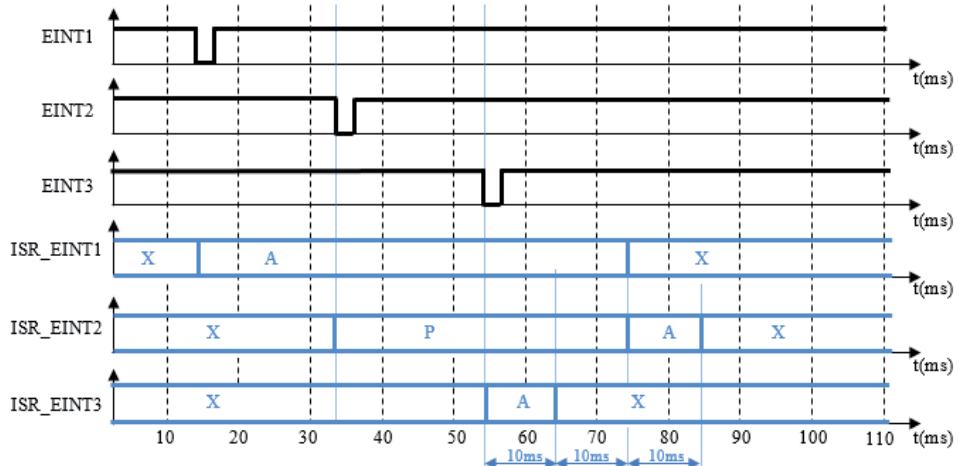
1. PRIGROUP= 0x4, Se define grupo de prioridad 4
2. En la Tabla C.12 se presenta la solución al Problema 3.11.2.

TABLA C.12. SOLUCIÓN AL PROBLEMA 3.11.2.

	Prioridad			Subprioridad				
	b7	b6	b5	b4	b3	b2	b1	b0
EINT1	0	0	1	1	1	X	X	X
EINT2	0	0	1	0	0	X	X	X
EINT3	0	0	0	1	0	X	X	X

3. A EINT3, por ser la que tiene el menor valor asignado en su campo de bits de prioridad.

4. En la Figura C.2 se muestra la solución al Problema 3.11.4.

**FIGURA C.2. SOLUCIÓN AL PROBLEMA 3.11.4.**

5. Las prioridades de las interrupciones son:

EINT1: $35 \times 4 = 140 = 0x8C \rightarrow$ Comienzo ISR: 0x1D2

EINT2: $36 \times 4 = 144 = 0x90 \rightarrow$ Comienzo ISR: 0x1FE

EINT3: $37 \times 4 = 148 = 0x94 \rightarrow$ Comienzo ISR: 0x218

6. En la Tabla C.13 se presenta la solución al Problema 3.11.6.

TABLA C.13. SOLUCIÓN AL PROBLEMA 3.11.6.

Dirección	Contenido (Little endian)			
0x0000 0000	00	20	00	10
0x0000 0004	6D ó 6E	01	00	00

SOLUCIÓN AL PROBLEMA 3.12.

1. La siguiente instrucción del código debe ser POP {R4,R2,PC}
2. En la Tabla C.14 se presenta la solución al Problema 3.12.2.

TABLA C.14. SOLUCIÓN AL PROBLEMA 3.12.2.

Línea de código	Registro o Dir. memoria	Contenido (Hexadecimal)			
1	0X10002000	F9	FF	FF	FF
	0X10001FFC	BB	AA	99	88
	0X10001FF8	33	22	11	00
	R13	10	00	1F	F8
2	R0	10	00	00	00
3	R4	33	DD	22	CC
	R2	11	BB	00	AA
	R0	0F	FF	FF	F8
4	R4	01	AC	22	CC
5	R2	11	BB	00	B6
6	0X0FFFFFF8	B6	00	BB	11

SOLUCIÓN AL PROBLEMA 3.13.

1. En la Tabla C.15 se presenta la solución al Problema 3.13.1

TABLA C.15. SOLUCIÓN AL PROBLEMA 3.13.1.

ISR	Tamaño (bytes)
Reset	1024
NMI	512
HARD FAULT	256
SYSTICK	128

La rutina de Reset comienza en la 0x400 (continuación de la tabla de vectores) y finaliza en la 0x7FF

La rutina de la NMI comienza en la 0x800 y finaliza en la 0x9FF

La rutina de Hard Fault comienza en la 0xA00 y finaliza en la 0xAF

2. En la Tabla C.16 se presenta la solución al Problema 3.13.2.

TABLA C.16. SOLUCIÓN AL PROBLEMA 3.13.2.

Dirección Tabla de vectores (hex) 4N 4N+3	Contenido (hex)				Descripción
	4N	4N+1	4N+2		
0x0000_0000	00	80	00	10	SP inicial. Última dirección de RAM múltiplo de 4 o siguiente al fin e RAM
0x0000_0004	01	04	00	00	PC inicial, inicio de rutina de Reset
0x0000_0008	01	08	00	00	ISR de la NMI

3. En la Tabla C.17 se presenta la solución al Problema 3.13.3.

TABLA C.17. SOLUCIÓN AL PROBLEMA 3.13.3.

Interrupción	Prioridad binario 3 bits	Prioridad Hex
Systick	0 0 0 X X X X X	0x00
Int. Hard. 1	1 0 0 X X X X X	0x80
Int. Hard. 0	1 0 1 X X X X X	0xA0

Subprioridad →

4. Systick desaloja a Int. hard. 1, Int_hardware_0 no se atiende por prioridad menor que BASEPRI. En la Figura C.3 se muestra la solución al Problema 3.13.4.

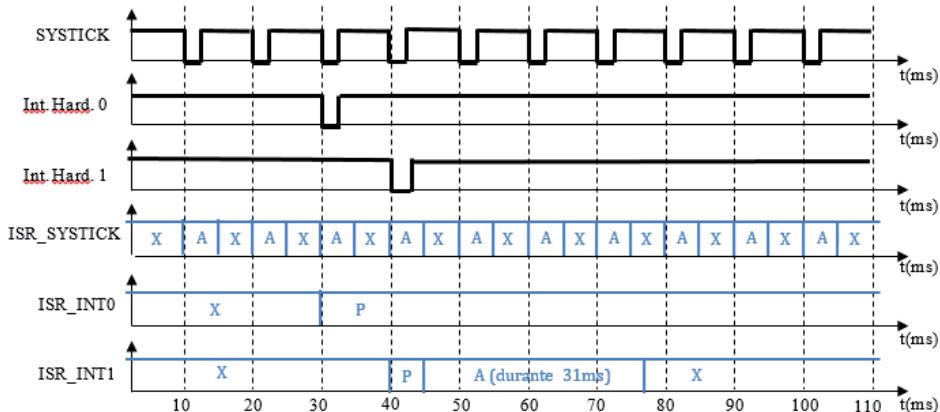


FIGURA C.3. SOLUCIÓN AL PROBLEMA 3.13.4.

SOLUCIÓN AL PROBLEMA 3.14.

1. El último vector es el de la excepción 38 que se guarda en la dirección 0x98 (38x4)
Por lo que el código empieza en la dirección 0x9c
2. En la Tabla C.18 se presenta la solución al Problema 3.14.2.

TABLA C.18. SOLUCIÓN AL PROBLEMA 3.14.2.

Nombre de excepción	Dirección de memoria	Contenido de memoria			
		4N	4N+1	4N+2	4N+3
RESET	0	00	40	00	10
	4	9C	00	00	00
NMI	8	9C	40	00	00
HARDFAULT	0xC	9C	44	00	00
SYSTICK	0x 3C	9C	48	00	00
Timer0	0x 44	9C	4C	00	00
EINT3	0x 94	9C	50	00	00
ADC	0x 98	9C	54	00	00

3. En la Tabla C.19 se presenta la solución al Problema 3.14.3.

TABLA C.19. SOLUCIÓN AL PROBLEMA 3.14.3.

Interrupción	Prioridad binario 5 bits	Prioridad Hex
EINT3	00 110	0x30
SysTick	00 111	0x38
Timer0	01 ***	0x40
ADC	10 ***	0x80

4. Solo puede interrumpir EINT3
5. Se debe realizar un muestreo (Polling) de las posibles fuentes (MR0 a MR3) para saber cuál fue la que solicitó la interrupción

SOLUCIÓN AL PROBLEMA 3.15.

1. El LPC1768 implementa 50 excepciones. El último vector está en $50 \times 4 = 200 = 0xC8$, por lo que el programa puede empezar en $0xCC$.
2. En la Tabla C.20 se presenta la solución al Problema 3.15.2.

TABLA C.20. SOLUCIÓN AL PROBLEMA 3.15.2.

Nombre de excepción	Dirección de memoria	Contenido de memoria			
		4N	4N+1	4N+2	4N+3
Reset	0x0000_0000	00	40	00	10
Reset	0x0000_0004	01	04	00	00
NMI	0x0000_0008	01	24	00	00
Hard Fault	0x0000_000C	01	2C	00	00
Systick	0x0000_003C	01	34	00	00
Timer 1	0x0000_0048	01	3C	00	00
EINT2	0x0000_0090	01	44	00	00
CAN	0x0000_00A4	01	4C	00	00

3. En la Tabla C.21 se presenta la solución al Problema 3.15.3.

TABLA C.21. SOLUCIÓN AL PROBLEMA 3.15.3.

Interrupción	Prioridad binario 5 bits	Prioridad Hex 8 bits
EINT2	011 01	0x68
SysTick	100 00	0x80
Timer1	010 00	0x40
CAN	011 00	0x60

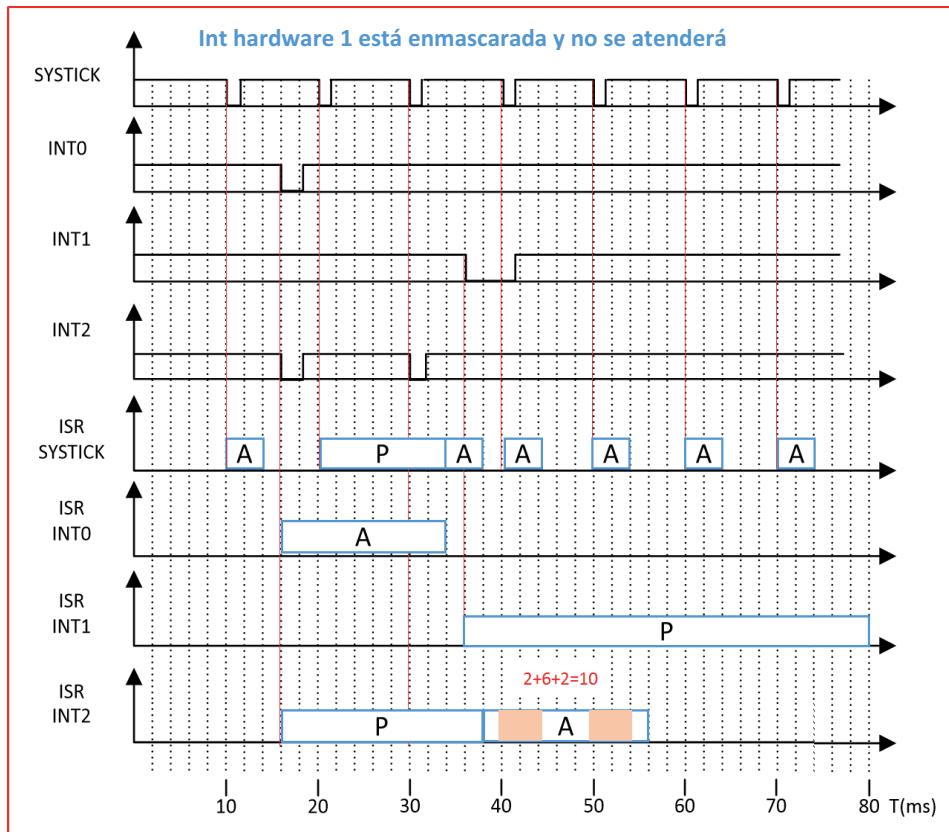
SOLUCIÓN AL PROBLEMA 3.16.

1. Las prioridades de las interrupciones son las mostradas en la Tabla C.22:

TABLA C.22. PRIORIDADES DEL PROBLEMA 3.16.1.

0	0	0	1	0 0 0 0	SYSTICK
0	0	1	0	0 0 0 0	Interrupt_0
1	0	0	0	0 0 0 0	Interrupt_1
0	1	0	0	0 0 0 0	Interrupt_2

En la Figura C.4 se muestra la solución al Problema 3.16.2.

**FIGURA C.4. SOLUCIÓN AL PROBLEMA 3.16.2.**

2. Cambiar Priority Group a 4
3. Enmascarando con cualquiera de los 3 registros especiales

PRIMASK = 1

FAULTMASK = 1

BASEPRI ≤ 10

4. La última excepción es la 18 (interrupción 2); por tanto, hay que reservar hasta la posición 0x48

La primera dirección disponible será la 0x4C

5. En la Tabla C.23 se presenta la solución al Problema 3.16.5.

TABLA C.23. SOLUCIÓN AL PROBLEMA 3.16.5.

Dirección Tabla de vectores (hex)	Contenido (hex)				Descripción
	4N	4N+1	4N+2	4N+3	
0x0	00 FC	40 3F	00	20	MSP
0x4	01	04	00	00	Inicio ISR Reset
0x8	01	10	00	00	Inicio ISR NMI
0x3C	01	20	00	00	Inicio ISR Systick

SOLUCIÓN AL PROBLEMA 3.17.

1. El valor inicial del registro PC, obtenido de la tabla de vectores, es el 0x0000_0168. En el momento del punto de ruptura el valor actual del PC es 0x0000_0952, por lo que el código ocupa 0x0000_0952-0x00_0168=0x7EA=2026 bytes. Es debido al código de inicialización que se incorpora al crear un proyecto en lenguaje C.

2. A partir de las imágenes anteriores se obtiene el valor del registro PC en cada situación. Al principio de la función inicio vale 0x0000_0640 y al final de la función inicio 0x0000_0878. La diferencia es el tamaño de la función inicio, 0x238=568 bytes.

Respecto al registro SP, dentro de la función tiene el valor 0x1000_0268 y al retornar 0x1000_0260, por lo que en la pila se habían introducido 8 bytes.

3. El número de la excepción que se está ejecutando se almacena en los 8 bits de menor peso del registro PSR. Al observar las cuatro figuras se determina que, de izquierda a derecha, este registro tiene los valores 0x22, 0x11, 0x23 y 0x24, que corresponde con los valores decimales 34, 17, 35 y 36.

4. Se está ejecutando la ISR de la excepción 0x11, 17 en decimal, comienza en la dirección 0x874 y finaliza en la dirección 0x950, por lo que el tamaño es la diferencia, 0xDC=220 bytes. El código de retorno es el 0xFFFF_FFF9, que indica que es un código de retorno de excepción por el que obtendrá de la pila la dirección de retorno.

5.a Las prioridades de las interrupciones son las mostradas en la Tabla C.24:

TABLA C.24. SOLUCIÓN AL PROBLEMA 3.17.4.

Interrupción	Prioridad	Subprioridad
Int 1	0 0 0 0	1 X X X
Int 2	0 0 0 1	0 X X X
Int 3	0 0 0 1	1 X X X

La interrupción 1 puede desalojar a las otras dos, y la interrupción 2, pese a ser más prioritaria que la interrupción 3, no la puede desalojar.

5.b Debe tener un nivel menor del 3, por ejemplo, el nivel 4 (expresada en 5 bits), 0010 0XXX.

5.c Poniendo a la Interrupción 4 un nivel de prioridad de 0, 0000 0XXX.

SOLUCIÓN AL PROBLEMA 3.18.

1. En la Tabla C.25 se presenta la solución al Problema 3.18.1.

TABLA C.25. SOLUCIÓN AL PROBLEMA 3.18.1.

Interrupción	Prioridad	Subprioridad
SisTick	0 0 0	0 0 X X X
EINT0	0 0 1	0 0 X X X
EINT1	0 1 0	0 0 X X X

El grupo de prioridad es el 4, por tanto, si la puede desalojar al tener mayor prioridad con desalojo.

2. El periodo de interrupción del SysTick es:

$$\text{LOAD} = 2000000$$

$$T_{\text{INT}} = (\text{LOAD}+1) \times T_{\text{CLK}} = (2000000+1) \times 10\text{ns} = 20\text{ms}$$

3. En la Figura C.5 se muestra la solución al Problema 3.18.3.

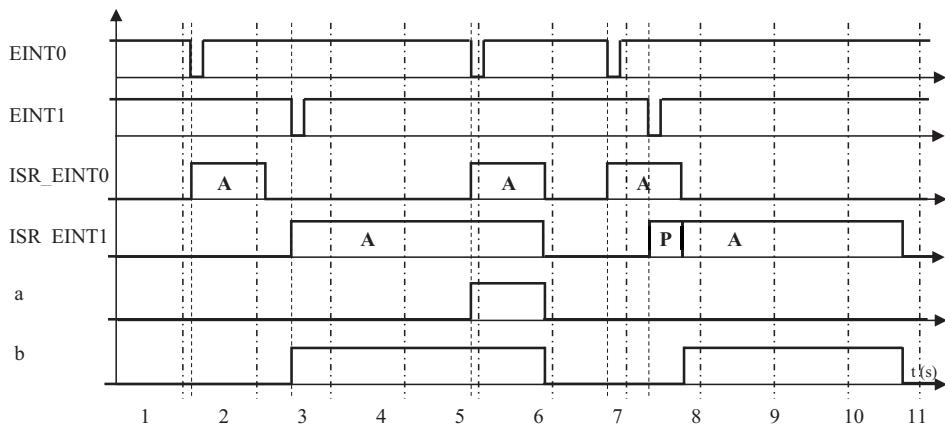


FIGURA C.5. SOLUCIÓN AL PROBLEMA 3.18.3.

4. En la Tabla C.26 se presenta la solución al Problema 3.18.4.

TABLA C.26. SOLUCIÓN AL PROBLEMA 3.18.4.

Interrupción	Prioridad	Subprioridad
SysTick	0	0 0 0 0 X X X
EINT0	0	0 1 0 0 X X X
EINT1	0	1 0 0 0 X X X

No hay desalojo entre ellas por lo que cuando entre la EINT0 el SysTick ya no entra más, no se incrementa el valor de m y se queda permanentemente en la ISR de la EINT0.

SOLUCIÓN AL PROBLEMA 3.19.

1. Si para 10ms el valor es 999999d, para 1ms es 99999d=0x1869F. La solución se muestra en la Tabla C.27.

TABLA C.27. SOLUCIÓN AL PROBLEMA 3.19.1.

Registro	Valor (hexadecimal)
STRELOAD	0x1869F

2. En la Tabla C.28 se presenta la solución al Problema 3.19.2.

TABLA C.28. SOLUCIÓN AL PROBLEMA 3.19.2.

Bit Registro	3 1	2 8	2 4	2 0	1 6	1 2	8	4	0
PINSEL0								0	0
PINMODE0								1	0
PINSEL4				0 1 0 1					
PINMODE4				0 0 0 0					
FIOODIR								1 1 1	
FIOOSET									1 1 1

3. En la Tabla C.29 se muestra una posible solución, pero no es la única.

Tabla C.29. Solución al Problema 3.19.3.

Interrupción	Valor 8 bits (binario)							
EINT1 (Reset)	0	0	0	0	0	X	X	X
Systick	1	1	1	1	0	X	X	X
EINT0 (Start/Stop)	1	1	1	1	1	X	X	X

4. En la Figura C.6 se muestra la solución al Problema 3.19.4.

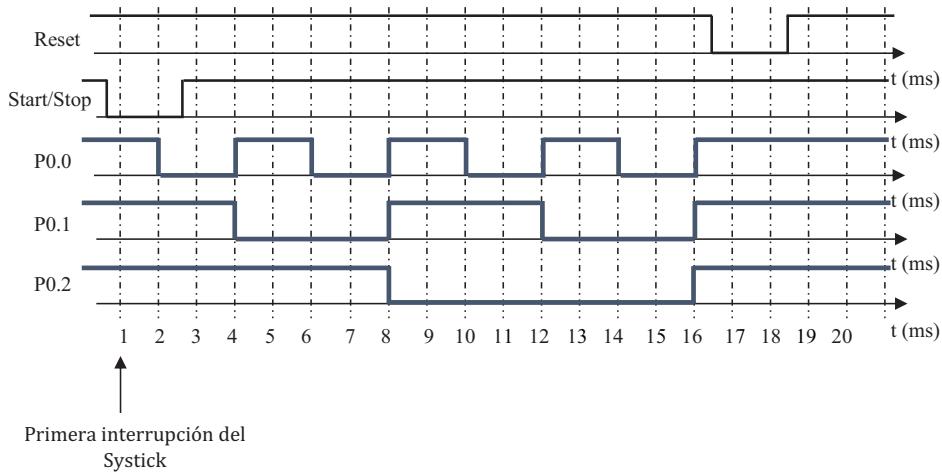


FIGURA C.6. SOLUCIÓN AL PROBLEMA 3.19.4.

SOLUCIÓN AL PROBLEMA 3.20.

1. En la Tabla C.30 se presenta la solución al Problema 3.20.1.

TABLA C.30. SOLUCIÓN AL PROBLEMA 3.20.1.

Excepción	Prioridad	
Ext_Int_1	0111 1	000
Ext_Int_2	1000 0	000
Ext_Int_18	0111 0	000

2. En la Tabla C.31 se presenta la solución al Problema 3.20.2.

TABLA C.31. SOLUCIÓN AL PROBLEMA 3.20.2.

Priority Group = 2, y por tanto no hay ningún bit de subprioridad.

Excepción	Excepción desalojada
Ext_Int_1	Ext_Int_2
Ext_Int_2	Ninguna
Ext_Int_18	Ext_Int_1 y Ext_Int_2

3. Solo puede interrumpir Ext_Int_18 ya que es la única con valor de prioridad inferior a 0x75

4. En la Figura C.7 se muestra la solución al Problema 3.20.4.

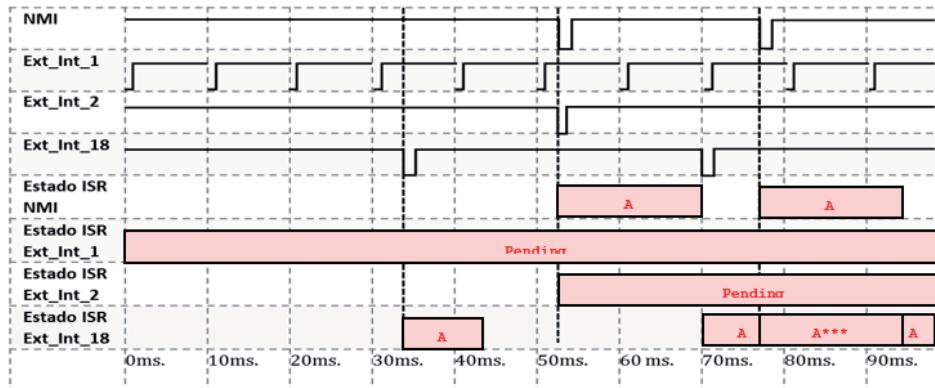


FIGURA C.7. SOLUCIÓN AL PROBLEMA 3.20.4.

5. En la Tabla C.32 se presenta la solución al Problema 3.20.5.

TABLA C.32. SOLUCIÓN AL PROBLEMA 3.20.5.

Excepción	Dirección de memoria
(SP)	0x0
RESET	0x4
NMI	0x8
HF	0xC
Ext_Int_1	0x44
Ext_Int_2	0x48
Ext_Int_18	0x88

6. La solución es 0x8C.

SOLUCIÓN AL PROBLEMA 3.21.

- En la Tabla C.33 se presenta la solución al Problema 3.21.1.

TABLA C.33. SOLUCIÓN AL PROBLEMA 3.21.1.

Dirección (hex)	Contenido de memoria				Descripción
	4N	4N+1	4N+2	4N+3	
0000_0000	00	40	00	00	Valor inicial del MSP
0000_0004	01	02	00	00	Dirección de la ISR del RESET
0000_0008	41	04	00	00	Dirección de la ISR del NMI
0000_000C	01	05	00	00	Dirección de la ISR del Hard Fault
0000_003C	21	06	00	00	Dirección de la ISR del Systick
0000_0050	01	0C	00	00	Dirección de la ISR de la IRQ externa #4

- Los tamaños de las rutinas son:

ISR del RESET: 0x0440-0x0200=0x240=576 Bytes

ISR del NMI: 0x0500-0x440=0x0C0=192 Bytes

3. El conjunto de vectores representado en la Tabla 3.2 representa un tamaño de 84 Bytes (0x0-0x53). No obstante, se indica que la rutina de RESET se ha ubicado a partir de la tabla de vectores, y dicha rutina empieza en la dirección 0x200. Por tanto, se puede concluir que la tabla de vectores podría llegar hasta la dirección 0x1FF, por lo que el tamaño realmente reservado es de 512 Bytes.

4. Tal como muestra la dirección 0 de la tabla de vectores, el puntero de pila MSP se ha iniciado con el valor 0x4000 (que se corresponde con la primera dirección de memoria RAM). Cuando se realice la primera transferencia a pila, el puntero de pila se decrementará en 4, siendo su contenido 0x3FFC, y tal como muestra la Tabla 3.1, esta dirección pertenece a memoria ROM, sobre la que NO se puede escribir. Por tanto, el contexto no se guardará adecuadamente. La manera de resolver este problema sería iniciando el puntero de pila en una dirección de RAM que permita que la pila crezca adecuadamente. En la situación de este problema, el valor más adecuado sería la dirección última de RAM (o la siguiente). La dirección 0 de la tabla de vectores debería haberse inicializado con el valor 0x5FFC (o 0x6000). Otra opción sería cargar en el registro R13 ese valor antes de que se produzca una solicitud de excepción: LDR R13, =0x5FFC.

5. PRIMASK=1 -> Solo se aceptan las excepciones de NMI y HardFault, siendo la primera más prioritaria (Prioridad NMI= -2; Prioridad HardFault=-1). En la Figura C.8 se muestra la solución al Problema 3.21.5.

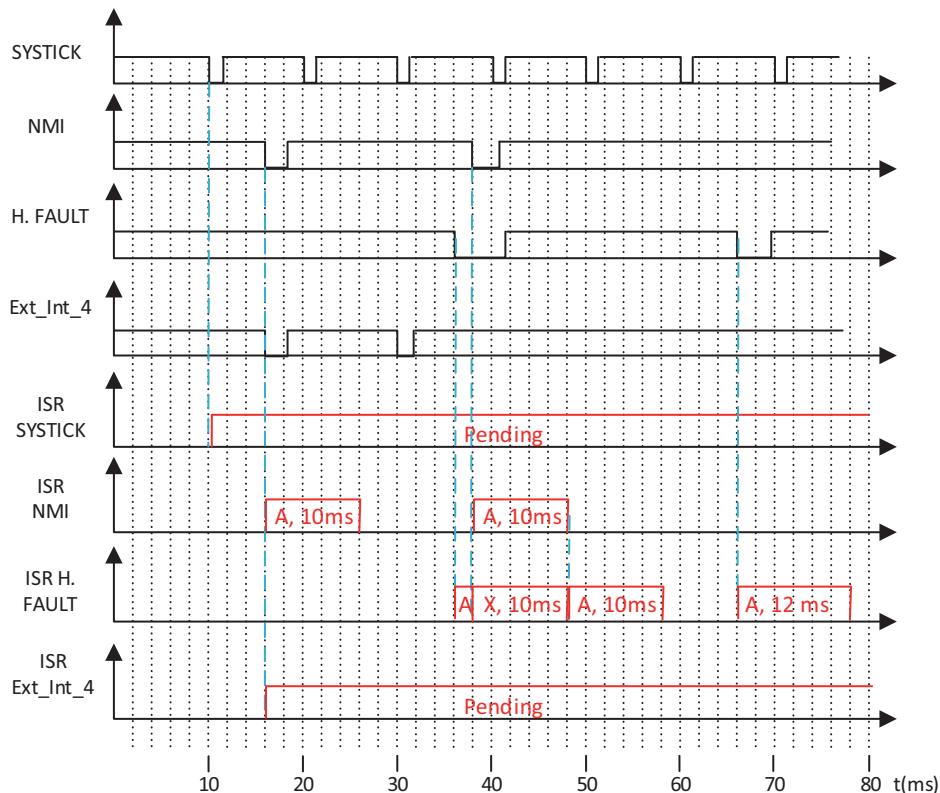


FIGURA C.8. SOLUCIÓN AL PROBLEMA 3.21.5.

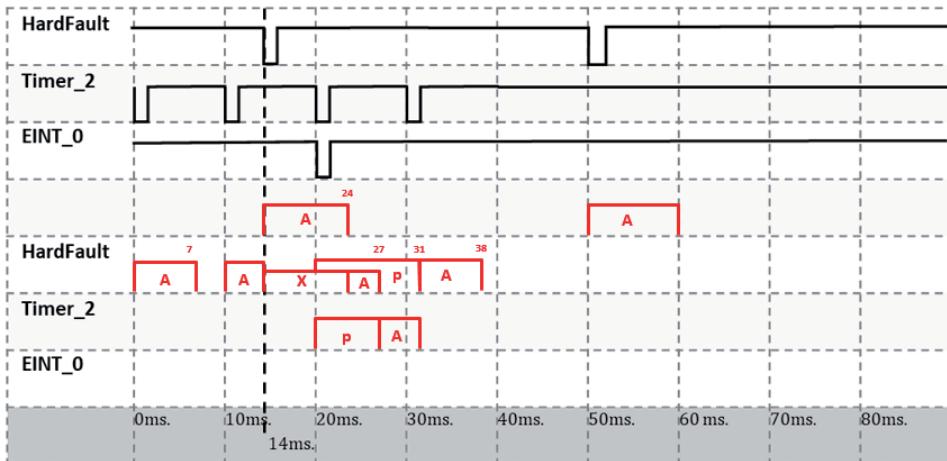
SOLUCIÓN AL PROBLEMA 3.22.

1. En la Tabla C.34 se presenta la solución al Problema 3.22.1.

TABLA C.34. SOLUCIÓN AL PROBLEMA 3.22.1.

Dirección (hex)	Contenido de memoria				Descripción
	4N	4N+1	4N+2	4N+3	
00	80	00	10		MSP Inicial
01	04	00	00		Vector de RESET. Dirección de comienzo del código
01	0C	00	00		Dirección de comienzo de la ISR NMI
01	14	00	00		Dirección de comienzo de la ISR HF
01	1C	00	00		Dirección de comienzo de la ISR Timer2
81	1C	00	00		Dirección de comienzo de la ISR EINT_0

2. En la Figura C.9 se muestra la solución al Problema 3.22.2.

**FIGURA C.9. SOLUCIÓN AL PROBLEMA 3.22.2.**

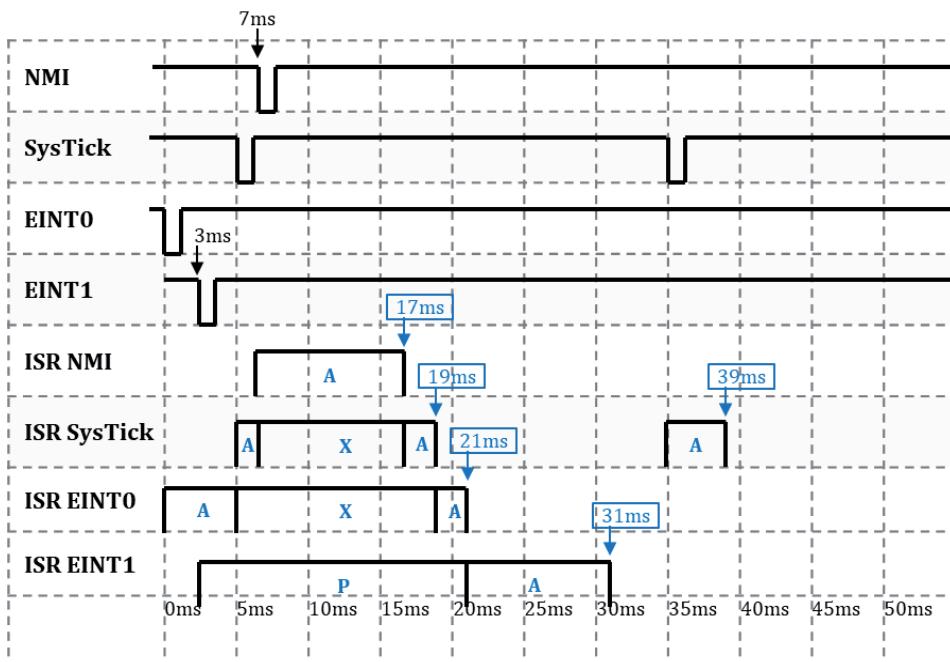
SOLUCIÓN AL PROBLEMA 3.23.

1. En la Tabla C.35 se presenta la solución al Problema 3.23.1.

TABLA C.35. SOLUCIÓN AL PROBLEMA 3.23.1.

Dirección (hex)	Contenido de memoria				Descripción
	4N	4N+1	4N+2	4N+3	
0x0000_0000	00	00	02	10	MSP Inicial
0x0000_0004	01	04	00	00	Vector de RESET. Dirección de comienzo del código
0x0000_0008	01	14	00	00	Dirección de comienzo de la ISR NMI
0x0000_000C	01	18	00	00	Dirección de comienzo de la ISR Hard Fault
0x0000_003C	01	1C	00	00	Dirección de comienzo de la ISR SysTick
0x0000_0088	81	1C	00	00	Dirección de comienzo de la ISR EINT0
0x0000_008C	81	20	00	00	Dirección de comienzo de la ISR EINT1

2. En la Figura C.10 se muestra la solución al Problema 3.23.2.

**FIGURA C.10. SOLUCIÓN AL PROBLEMA 3.23.2.**

APÉNDICE D. SOLUCIONES A LOS PROBLEMAS DEL CAPÍTULO 4

SECCIÓN 1. AMPLIACIÓN DE MEMORIAS

SOLUCIÓN AL PROBLEMA 4.1.

Hay que realizar una ampliación tanto de la longitud de palabra, como del número de palabras. Con dos chips de 1Gx8 ampliando longitud de palabra conseguiríamos un sistema de memoria de 1Gx16, y con dos sistemas de estos ampliando el número de palabras, 2Gx16.

Para seguir disponiendo de señal de chip select será necesario incluir un par de puertas OR que permitan inhibir las entradas CS#. El sistema completo se muestra en la Figura D.1.

En el circuito, las señales CSs#, WEs# y OEs# corresponden a las señales de control del sistema de memoria resultante.

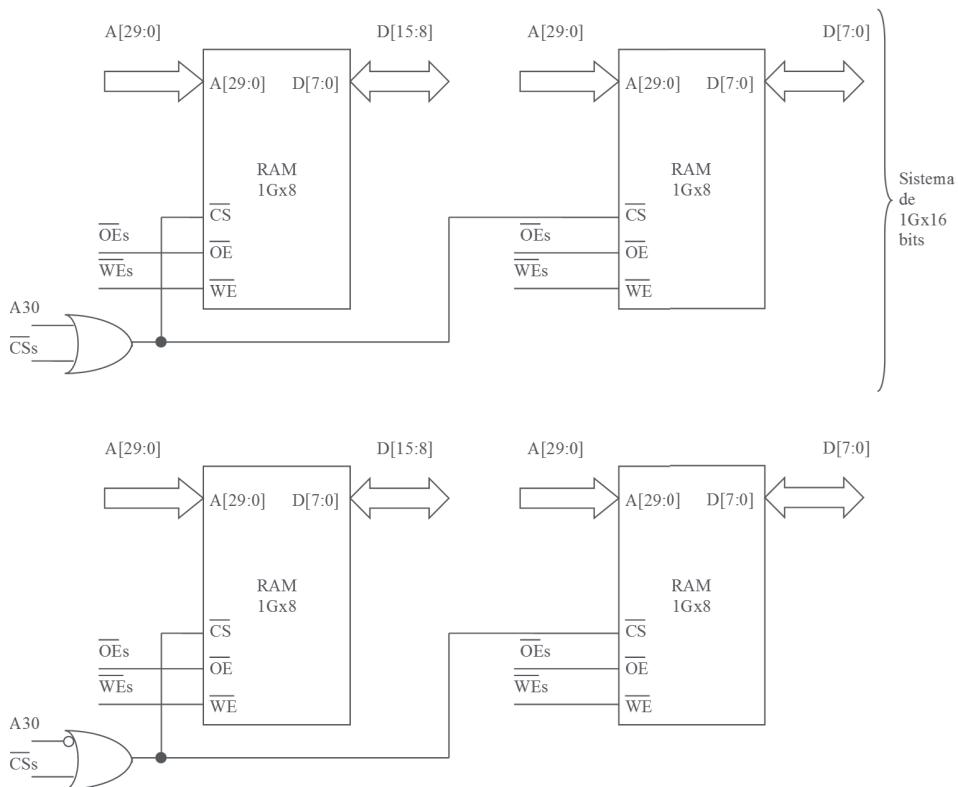


FIGURA D.1. SOLUCIÓN AL PROBLEMA 4.1.

SOLUCIÓN AL PROBLEMA 4.2.

1. La Figura D.2 muestra el sistema de 512Kx8.

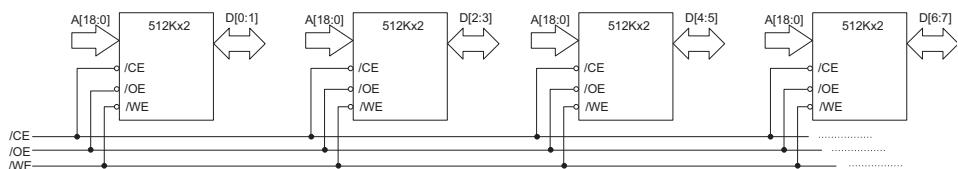


FIGURA D.2. IMPLEMENTACIÓN DE 512Kx8.

2. La Figura D.3 muestra el sistema de 512Kx16.

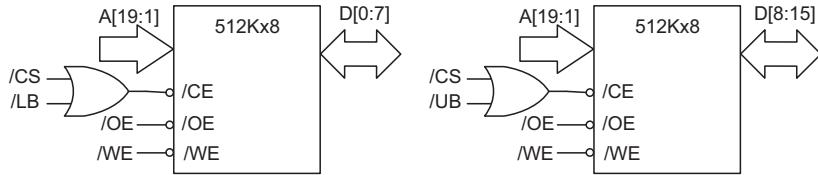


FIGURA D.3. IMPLEMENTACIÓN DE UN SISTEMA DE 512Kx16.

SOLUCIÓN AL PROBLEMA 4.3.

1. Tamaño: $2^{18} \times 16$ bits $\rightarrow 256\text{Kx16}$; Tamaño en bytes: 512Kbytes. La Figura D.4 muestra el etiquetado de los buses con las líneas correspondientes del microprocesador.

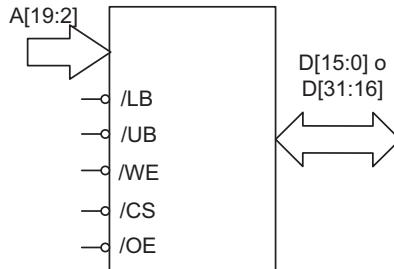


FIGURA D.4. ETIQUETADO DE BUSES.

2. La Figura D.5 muestra el bloque de memoria resultante.

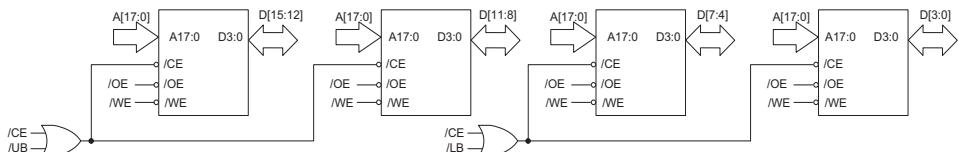


FIGURA D.5. BLOQUE DE MEMORIA EQUIVALENTE.

SOLUCIÓN AL PROBLEMA 4.4.

La Figura D.6 muestra la ampliación de memoria requerida.

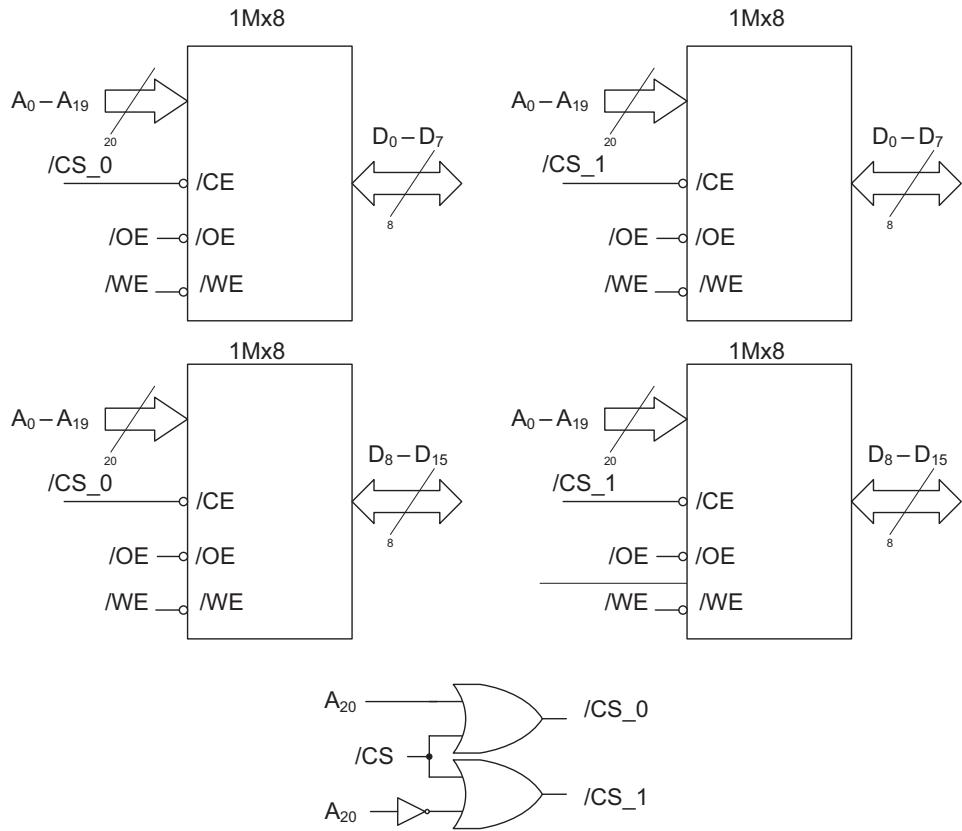


FIGURA D.6. AMPLIACIÓN DE MEMORIA REQUERIDA.

SECCIÓN 2. DISEÑO DE MAPAS DE MEMORIA

SOLUCIÓN AL PROBLEMA 4.5.

La Tabla D.1 muestra las direcciones de inicio y fin de los bloques que conforman el sistema.

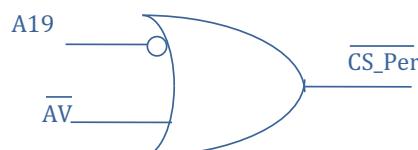
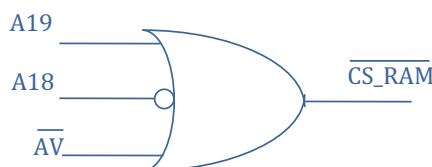
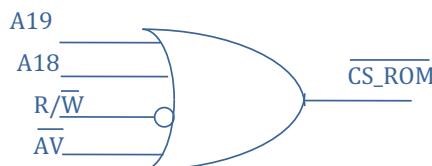
TABLA D.1. DIRECCIONES DE INICIO Y FIN DE LOS BLOQUES DEL SISTEMA DE MEMORIA.

Circuito/chip	Dir. inicio	Dir. Fin	Funcionalidad/observaciones (si procede)
ROM0	0x00000	0x3FFFC	Mapeado en banco 4N
ROM1	0x00001	0x3FFFD	Mapeado en banco 4N+1
ROM2	0x00002	0x3FFE	Mapeado en banco 4N+2
ROM3	0x00003	0x3FFF	Mapeado en banco 4N+3
RAM0	0x40000	0x7FFFC	Mapeado en banco 4N
RAM1	0x40001	0x7FFFD	Mapeado en banco 4N+1
RAM2	0x40002	0x7FFE	Mapeado en banco 4N+2
RAM3	0x40003	0x7FFF	Mapeado en banco 4N+3
PER0	0x80000	0x8003C	Mapeado en banco 4N
PER1	0x80001	0x8003D	Mapeado en banco 4N+1
PER2	0x80002	0x8003E	Mapeado en banco 4N+2
PER3	0x80003	0x8003F	Mapeado en banco 4N+3
---	---	---	

2. Puesto que no se especifica que se utilice ningún tipo de decodificación en particular, optamos por la decodificación incompleta pues es más sencilla de implementar. Tabla D.2 muestra la tabla de decodificación y la Figura D.7 la implementación.

TABLA D.2. LÍNEAS IMPLICADAS EN LA LÓGICA DE SELECCIÓN.

A19	A18	Bloque activo
0	0	ROM
0	1	RAM
1	X	Periféricos

**FIGURA D.7. LÓGICA DE SELECCIÓN REQUERIDA.**

3. La Figura D.8 muestra la conexión de los elementos del sistema.

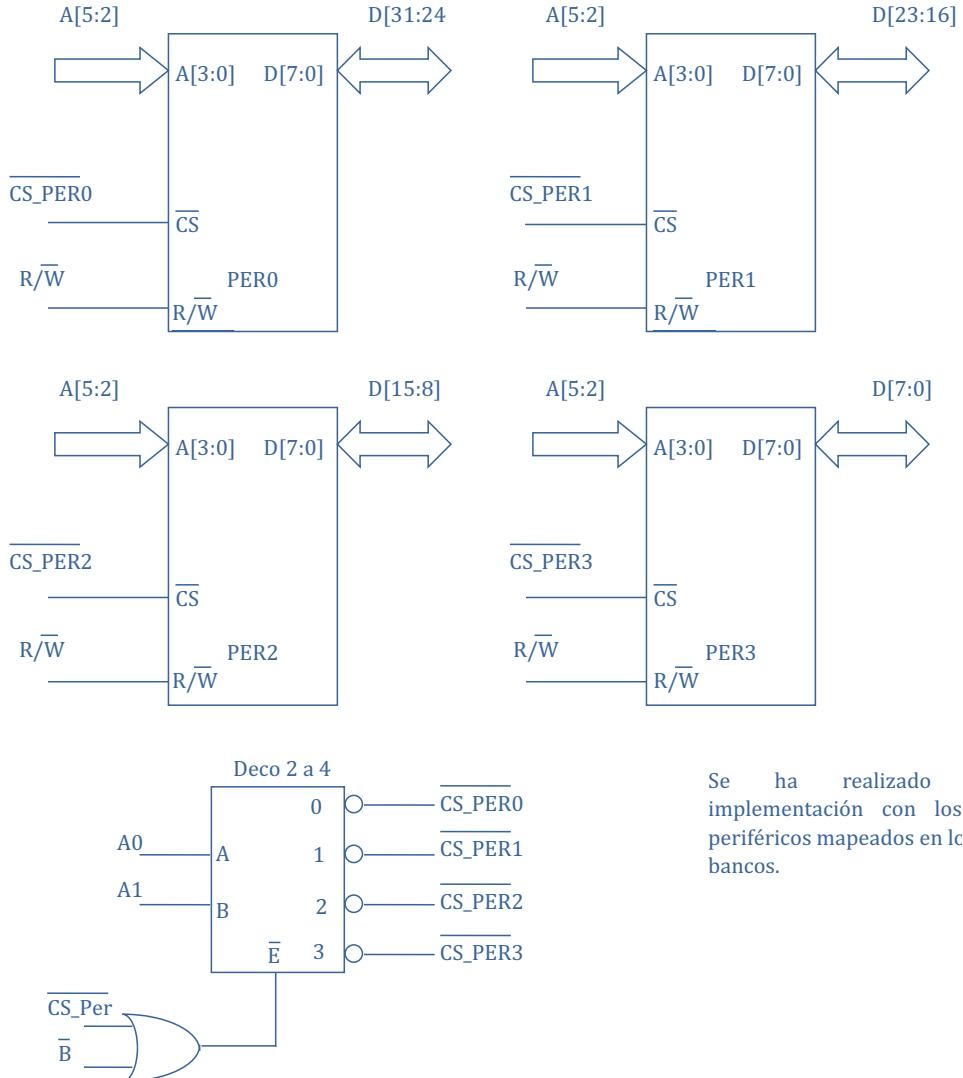


FIGURA D.8. CONEXIÓN DE TODOS LOS ELEMENTOS DEL SISTEMA.

SOLUCIÓN AL PROBLEMA 4.6.

- Las necesidades de memoria se indican a continuación:

ROM: $16\text{Mbytes} + 5\text{Mbytes} + 3\text{Mbytes} = 24\text{Mbytes}$ -> Valor comercial 32Mx8

RAM: $1\text{Mbyte} + 2\text{Mbytes} = 3\text{Mbytes}$ -> Valor comercial 4Mx8

- Hay que realizar una ampliación del número de palabras. La solución tanto de 3 como de 4 chips de 8Mbytes (24Mbytes o 32 Mbytes) son correctas. La Figura D.9 muestra el resultado de la implementación. En el circuito, las señales CSs#, y OEs# corresponden a las señales de control del sistema de memoria resultante.

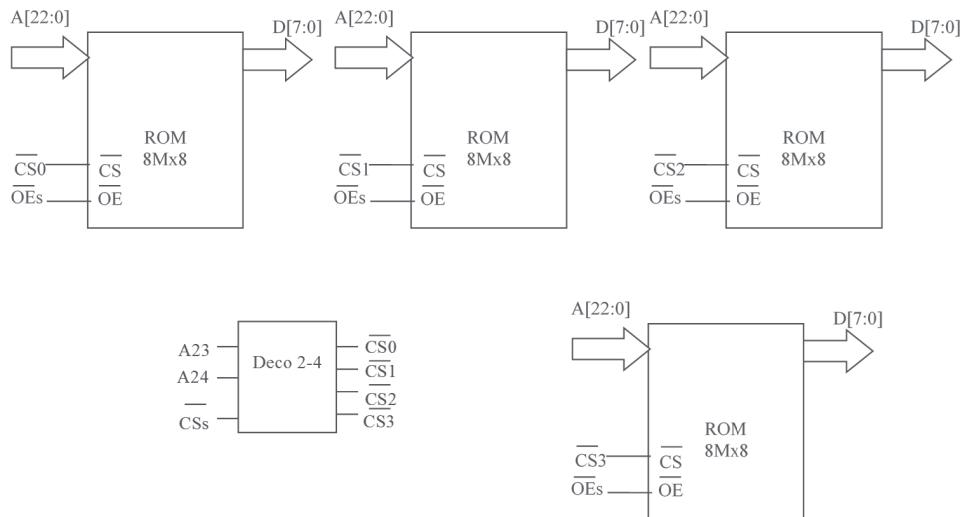


FIGURA D.9. SISTEMA DE MEMORIA RESULTANTE.

SOLUCIÓN AL PROBLEMA 4.7.

La Figura D.10 muestra la ampliación de memoria.

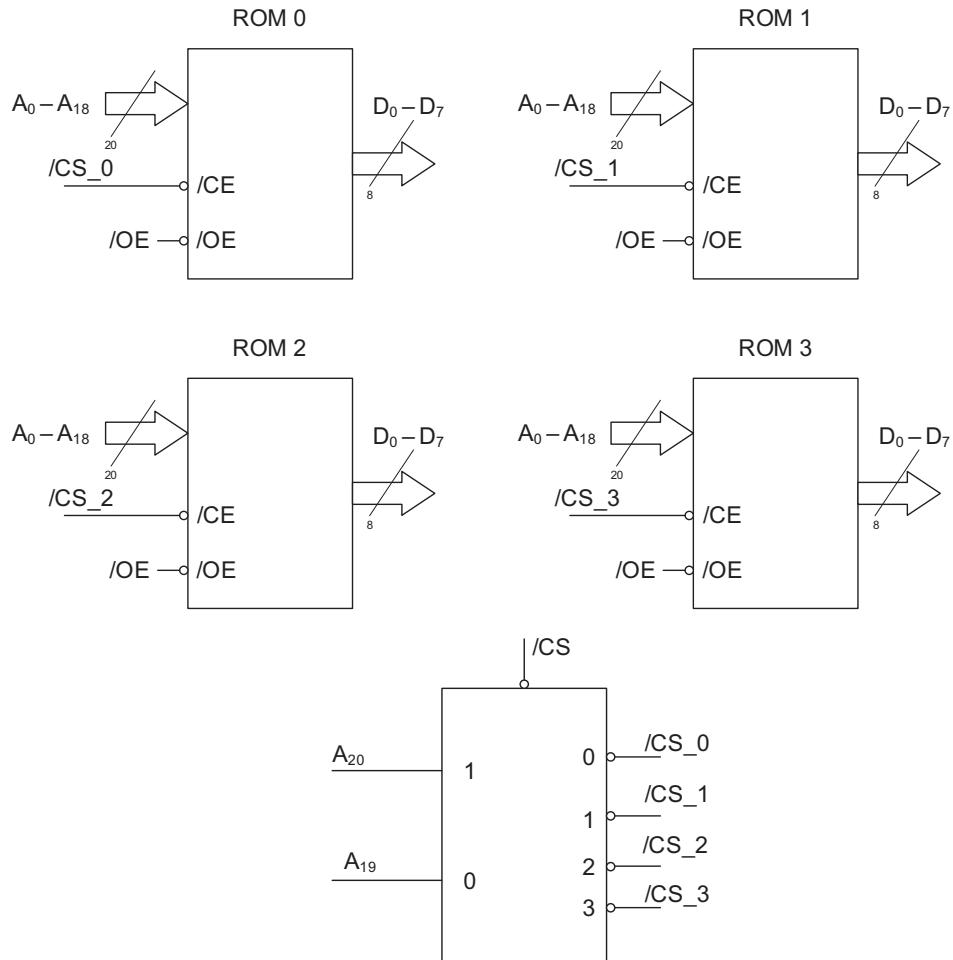


FIGURA D.10. AMPLIACIÓN DE MEMORIA RESULTANTE.

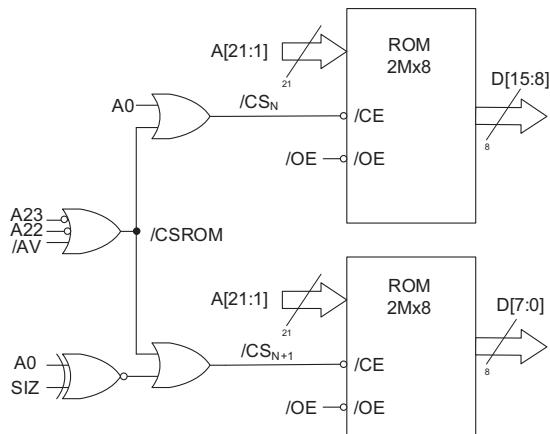
2. La Figura D.11 muestra la resolución del apartado.

4Mbytes \rightarrow 2Mx16 \rightarrow 2 chips
 2Mx8. El rango de direcciones es 0xC0_0000 – 0xFF_FFFF.
 Un chip implementa las direcciones pares y otro las impares.

SIZ=0 \rightarrow 8 bits

SIZ=1 \rightarrow 16 bits

A0	SIZ	/CS _N	/CS _{N+1}
0	0	0	1
0	1	0	0
1	0	1	0
1	1	1	1



$$\overline{CS}_N = A0$$

$$\overline{CS}_{N+1} = \overline{A0} \oplus SIZ$$

FIGURA D.11. IMPLEMENTACIÓN DEL SISTEMA.

3. La Tabla D.3 y la Tabla D.4 muestran dos posibles soluciones el ejercicio.

TABLA D.3. SOLUCIÓN 1.

Dirección	2N	2N+1	
0X80_0000	R0		PER0
0X80_0002	R1		
0X80_0004	R2		
0X80_0006	R3		
0X80_0008	R0		PER1
0X80_000A	R1		
0X80_000C	R2		
0X80_000E	R3		

TABLA D.4. SOLUCIÓN 2.

Dirección	2N	2N+1	Dirección
	PER0	PER1	
0X80_0000	R0	R0	0X80_0001
0X80_0002	R1	R1	0X80_0003
0X80_0004	R2	R2	0X80_0005
0X80_0006	R3	R3	0X80_0007

SOLUCIÓN AL PROBLEMA 4.8.

1. La Tabla D.5 muestra la resolución del Problema 4.8.1.

TABLA D.5. CAPACIDAD EN BYTES DE LOS SISTEMAS.

Sistema	Capacidad en bytes	Nº de chips
ROM	8 MBytes	2x4Mbytes
RAM1	512Kbytes	2x256Kbytes
RAM2	4Mbytes	2x2Mbytes
IO	2Mbytes	

2. La Tabla D.6 muestra la resolución del Problema 4.8.2.

TABLA D.6. DIRECCIONES DE INICIO Y FIN DE LOS BLOQUES.

Sistema	Dirección inicio (hex)	Dirección fin (hex)
ROM	0x000_0000	0x07F_FFFF
RAM1	0x100_0000	0x107_FFFF
RAM2	0x1C0_0000	0x1FF_FFFF
IO	0x180_0000	0x19F_FFFF

3. Se mantiene decodificación completa (se recuerda que es organización *big endian*). La Figura D.12 muestra la solución.

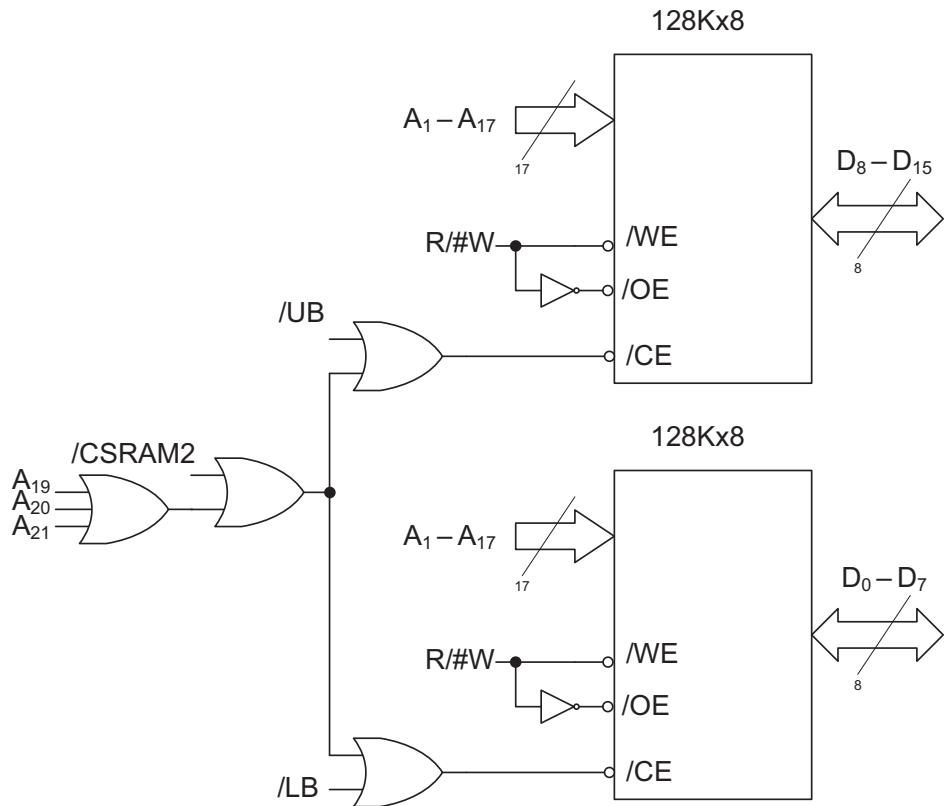


FIGURA D.12. CONEXIÓN DE LA MEMORIA AL MICROPROCESADOR.

SOLUCIÓN AL PROBLEMA 4.9.

- La Tabla D.7 muestra la resolución del Problema 4.9.1.

TABLA D.7. DIRECCIONES DE INICIO Y FIN DE LOS BLOQUES DEL SISTEMA.

Chip	Dirección inicio (hex)	Dirección fin (hex)
ROM0	0x00_0000	0x1F_FFFE
ROM1	0x00_0001	0x1F_FFFF
I/O	0x20_0000	0x21_FFFF
PER0	0x22_0000	0x22_001E
PER1	0x22_0001	0x22_001F
RAM0	0xC0_0000	0xFF_FFFE
RAM1	0xC0_0001	0xFF_FFFF

- La Tabla D.8 y la Figura D.13 muestran la solución del Problema 4.9.2.

TABLA D.8. TABLA DE DECODIFICACIÓN.

A₂₃ A₂₂ A₂₁ A₂₀	A₁₉ A₁₈ A₁₇ A₁₆	Sistema
0 0 0 X	X X X X	ROM
0 0 1 0	0 0 0 X	I/O
0 0 1 0	0 0 1 0	PERS
1 1 X X	X X X X	RAM

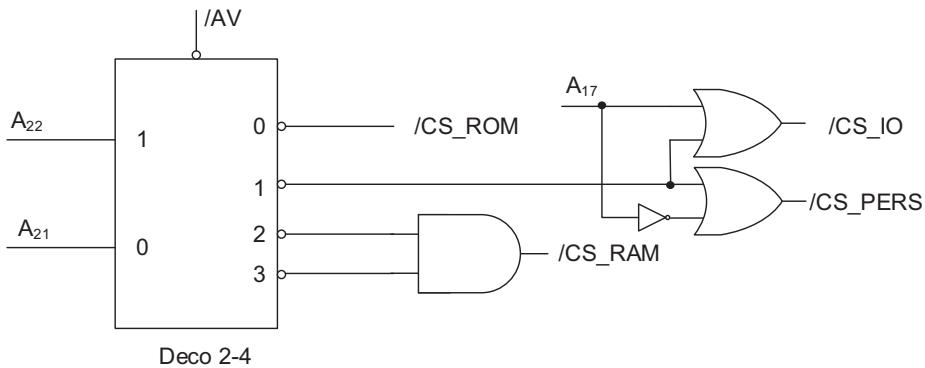


FIGURA D.13. IMPLEMENTACIÓN DE LA LÓGICA DE SELECCIÓN.

3. La Figura D.14 muestra la solución del Problema 4.9.3.

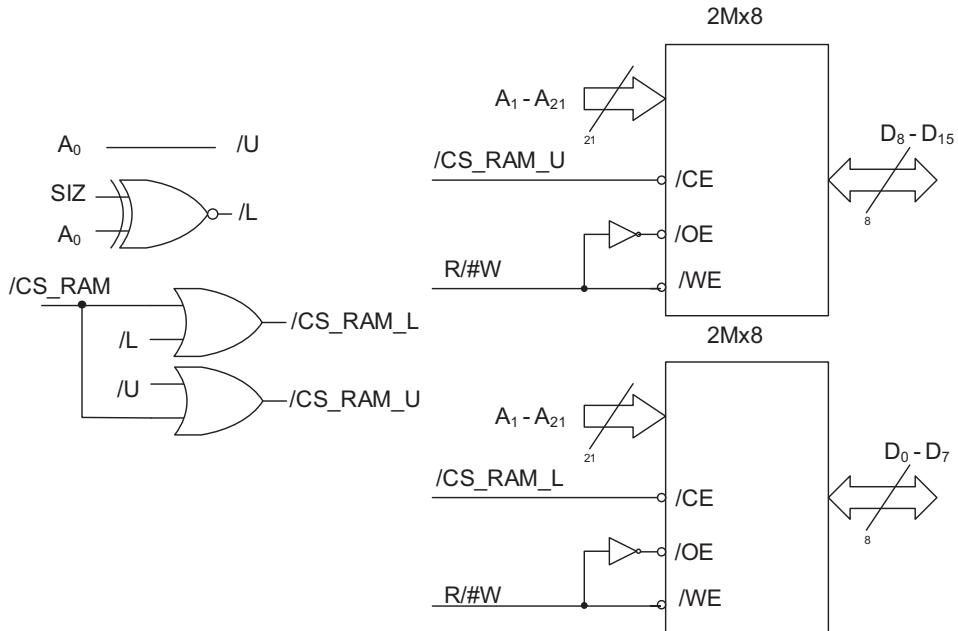


FIGURA D.14. CONEXIÓN DE LA MEMORIA AL MICROPROCESADOR.

SOLUCIÓN AL PROBLEMA 4.10.

1. A continuación se muestra la solución.

Espacio de direccionamiento total 16 Mbytes
 (16 Kbytes I/O + 256 bytes Periféricos)

8M ROM → 4 chips de 2 Mbytes

6M RAM → No es posible usar 4 chips de 2 Mbytes. Excedería la capacidad de direccionamiento

4 chips de 1 Mbytes + 4 chips de 512Kbytes

2. La Tabla D.9 y la Tabla D.10 muestran la solución del Problema 4.10.2.

TABLA D.9. DIRECCIONES DE INICIO Y FIN DE CADA BLOQUE.

Direcciones	Chips			
0x0 0x7F:FFFF	ROM0	ROM1	ROM2	ROM3
0x80:0000 0xBF:FFFF	RAM0	RAM1	RAM2	RAM3
0xC0:0000 0xDF:FFFF	RAM4	RAM5	RAM6	RAM7
0xE0:0000 0XE0:3FFF	I/O			
0xE0:4000 0XE0:4OFF	PER0	PER1	PER2	PER3

TABLA D.10. DIRECCIONES BASE DE LOS PERIFÉRICOS.

A[23:20]	A[19:16]	A[15:12]	A[11:8]	A[7:4]	A[3:0]	Chips
1100	0000	0100	0000	0000	0000	Per0
1100	0000	0100	0000	0000	0001	Per1
1100	0000	0100	0000	0000	0010	Per2
1100	0000	0100	0000	0000	0011	Per3

3. La Tabla D.11 y la Figura D.15 muestran la solución del Problema 4.10.3.

TABLA D.11. TABLA DE DECODIFICACIÓN.

A[23:20]	A[19:16]	A[15:12]	A[11:8]	A[7:4]	A[3:0]
xxxx	xxxx	xxxx	xxxx	xxxx	xxxx

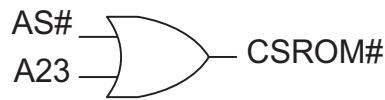


FIGURA D.15. LÓGICA DE SELECCIÓN.

4. La Figura D.15 muestra la solución del Problema 4.10.4.

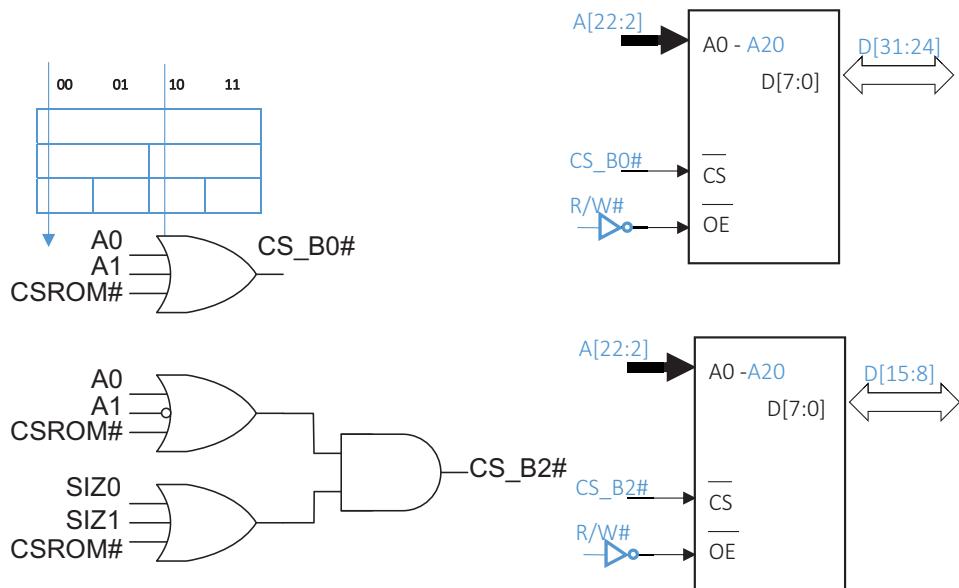
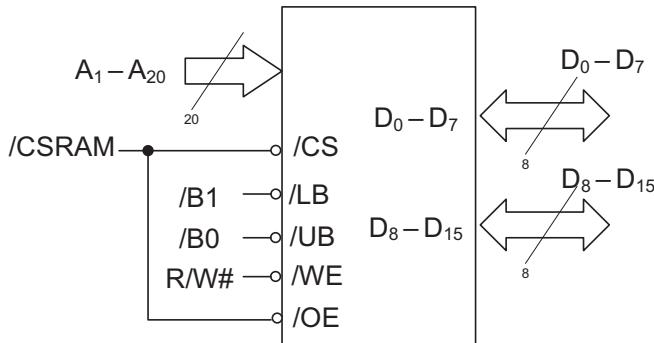


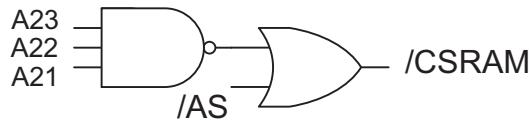
FIGURA D.16. CONEXIÓN DE LOS BANCOS DE MEMORIA 0 Y 2.

SOLUCIÓN AL PROBLEMA 4.11.

1. La Figura D.17 muestra la solución requerida.

**FIGURA D.17. BLOQUE DE MEMORIA REQUERIDO.**

2. La Figura D.18 muestra la generación del /CSRAM.

**FIGURA D.18. GENERACIÓN DE /CSRAM.****SECCIÓN 3. DISEÑO DE SISTEMAS DE MEMORIA CON EL EMC****SOLUCIÓN AL PROBLEMA 4.12.**

1. Dado que se debe segmentar la memoria en **4 bancos**, son necesarios:
 - 4 chips de SRAM (4x4Mx8) aunque la memoria total utilizada sean 16 Mbytes en lugar de 4 MB
 - 4 chips de FLASH (4x8Mx8) 32Mbytes
 - Los cuatro periféricos indicados de 128K
2. La memoria SRAM y FLASH pueden colocarse en un mismo subespacio de los 4 proporcionados por el EMC ya que en ambos casos el tamaño a acceder es de hasta 32 bits (la SRAM con accesos en 8, 16 y 32, y la FLASH siempre en 32 bits -4 bancos siempre-). Por ejemplo, en la zona seleccionada con CS0#. La Tabla D.12 muestra la solución para los bloques de memoria.

TABLA D.12. DIRECCIONES DE INICIO Y FIN DE CADA CHIP DE MEMORIA.

	Banco (chip) 0	Banco (chip) 1	Banco (chip) 2	Banco (chip) 3
FLASH	8000 0000 h	8000 0001 h	8000 0002 h	8000 0003 h
	81FF FFFC h	81FF FFFD h	81FF FFFE h	81FF FFFF h
SRAM	8200 0000 h	8200 0001 h	8200 0002 h	8200 0003 h
	82FF FFFC h	82FF FFFD h	82FF FFFE h	82FF FFFF h
Zona Vacía	8300 0000 h			83FF FFFF h

Los periféricos se colocarán en otro subespacio configurado en accesos a 8 bits (1 solo banco) para que las direcciones sean consecutivas. Por ejemplo, con CS1# (EMC_CS1#). La Tabla D.13 muestra la solución.

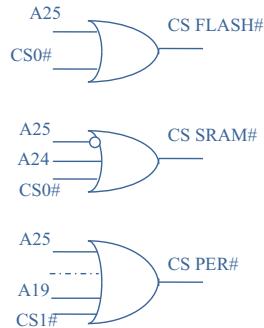
TABLA D.13. DIRECCIONES DE INICIO Y FIN DE LOS PERIFÉRICOS.

PER 1	PER 2	PER 3	PER 4
9000 0000 - 9001 FFFF	9002 0000 - 9003 FFFF	9004 0000 - 9005 FFFF	9006 0000 - 9007 FFFF

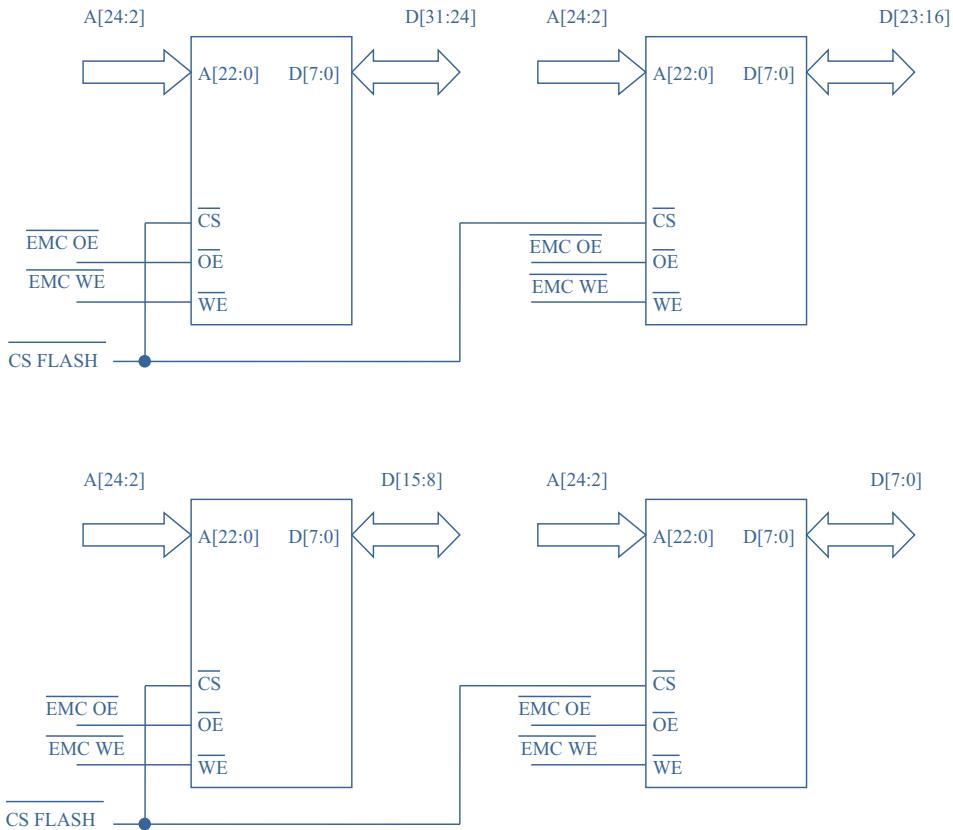
3. El registro *Memory Width* debe configurarse como sigue: Zona 0 [10]; Zona 1 [00]
4. La Tabla D.14 y la Figura D.19 muestra la solución del Problema 4.12.4

TABLA D.14. TABLA DE DECODIFICACIÓN.

A ₂₅ A ₂₄	A ₂₃ A ₂₂ A ₂₁ A ₂₀	A ₁₉ A ₁₈ A ₁₇ A ₁₆	A ₁₅A ₀	Sistema
0 X	0 0 0 X	X X X X	X ... X	FLASH, EMC_CS#0
1 0	0 0 0 X	X X X X	X ... X	SRAM, EMC_CS#0
0 0	0 0 0 0	0 X X X	X ... X	PER, EMC_CS#1

**FIGURA D.19. LÓGICA DE SELECCIÓN.**

5. La Figura D.20 muestra la conexión de la memoria FLASH.

**FIGURA D.20. CONEXIÓN DEL SISTEMA DE MEMORIA FLASH.**

SOLUCIÓN AL PROBLEMA 4.13.

Existen varias soluciones correctas. La Figura D.21 y Figura D.22 muestran una de ellas para cada espacio de memoria.

No volátil: STATICCONF1[1:0]=0x2

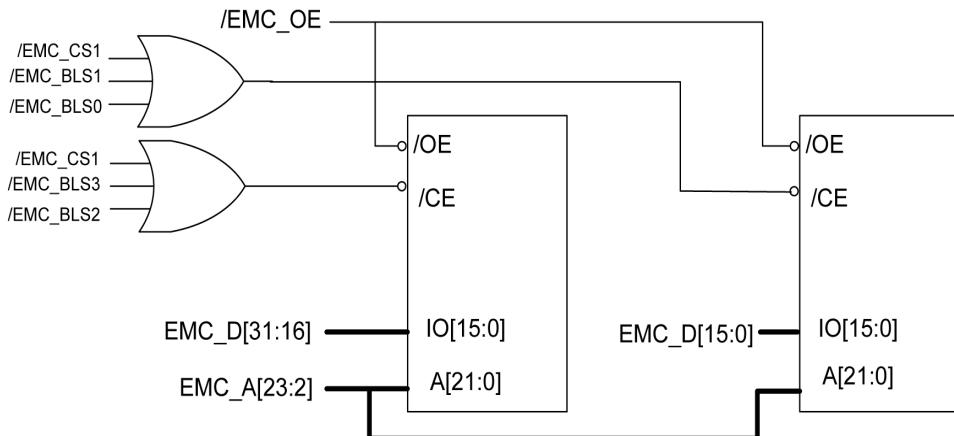


FIGURA D.21. CONEXIÓN DE LAS MEMORIAS NO VOLÁTILES (SST38VF6401).

Volátil: STATICCONF2[1:0]=0x2

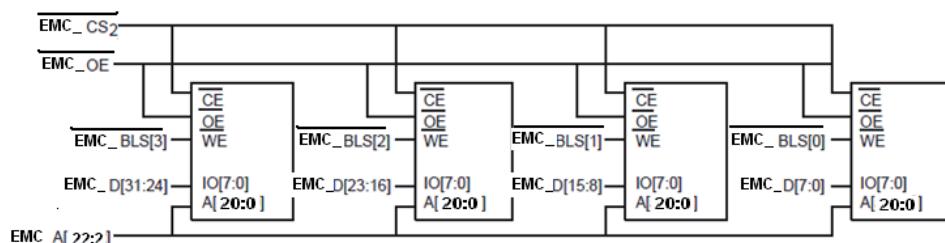


FIGURA D.22. CONEXIÓN DE LAS MEMORIAS VOLÁTILES (BS62LV1600).

SOLUCIÓN AL PROBLEMA 4.14.

- La Tabla D.15 muestra la solución del Problema 4.14.1.

TABLA D.15. CHIPS Y TAMAÑO.

	Nº chips	Tamaño de cada chip
ROM	4 (o 1)	4 de 2Mbytes (o 1 de 2Mx32)
RAM	4	8Mbytes
Periféricos	8	256Kx32 (equivale a 1Mbyte)

- La Tabla D.16 muestra la solución al Problema 4.14.2.

TABLA D.16. DIRECCIONES DE INICIO Y FIN DE CADA CHIP.

Bloque	Dirección Base	Dirección final
ROM		
1	0x9A00_0000	0x9A7F_FFFC
2	0x9A00_0001	0x9A7F_FFFD
3	0x9A00_0002	0x9A7F_FFFE
4	0x9A00_0003	0x9A7F_FFFF
RAM		
1	0x9800_0000	0x99FF_FFFC
2	0x9800_0001	0x99FF_FFFD
3	0x9800_0002	0x99FF_FFFE
4	0x9800_0003	0x99FF_FFFF
Periféricos		
0	0X9A80_0000	0X9A8F_FFFF
1	0X9A90_0000	0X9A9F_FFFF
2	0X9AA0_0000	0X9AAF_FFFF
3	0X9AB0_0000	0X9ABF_FFFF
4	0X9AC0_0000	0X9ACF_FFFF
5	0X9AD0_0000	0X9ADF_FFFF
6	0X9AE0_0000	0X9AEF_FFFF
7	0X9AF0_0000	0X9AFF_FFFF

3. La Figura D.23 muestra la lógica de selección.

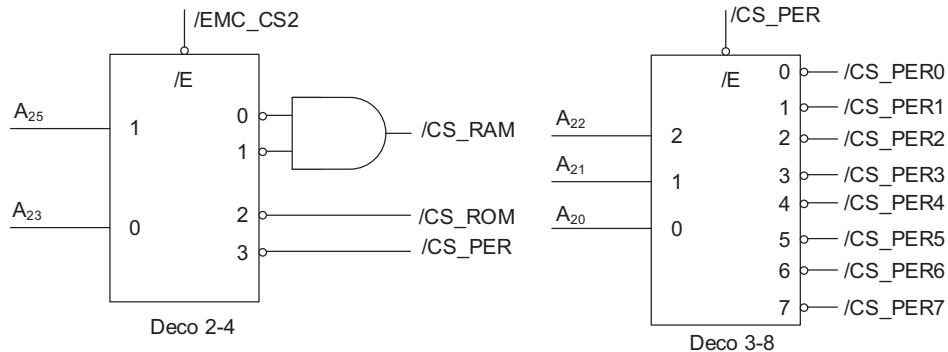


FIGURA D.23. LÓGICA DE SELECCIÓN.

4. La Figura D.24 muestra la conexión de los periféricos.

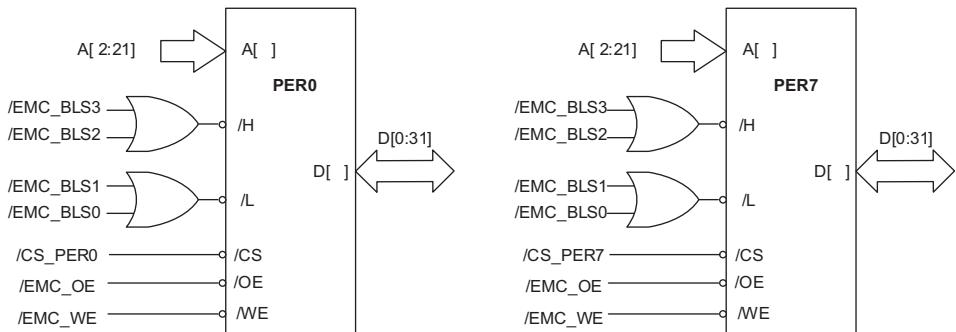


FIGURA D.24. CONEXIÓN DE LOS PERIFÉRICOS 0 Y 7.

SOLUCIÓN AL PROBLEMA 4.15.

1. La Tabla D.17 muestra la solución del Problema 4.15.1.

TABLA D.17. CHIPS Y TAMAÑO.

	Nº chips	Tamaño de cada chip
ROM	4	1Mbyte
RAM	4	2Mbytes
Periféricos 8 bits	4	256Kbytes

- 2 CS1#, dirección base 0x9000_0000 del mapa del LPC1788. La Tabla D.18 muestra la solución del Problema 4.15.2.

TABLA D.18. DIRECCIONES DE INICIO Y FIN DE CADA BLOQUE.

Bloque	Dirección Base	Dirección final
ROM (4Mbytes)		
0	0x9080_0000	0x90BF_FFFC
1	0x9080_0001	0x90BF_FFFD
2	0x9080_0002	0x90BF_FFFE
3	0x9080_0003	0x90BF_FFFF
RAM (8Mbytes)		
0	0x9000_0000	0x907F_FFFC
1	0x9000_0001	0x907F_FFFD
2	0x9000_0002	0x907F_FFFE
3	0x9000_0003	0x907F_FFFF
Periféricos (1Mbyte)		
0	0x90C0_0000	0x90CF_FFFC
1	0x90C0_0001	0x90CF_FFFD
2	0x90C0_0002	0x90CF_FFFE
3	0x90C0_0003	0x90CF_FFFF

3. Hay que realizarlo con **decodificación completa**. La Figura D.25 muestra dos posibles soluciones.

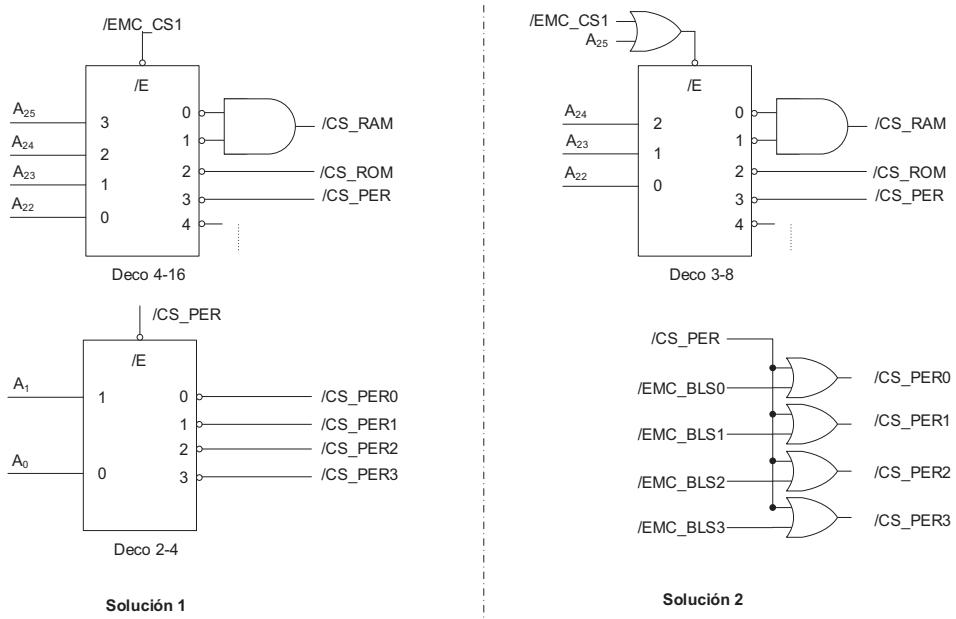


FIGURA D.25. LÓGICA DE SELECCIÓN.

4. La Figura D.26 muestra la conexión de los bancos 0 y 3.

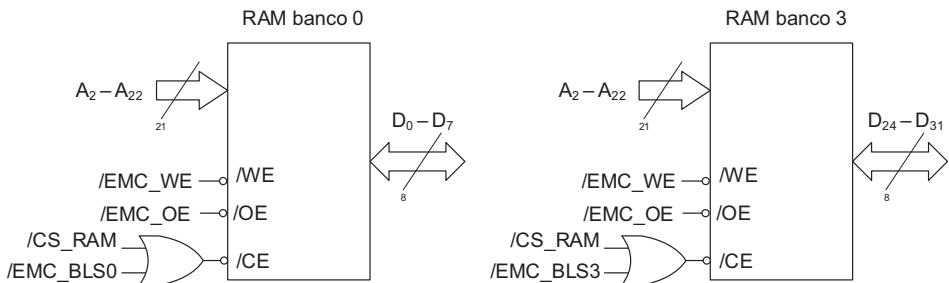


FIGURA D.26. CONEXIÓN AL MICROPROCESADOR DE LOS BANCOS 0 Y 3.

SOLUCIÓN AL PROBLEMA 4.16.

1. La Tabla D.19 muestra la solución al Problema 4.16.1.

TABLA D.19. CHIPS Y TAMAÑO.

Tipo de dispositivo	Nº chips	Tamaño de cada chip
EEPROM	4	8Mbytes
SRAM	4	4Mbytes
FLASH	4	4Mbytes

2. La Tabla D.20 muestra la solución del Problema 4.16.2.

TABLA D.20. DIRECCIONES DE INICIO Y FIN DE LOS ELEMENTOS DEL MAPA DE MEMORIA.

Tipo de dispositivo	Dirección Base	Dirección final
EEPROM (8Mbytes)		
0	0x9800_0000	0x99FF_FFFC
1	0x9800_0001	0x99FF_FFFD
2	0x9800_0002	0x99FF_FFFE
3	0x9800_0003	0x99FF_FFFF
SRAM (4Mbytes)		
0	0x9A00_0000	0x9AFF_FFFC
1	0x9A00_0001	0x9AFF_FFFD
2	0x9A00_0002	0x9AFF_FFFE
3	0x9A00_0003	0x9AFF_FFFF
FLASH (4Mbytes)		
0	0x9B00_0000	0x9BFF_FFFC
1	0x9B00_0001	0x9BFF_FFFD
2	0x9B00_0002	0x9BFF_FFFE
3	0x9B00_0003	0x9BFF_FFFF

3. La Figura D.27 muestra la lógica de selección.

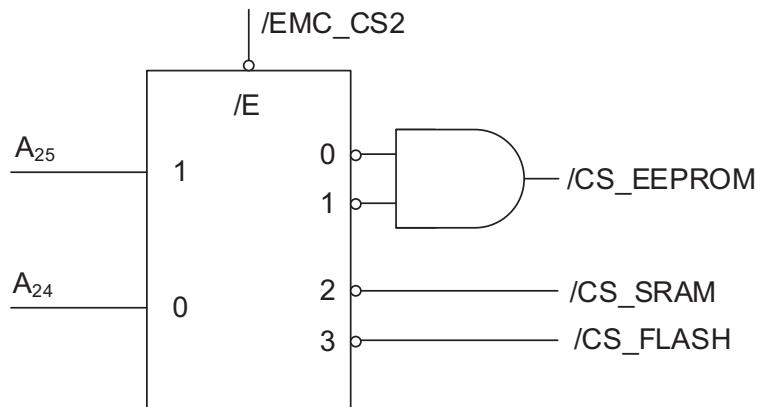
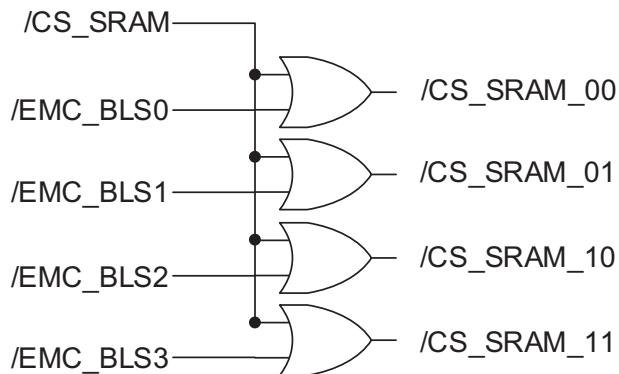


FIGURA D.27. LÓGICA DE SELECCIÓN.

4. La Figura D.28 muestra la conexión al microprocesador de todos los chips de SRAM.



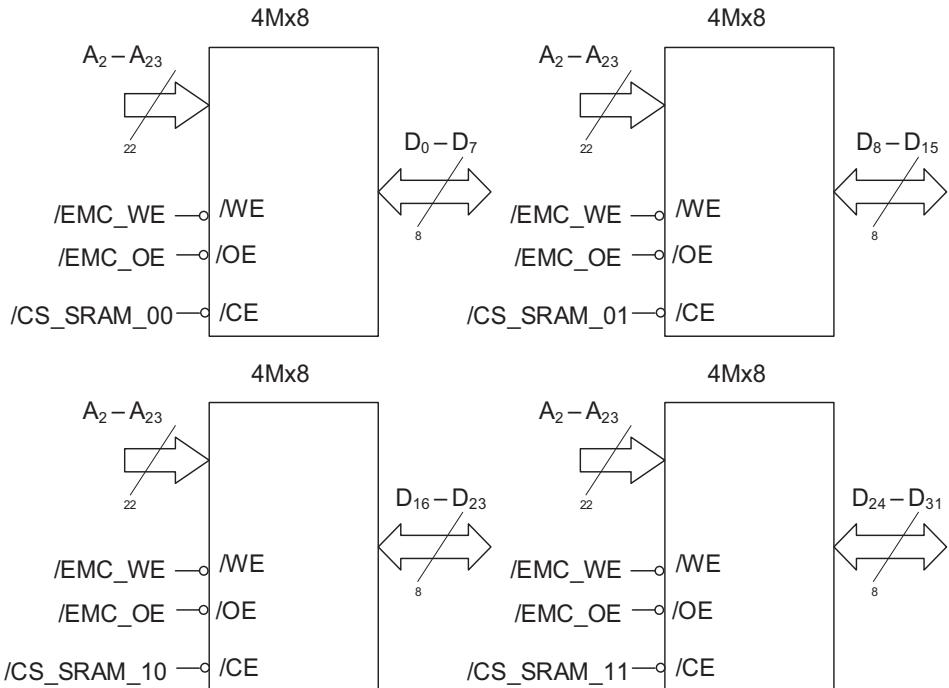


FIGURA D.28. CONEXIÓN AL MICROPROCESADOR.

SOLUCIÓN AL PROBLEMA 4.17.

1. La Figura D.29 muestra la conexión de la memoria al microprocesador.

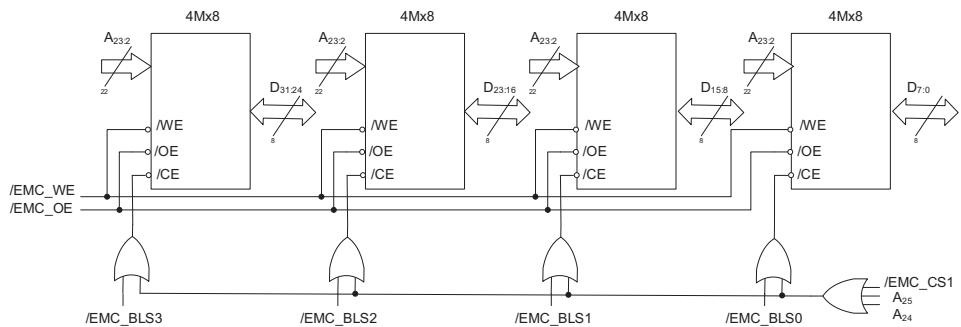


FIGURA D.29. CONEXIÓN DE LOS CHIPS DE MEMORIA AL MICROPROCESADOR.

2. La Tabla D.21 muestra la solución del Problema 4.17.2.

TABLA D.21. DIRECCIONES DE INICIO Y FIN DE LOS PERIFÉRICOS.

Sistema	Dirección inicio (hex)	Dirección fin (hex)
Per16_0	0x9800_0000	0x9800_003F
Per16_1	0x9800_0040	0x9800_007F
Per8_0	0x9800_0080	0x9800_0086
Per8_1	0x9800_0081	0x9800_0087
Per8_2	0x9800_0088	0x9800_008E
Per8_3	0x9800_0089	0x9800_008F

Distribución	
Per16_0	
Per16_1	
Per8_0	Per8_1
Per8_2	Per8_3

SOLUCIÓN AL PROBLEMA 4.18.

1. La Tabla D.22 y la Figura D.30 muestran la solución de este apartado.

TABLA D.22. TABLA DE DECODIFICACIÓN.

A ₃₁ ... A ₂₈	A ₂₇ ... A ₂₄	A ₂₃ ... A ₂₀	A ₁₉ ... A ₁₆	A ₁₅ ... A ₁₂	A ₁₁ ... A ₈	A ₇ ... A ₄	A ₃ ... A ₀
1 0 0 1	1 0 1 1	1 1 1 1	1 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 X X X

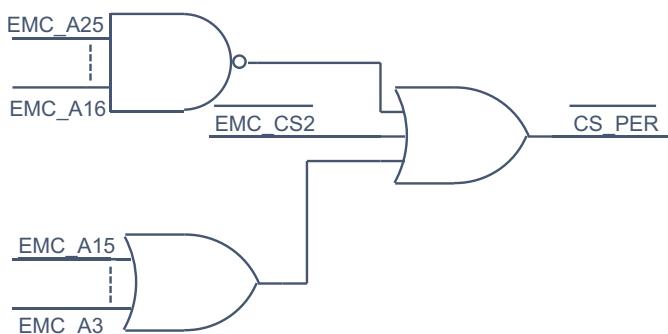
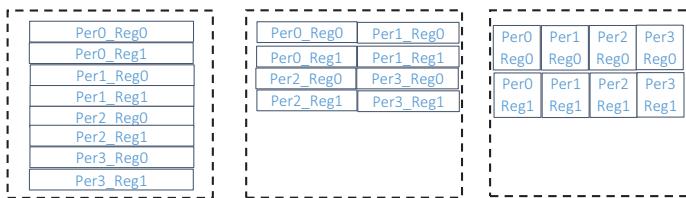


FIGURA D.30. LÓGICA DE SELECCIÓN DE LOS PERIFÉRICOS.

2 La Tabla D.23 muestra las direcciones de los registros.

TABLA D.23. DIRECCIONES DE LOS REGISTROS.

Periférico	Registro	MW=00	MW=01	MW=10
Per0	Reg0	0x9BFF_00000	0x9BFF_00000	0x9BFF_00000
	Reg1	0x9BFF_00001	0x9BFF_00002	0x9BFF_00004
Per1	Reg0	0x9BFF_00002	0x9BFF_00001	0x9BFF_00001
	Reg1	0x9BFF_00003	0x9BFF_00003	0x9BFF_00005
Per2	Reg0	0x9BFF_00004	0x9BFF_00004	0x9BFF_00002
	Reg1	0x9BFF_00005	0x9BFF_00006	0x9BFF_00006
Per3	Reg0	0x9BFF_00006	0x9BFF_00005	0x9BFF_00003
	Reg1	0x9BFF_00007	0x9BFF_00007	0x9BFF_00007



3. La Figura D.31 muestra la conexión de los periféricos.

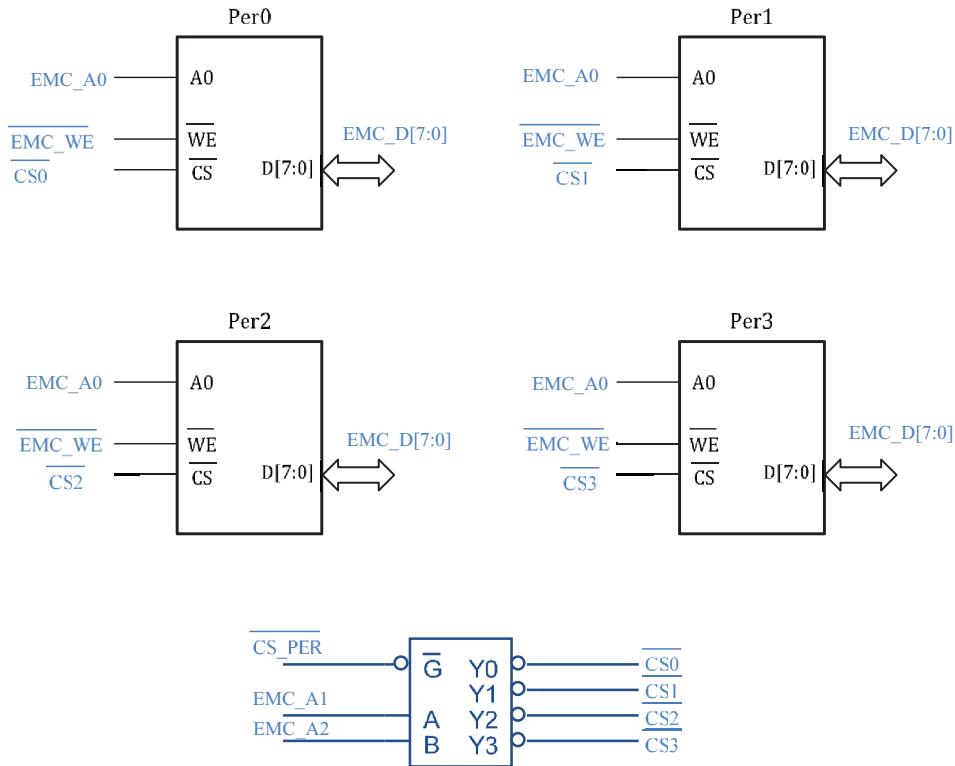


FIGURA D.31. CONEXIÓN DE LOS PERIFÉRICOS.

SOLUCIÓN AL PROBLEMA 4.19.

1. La Tabla D.24 y la Tabla D.25 muestran la solución del Problema 4.19.1.

TABLA D.24. CONFIGURACIÓN DEL CAMPO MW.

CSx	MW	
CS0	01	16 bits
CS3	00	8 bits

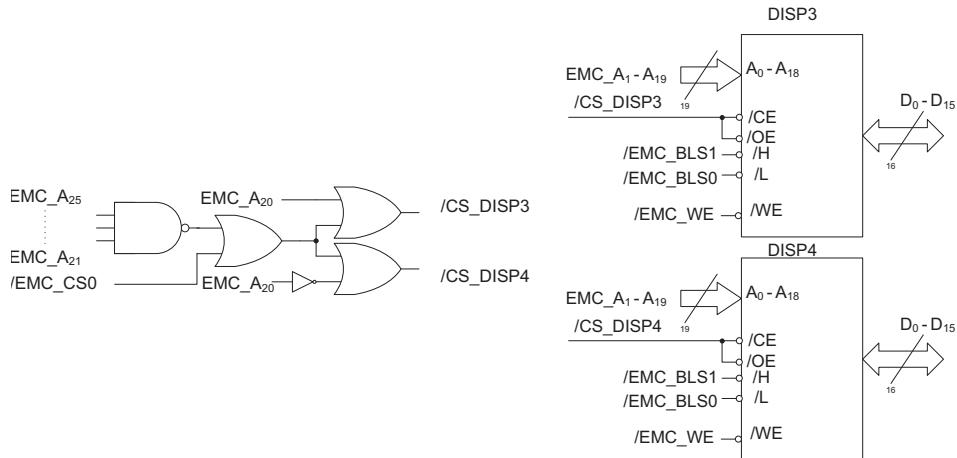
TABLA D.25. DIRECCIONES DE LOS ELEMENTOS.

Elemento	Dirección Base	Dirección final
DISP0	9C00_0000	9C03_FFFF
DISP1	9C04_0000	9C07_FFFF
DISP2	9C08_0000	9C0B_FFFF
DISP3	83E0_0000	83EF_FFFF
DISP4	83F0_0000	83FF_FFFF

2. La Tabla D.26 y la Figura D.32 muestran la solución del Problema 4.19.2.

TABLA D.26. TABLA DE DECOFIGACIÓN.

A25 A24 A23 A22 A21 A20	
1 1 1 1 1 0	DISP3
1 1 1 1 1 1	DISP4

**FIGURA D.32. CONEXIÓN DE LOS DISPOSITIVOS DISP3 Y DISP4.**

SOLUCIÓN AL PROBLEMA 4.20.

La Figura D.33 muestra la solución al Problema 4.20.1, y la Tabla D.27 al Problema 4.20.2.

TABLA D.27. RANGO DE DIRECCIONES DE LOS CHIPS.

Memoria	Dir. Inicio (hex)	Dir. Fin (hex)
A	0x9C00_0000	0x9cff_FFFF
B	0x9D00_0000	0x9dff_FFFF
C	0x9E00_0000	0x9eff_FFFF
D	0x9F00_0000	0x9fff_FFFF

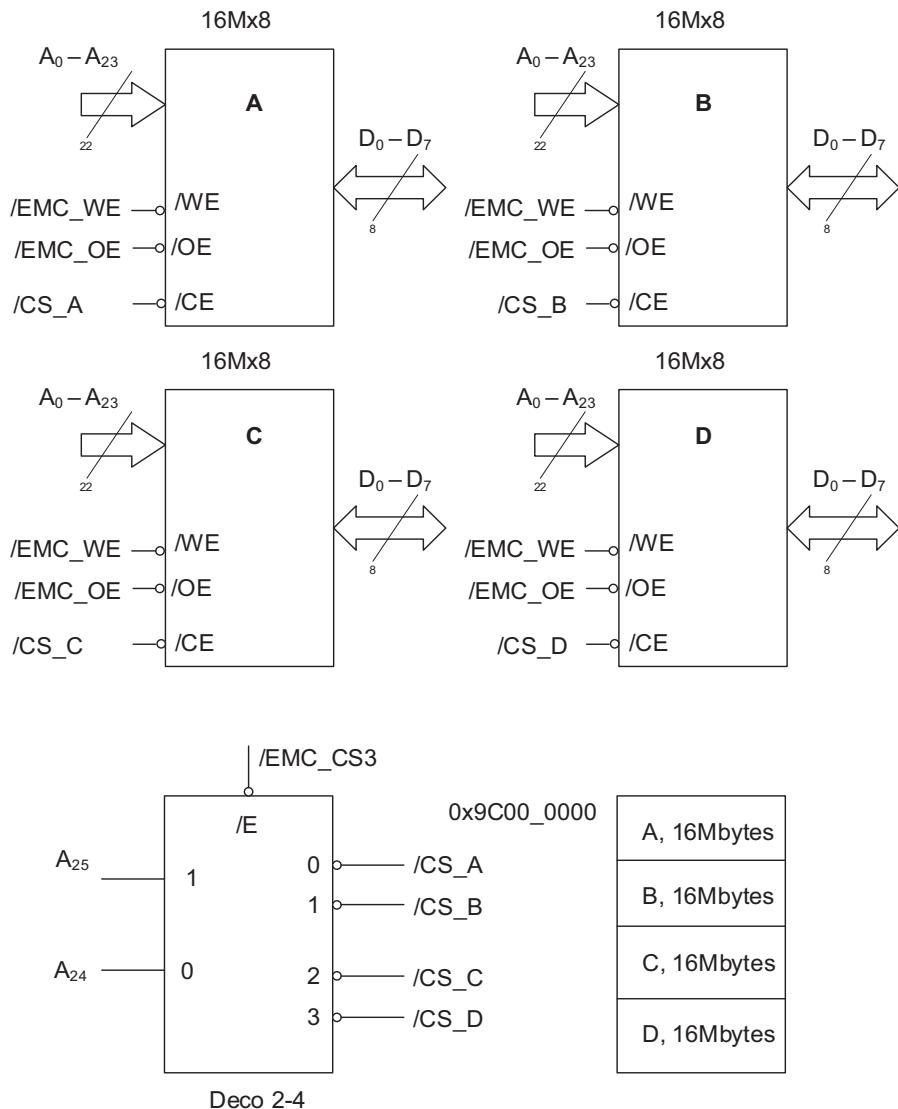


FIGURA D.33. CONEXIONADO DE LAS MEMORIAS.

SECCIÓN 4. ANÁLISIS DE CRONOGRAMAS DE ACCESO

SOLUCIÓN AL PROBLEMA 4.21.

- La Figura D.34 muestra el cronograma.

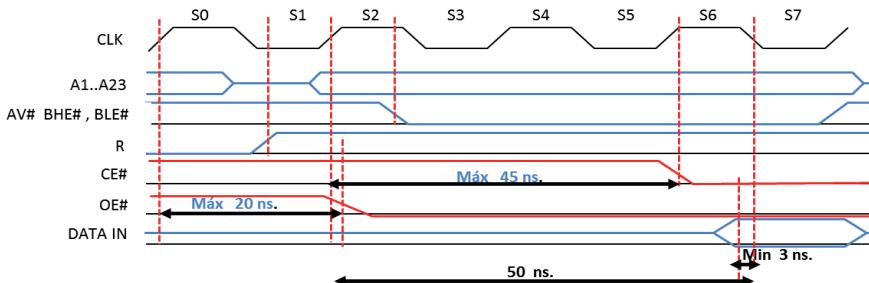


FIGURA D.34. CRONOGRAMA DEL PROBLEMA 4.21.

- A continuación, se realiza el estudio de los tiempos de selección.

$$T_{Sel\ Min\ CE} = 50 - 45 - 3 = 2\ ns$$

$$T_{Acc\ Máx\ CE} = 90\ ns$$

$T_{Sel\ Min\ CE} - T_{Acc\ Máx\ CE} = 2 - 90 = -88\ ns \rightarrow$ **Se necesitan 5 ciclos de espera (5x20 ns = 100 ns)**

$$T_{Sel\ Min\ OE} - T_{Acc\ Máx\ OE} = 47 - 35 = 12\ ns \rightarrow$$
 No se necesitan ciclos de espera

SOLUCIÓN AL PROBLEMA 4.22.

- La Figura D.35 muestra el cronograma.

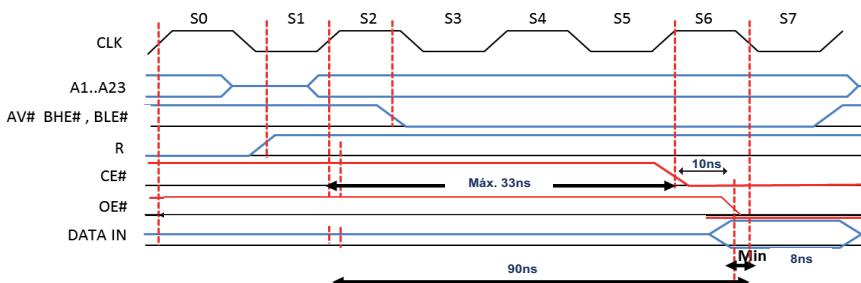


FIGURA D.35. CRONOGRAMA DEL PROBLEMA 4.22.

2. A continuación, se realiza el estudio de los tiempos de selección.

$$T_{Sel \text{ Mín CE}} = 90 - 33 - 8 = 49 \text{ ns}$$

$$T_{Acc \text{ Máx CE}} = 70 \text{ ns}$$

$$T_{Sel \text{ Min CE}} - T_{Acc \text{ Máx CE}} = 49 - 70 = -21 \text{ ns} \rightarrow \text{Se necesitan 2 ciclos de espera}$$

(2x40 ns = 80 ns)

$$T_{Sel \text{ Min OE}} - T_{Acc \text{ Máx OE}} = 39 - 30 = 9 \text{ ns} \rightarrow \text{No se necesitan ciclos de espera}$$

SOLUCIÓN AL PROBLEMA 4.23.

1 La Figura D.36 muestra el cronograma.

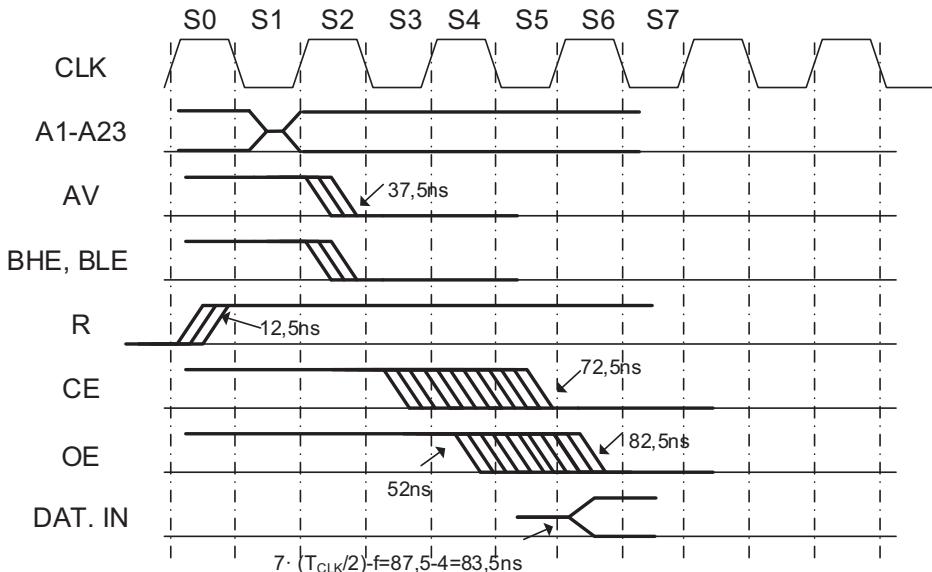


FIGURA D.36. CRONOGRAMA DEL PROBLEMA 4.23.

2. A continuación, se realiza el estudio de los tiempos de selección.

$$T_{ACC-\text{maxCE}} = 70 \text{ ns};$$

$$T_{ACC-\text{maxOE}} = 30 \text{ ns};$$

$$T_{SelCE} - T_{ACC-\text{maxCE}} = 87,5 - 72,5 - 4 = 11 \text{ ns}; \leftarrow \text{más restrictivo}$$

$$T_{SelOE} - T_{ACC-\text{maxOE}} = 87,5 - 30 - 4 = 53 \text{ ns}.$$

$$T_{SelCE} = 87,5 - 72,5 - 4 = 11 \text{ ns}$$

$$T_{SelOE} = 87,5 - 30 - 4 = 53 \text{ ns}$$

$$\text{ciclos de reloj} = \frac{59}{T_{CLK}} = \frac{59}{25} = 2,3 \rightarrow 3 \text{ ciclos}$$

SOLUCIÓN AL PROBLEMA 4.24.

1. La Figura D.37 muestra la solución del Problema 4.24.1.

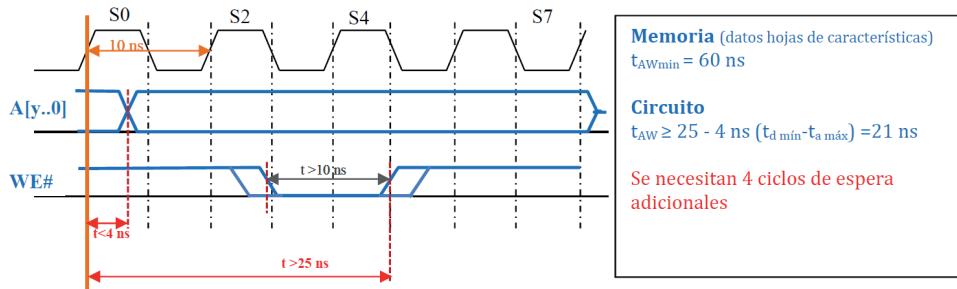


FIGURA D.37. CRONOGRAMA DEL PROBLEMA 4.24.1.

2. La Figura D.38 muestra la solución del Problema 4.24.2.

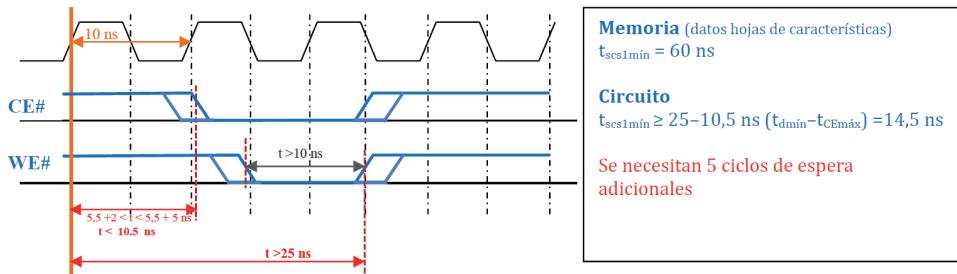


FIGURA D.38. CRONOGRAMA DEL PROBLEMA 4.24.2.

3. La Figura D.39 muestra la solución del Problema 4.24.3.

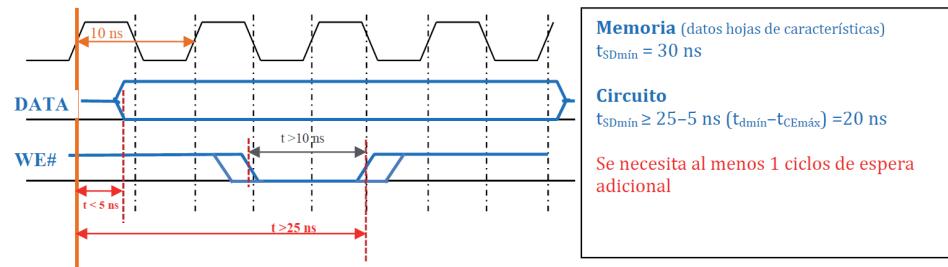


FIGURA D.39. CRONOGRAMA DEL PROBLEMA 4.24.3.

SOLUCIÓN AL PROBLEMA 4.25.

La Figura D.40 muestra el cronograma, y a continuación se realiza el cálculo de los tiempos de selección.

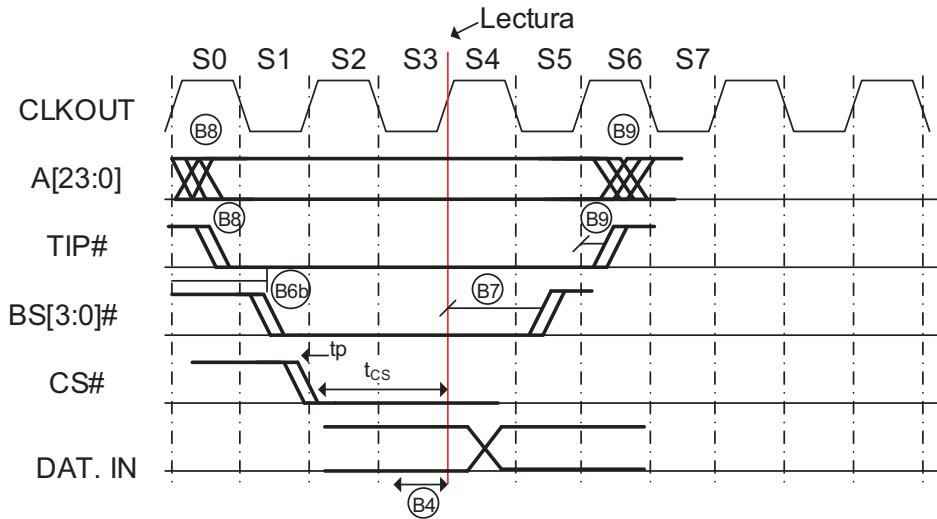


FIGURA D.40. CRONOGRAMA DEL PROBLEMA 4.25.

$$t_{B6B} = 0,5 \cdot t_{CYC} + 10 = 0,5 \cdot 40 + 10 = 30\text{ns}$$

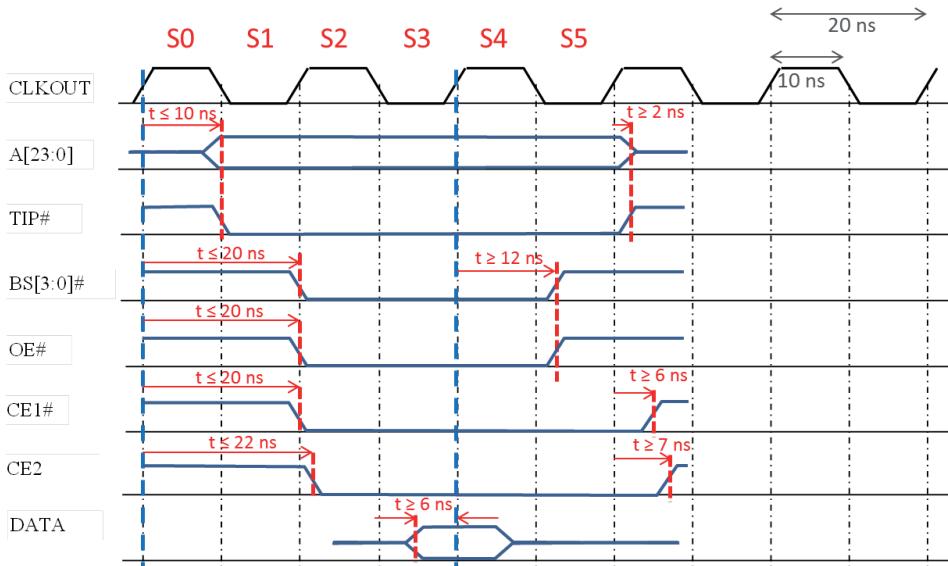
$$T_{CLK} = t_{CYC} = 40\text{ns}$$

$$t_{CS} = 4 \cdot \frac{T_{CLK}}{2} - t_{B6bmx} - t_{pmx} - t_{B4} = 4 \cdot 20 - 30 - 1 - 6 = 43\text{ns}$$

Para que no se requieran ciclos de espera el **tiempo de acceso de la memoria debe ser inferior a 43ns**.

SOLUCIÓN AL PROBLEMA 4.26.

1. La Figura D.41 muestra el cronograma de acceso,

**FIGURA D.41. CRONOGRAMA DEL PROBLEMA 4.26.**

2. La Tabla D.28 muestra los tiempos de selección y acceso. Más abajo se encuentra el desarrollo.

TABLA D.28. TIEMPOS DE SELECCIÓN Y ACCESO.

Tiempo	Valor selección (Microprocesador)	Valor acceso (Memoria)
Dirección válida a dato válido	24 ns	70 ns
Chip enable a dato válido	12 ns	70 ns
Output enable a dato válido	14 ns	25 ns
Byte select a dato válido	14 ns	70 ns

Cálculos tiempos selección:

$$t_{SEL_{ADD-data}} \geq 4 \frac{T}{2} - t_{ADD_{máx}} - t_{data_{mín}} = 40 - 10 - 6 = 24 \text{ ns}$$

$$t_{SEL_{CE\#-data}} \geq 4 \frac{T}{2} - t_{CE\#_{máx}} - t_{data_{mín}} = 40 - 22 - 6 = 12 \text{ ns}$$

$$t_{SEL_{OE\#-data}} \geq 4 \frac{T}{2} - t_{OE\#_{máx}} - t_{data_{mín}} = 40 - 20 - 6 = 14 \text{ ns}$$

$$t_{SEL_{BS\#-data}} \geq 4 \frac{T}{2} - t_{BS\#_{máx}} - t_{data_{mín}} = 40 - 20 - 6 = 14 \text{ ns}$$

En ninguno de los casos hay tiempo suficiente para cumplir el tiempo de acceso de la memoria. El caso más desfavorable es el de CE#, ya que es el que menos tiempo de acceso deja.

$$t_{SEL_{CE}} - t_{acc_{CE\#}} = 12 - 70 = -58 \text{ ns} \quad \text{Se necesitan 3 ciclos de espera} \left(\frac{58}{T_{CLK}} = \frac{58}{20} = 2,9 \approx 3 \right)$$

SOLUCIÓN AL PROBLEMA 4.27.

- La Figura D.42 muestra el análisis de la señal que controla la operación, en este caso /WE.

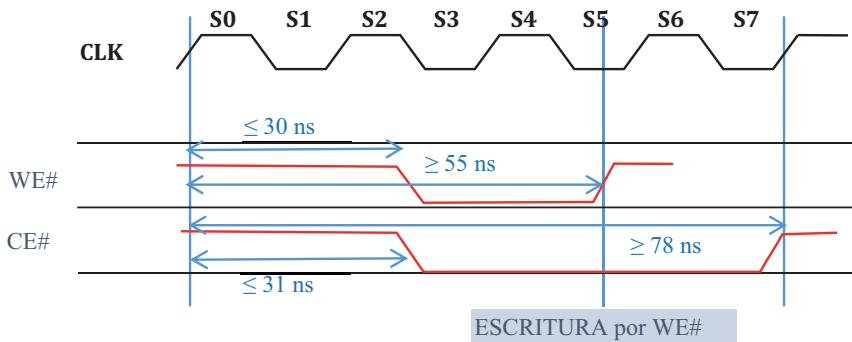


FIGURA D.42. CRONOGRAMA DEL PROBLEMA 4.27.1.

2. La Figura D.43 muestra el cronograma y la Tabla D.29 los tiempos resultantes.

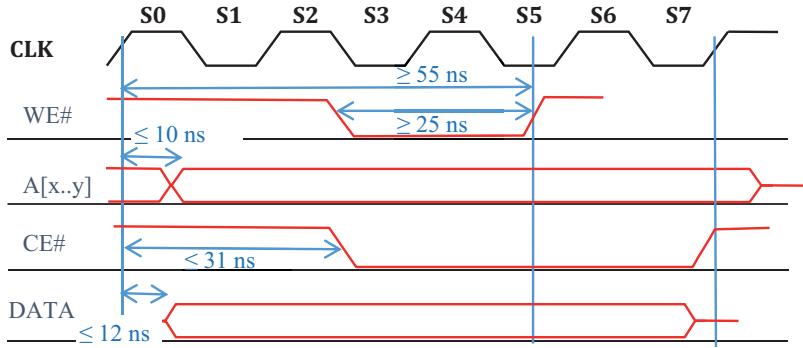


FIGURA D.43. CRONOGRAMA DEL PROBLEMA 4.27.2.

TABLA D.29. TIEMPOS DE SELECCIÓN Y ACCESO.

Tiempo	Valor selección (Microporcesador)	Valor acceso (Memoria)
Dirección válida a fin de escritura	≥ 45 ns	45 ns _{mín}
<i>Chip enable</i> a fin de escritura	≥ 24 ns	45 ns _{mín}
Dato válido a fin de escritura	≥ 43 ns	25 ns _{mín}
Ancho del pulso de escritura	≥ 25 ns	40 ns _{mín}

Como muestra la tabla, hay dos condiciones que no se cumplen (*Chip enable* a fin de escritura y ancho de pulso de escritura). La más restrictiva es la primera, que requiere al menos 21ns (45-24), lo que implica tener que introducir dos ciclos de espera (ya que el $T_{CLK}=20$ ns y se requieren 21ns).

SOLUCIÓN AL PROBLEMA 4.28.

1. La Figura D.44 muestra el cronograma de acceso.

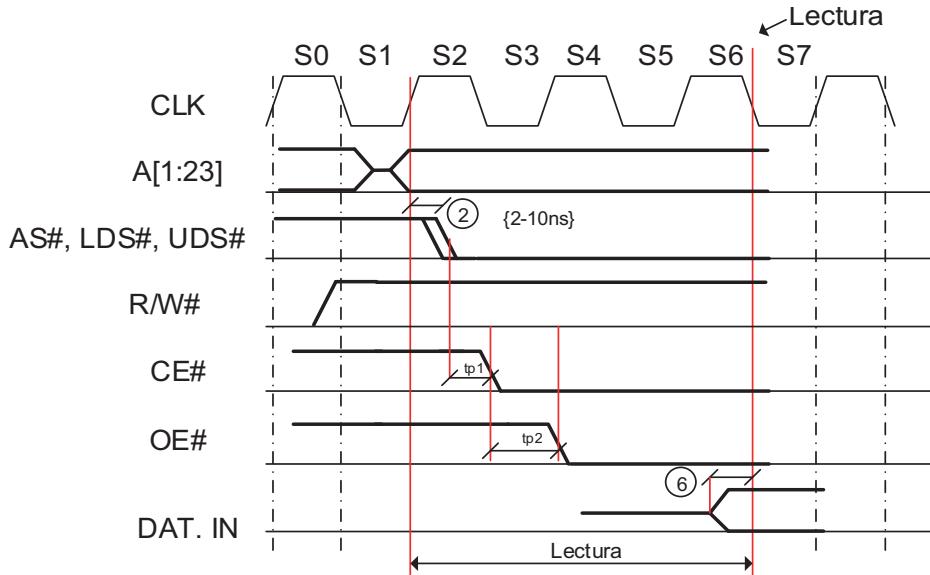


FIGURA D.44. CRONOGRAMA DEL PROBLEMA 4.28.

2. A continuación, se realiza el análisis de los tiempos de selección.

Tiempo de acceso a la memoria, $t_{acc} = 79\text{ns}$ (t_{CE})

Tiempo desde OE# > 30ns

$$t_{selecc\mu P} = 5 \cdot \frac{T_{CLK}}{2} - t_{2mx} - t_{p1} = (37 - t_{p1})\text{ns}$$

$$t_{selecc\mu P} + T_{CLK} > t_{acc} \Rightarrow (37 - t_{p1}) + 20 > 70 \Rightarrow t_{p1} < 37 + 20 - 70 = -13\text{ns} \rightarrow \text{IMPOSIBLE}$$

t_{p1} no puede ser negativo. Por tanto, no hay ningún valor de t_{p1} (ni de t_{p2}) que permitan realizar correctamente la operación con un solo ciclo de espera. Se requiere más de un ciclo de espera.

SECCIÓN 5. PROBLEMAS GLOBALES

SOLUCIÓN AL PROBLEMA 4.29.

1. A continuación, se realiza el análisis de los chips de memoria requeridos.
 ROM: programa (60Kbytes), Tabla de Vectores (0,5Kbytes), constantes (20Kbytes) → 80,5Kbytes

RAM: paso de parámetros (12Kbytes), lectura, escritura, pila (20Kbytes) → 32Kbytes

Implementación en 2 bancos:

- ROM: 128Kbytes → 2 chips de 64Kbytes → 2x64Kbytes
- RAM: 32Kbytes → 2 chips de 16Kbytes → 2x16Kbytes

2. La Tabla D.30 muestra el mapa funcional del sistema.

TABLA D.30. MAPA FUNCIONAL DEL SISTEMA.

Dirección	Función	Elemento
0x0_0000	T. Vectores	ROM
	Programa	
	Constantes	
0x1_FFFF	Libre	
		LIBRE
0xB_C000	E/S	E/s
0xB_FFFF	E/S	
0xC_0000	Parámetros;	RAM
0xF_FFFF	Pila	

ROM: 0x0_0000 – 0x1_FFFF → 128Kbytes

E/S: 0xB_C000 – 0xB_FFFF → 16Kbytes

RAM: 0xC_0000 – 0xF_FFFF → 256Kbytes

3. La Tabla D.31 muestra la tabla de decodificación y la Figura D.45 la lógica de selección que se deduce de la tabla.

Tabla D.31. Tabla de decodificación.

Bloque	A ₁₉ A ₁₈ A ₁₇ A ₁₆	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
ROM	0 0 0 X	XXXX	XXXX	XXXX	XXXX
E/S	1 0 1 1	1 1 XX	XXXX	XXXX	XXXX
RAM	1 1 XX	XXXX	XXXX	XXXX	XXXX

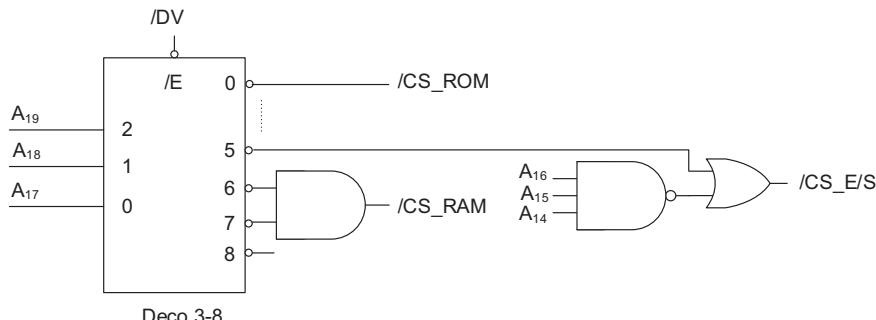


FIGURA D.45. LÓGICA DE SELECCIÓN.

4. La Figura D.46 muestra la solución del Problema D.46.4.

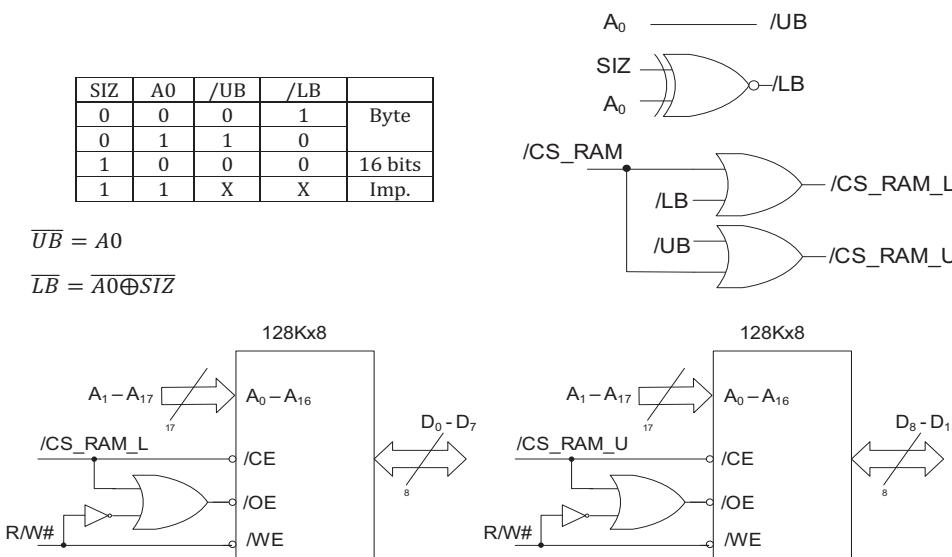


FIGURA D.46. CONEXIÓN DE LA MEMORIA RAM AL MICROPROCESADOR.

5. Tamaño de la memoria FLASH: $2^{22} \rightarrow 4\text{Mbytes Upper} / 4\text{ Mbytes Lower}; 2 \times 4\text{Mbytes}$

El tamaño de ROM requerido es de 128Kbytes, 2x64Kbytes. Para la implementación se deben conectar las líneas A1-A16 del microprocesador a las líneas A0-A15 de la memoria. El resto de las líneas, A16-A21, se fuerzan a 0. La línea A0 del microprocesador se utiliza para generar /UB y /LB.

6. La Figura D.47 muestra el cronograma de acceso. Se incluyen debajo el cálculo del tiempo de selección.

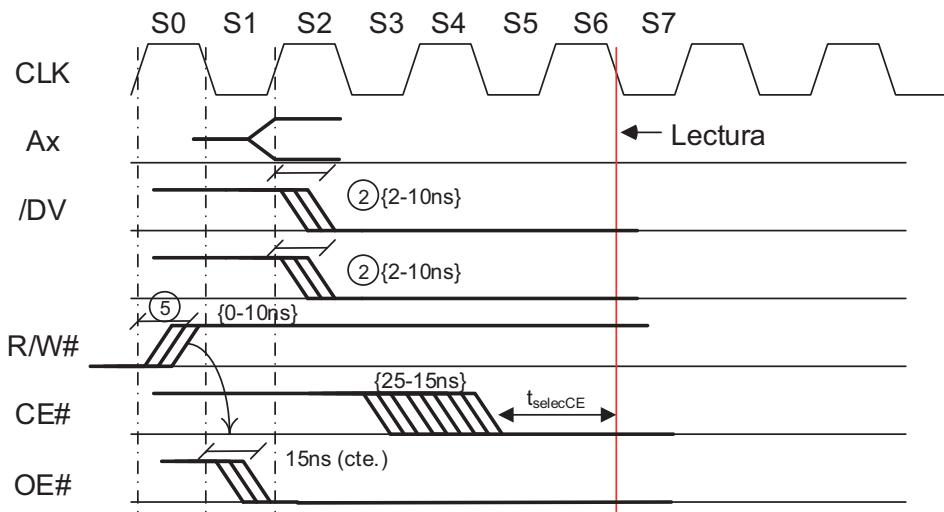


FIGURA D.47. CRONOGRAMA DEL PROBLEMA 4.29.6.

Tiempo de acceso de la memoria, $t_{acc} = 70\text{ns}$

Tiempo desde activación de CE# $\rightarrow 70\text{ns}$

Tiempo desde activación de OE# $\rightarrow 30\text{ns}$

$T_{CLK} = 20\text{ns}$

Como CE# se activa después de OE#, hay que verificar que cumpla la restricción desde la activación de CE#:

$$t_{selecCE} = 5 \cdot \frac{T_{CLK}}{2} - t_{2mx} - t_{LSmx} - t_6 = 5 \cdot 10 - 10 - 35 - 3 = 2\text{ns}$$

Como tselecCE tiene que ser superior a 70ns, se necesitan al menos 68ns
 → se requieren 4 ciclos de espera.

SOLUCIÓN AL PROBLEMA 4.30.

- La Figura D.48 muestra la solución del Problema 4.30.1.

1001	11xx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
------	------	------	------	------	------	------	------

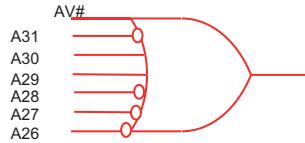


FIGURA D.48. GENERACIÓN DE LA SEÑAL $CS_{ext}\#$.

- La Tabla D.32 muestra la solución del Problema 4.30.2.

TABLA D.32. TABLA DE VERDAD PARA GENERAR LA SEÑAL $S_0B\#$.

SZ1	SZ0	A1	A0	$S_0B\#$
0	0	0	0	1
	0	1		1
	1	0		1
	1	1		0
0	1	0	0	1
	0	1		x
	1	0		0
	1	1		x
1	0	0	0	0
Resto				x
				x
				x
				x

		x	
	*	*	x
x	*	*	*
*	*	*	*

$$(S_0B\#)\# = SZ1 + A1 \cdot A0 + A1 \cdot SZ0 \rightarrow S_0B\# = SZ1\# \cdot (A1\# + A0\#) \cdot (A1\# + SZ0\#)$$

3. a) La Tabla D.33 muestra la solución del Problema 4.30.3a

TABLA D.33. DIRECCIONES DE INICIO Y FIN DE LOS BLOQUES DE MEMORIA.

RAM0	RAM1	RAM2	RAM3
RAM4	RAM5	RAM6	RAM7
Per0	Per1	Per2	Per3

Chip	Dirección inicio (hex)	Dirección fin (hex)
RAM0	0x9C00:0000	0x9C07:FFFC
RAM1	0x9C00:0001	0x9C07:FFFD
RAM2	0x9C00:0002	0x9C07:FFFE
RAM3	0x9C00:0003	0x9C07:FFFF
RAM4	0x9C08:0000	0x9C0F:FFFC
RAM5	0x9C08:0001	0x9C0F:FFFD
RAM6	0x9C08:0002	0x9C0F:FFFE
RAM1	0x9C08:0003	0x9C0F:FFFF
Per0	0x9C10:0000	0x9C11:FFFC
Per1	0x9C10:0001	0x9C11:FFFD
Per2	0x9C10:0002	0x9C11:FFFE
Per3	0x9C10:0003	0x9C11:FFFF

b) La Tabla D.34 y la Figura D.49 muestran la solución del Problema 4.30.3b.

TABLA D.34. DIRECCIONES DE INICIO Y FIN DE LOS BLOQUES DE MEMORIA.

1001	1100	0000 <u>0</u>	<u>0</u> XXX	XXXX	XXXX	XXXX	XXXX	RAMa
1001	1100	0000 <u>0</u>	<u>1</u> XXX	XXXX	XXXX	XXXX	XXXX	RAMb
1001	1100	000 <u>1</u>	<u>0</u> 00x	XXXX	XXXX	XXXX	XXXX	PER

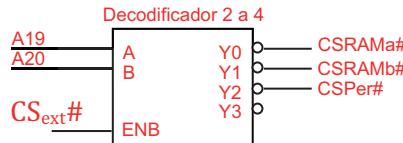


FIGURA D.49. GENERACIÓN DE LAS SEÑALES DE SELECCIÓN.

c) La Figura D.50 muestra la solución del Problema 4.30.3c.



FIGURA D.50. CONEXIÓN DE LA MEMORIA Y PERIFÉRICO AL MICROPROCESADOR.

4. La Figura D.51 y la Tabla D.35 muestran la solución del Problema 4.30.5.

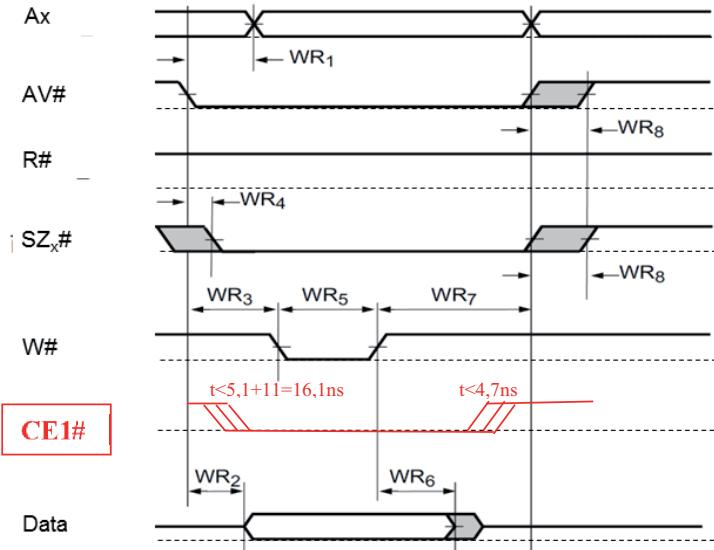


FIGURA D.51. CRONOGRAMA DE ACCESO.

$$t_{AV \text{ to } WR_{min}} = WR_{3min} + WR_{5min} = 12,7 + 7,7 = 20,4\text{ns}$$

Tiempos de selección

$$t_{AWmin} = WR_{3min} + WR_{5min} - WR_{1max} = 20,4 - 4,7 = 15,7\text{ns}$$

$$t_{CE1min} = WR_{3min} + WR_{5min} - (WR_{4max} + t_{LSmax}) = 20,4 - (5,1 + 11) = 4,3\text{ns}$$

$$t_{PWEmin} = 7,7\text{ns} / 6,2\text{ns} \text{ (depende del resto de tiempos)}$$

$$t_{SDmin} = WR_{3min} + WR_{5min} - WR_{2max} = 20,4 - 5,1 = 15,3\text{ns}$$

TABLA D.35. TIEMPOS DE ACCESO Y SELECCIÓN.

Tiempo	t_{ACC}	t_{Sel}	¿Se cumple? (Si / No)
t_{AW}	45ns	15,7ns	No
t_{SCS1}	45ns	4,3ns	No
t_{PWE}	40ns	7,7ns/6,2ns	No
t_{SD}	25ns	15,3ns	No

SOLUCIÓN AL PROBLEMA 4.31.

- La funcionalidad de la línea A0 es realizada por UDS# y LDS#, y la combinación de estas líneas indica cuando se accede a dirección par (A0=0) o dirección impar (A0=1). Además, el espacio de direccionamiento queda dividido en dos bancos: par e impar; siendo su tamaño $2^{23+1} \times 8 = 16\text{Mbytes}$.
- ROM: Programa (30K)+Datos permanentes (20K)+Tabla de vectores (1K)=51Kbytes®
Se implementan 64KBytes, en dos bancos de 32Kbytes cada uno.
RAM: Pila (20K)+Paso de parámetros (12K)=32Kbytes® Se implementan 32Kbytes, en dos bancos de 16Kbytes cada uno.

La Tabla D.36 muestra el mapa funcional.

TABLA D.36. MAPA FUNCIONAL.

000000 00FFFF	Tabla de vectores	ROM
	Programa	
	Datos permanentes	
FF0000		Vacío
	E/S	16K {FF0000-FF3FFF}
FFFFFF	Ampliación RAM	Reservado para ampliación RAM, 16K {FF4000-FF7FFF}
	Pila y paso de parámetros	RAM {FF8000-FFFFFF}

- La Tabla D.37 muestra la tabla de decodificación del sistema para generar las líneas de selección.

TABLA D.37. TABLA DE DECODIFICACIÓN.

A23	A22	A21	A20	A19	A18	Bloque
0	0	0	0	X	X	ROM
1	1	1	1	0	0	E/S
1	1	1	1	0	1	Ampliación RAM
1	1	1	1	1	X	RAM

Ecuaciones para implementación con puertas OR e inversores:

$$\overline{CS_{ROM}} = A20 + \overline{AS}$$

$$\overline{CS_{E/S}} = \overline{A20} + A19 + A18 + \overline{AS}$$

$$\overline{CS_{AMP_RAM}} = \overline{A20} + A19 + \overline{A18} + \overline{AS}$$

$$\overline{CS_{RAM}} = \overline{A20} + \overline{A19} + \overline{AS}$$

5. La Figura D.52 muestra la conexión completa.

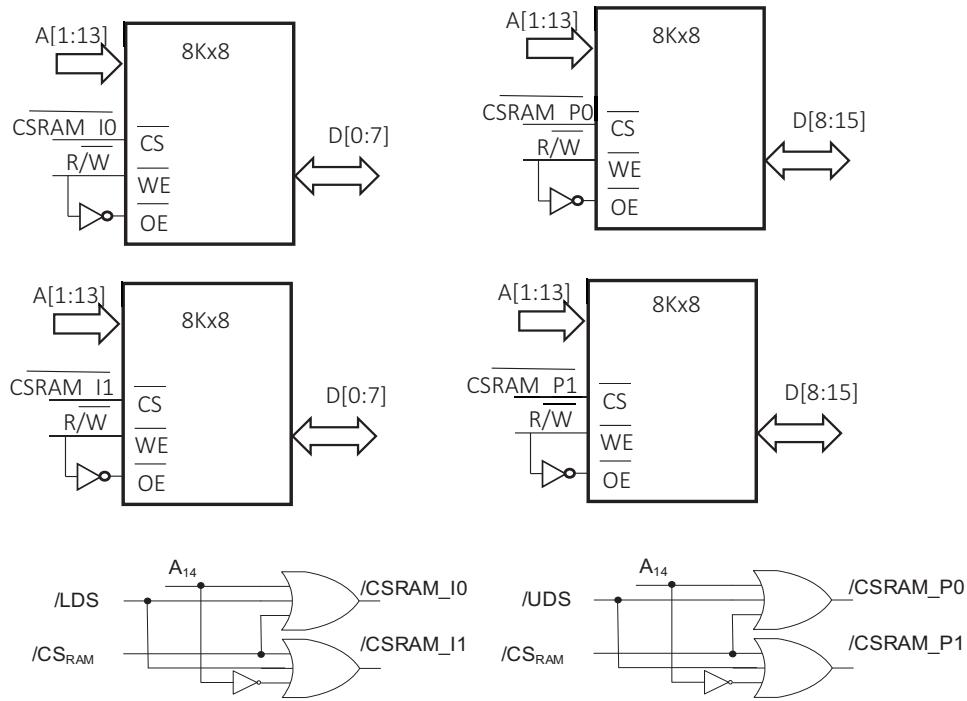


FIGURA D.52. CONEXIÓN DE LA MEMORIA RAM.

6. Tamaño de la memoria FLASH: $2^{22} \times 4\text{Mbytes}$ Upper; 4MBytes Lower, dos bancos de 4MBytes.

Para la ROM se necesitan 66KBytes, en dos bancos de 32KBytes.

Se deben conectar las líneas del microprocesador A[1:15] a las líneas A[0:14] de la memoria. El resto del bus de direcciones de la memoria, A[15:21], se fuerza a cero (por ejemplo, llevando esas líneas a masa). Las líneas UDS# y LDS# del microprocesador, se conectan respectivamente a las líneas UB# y LB# de la memoria.

7. La Figura D.53 muestra el cronograma con la solución.

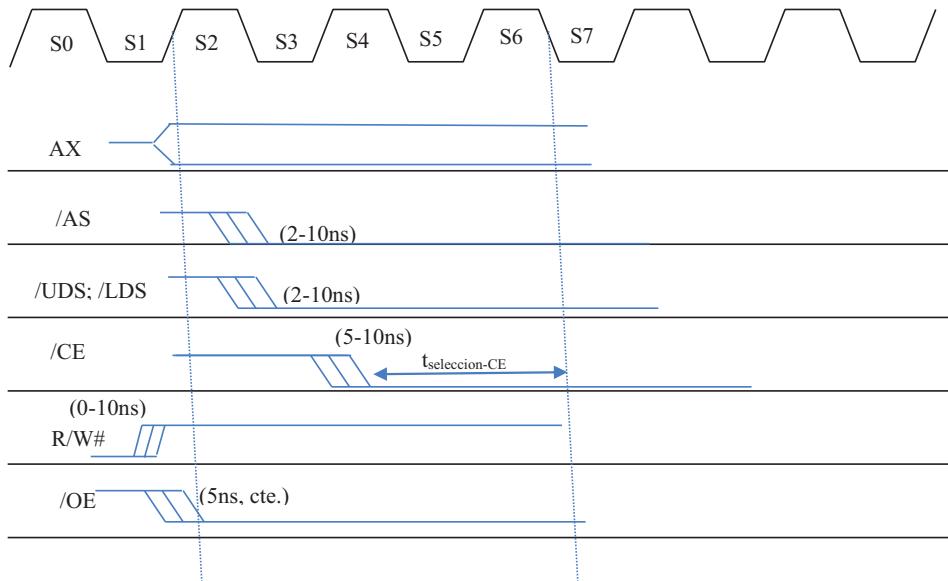


FIGURA D.53. CRONOGRAMA DE ACCESO.

Tiempo de acceso de la memoria: 70ns

Tiempo desde activación de /CE: 70ns

Tiempo desde activación de /OE: 30ns

$$T_{CLK} = 20\text{ ns}$$

Como /CE se activa después de /OE, hay que verificar que se cumpla la restricción desde la activación de /CE:

$$t_{seleccion-CE} = 5 \cdot \frac{T_{CLK}}{2} - t_{2mx} - t_{LSmx} - t_6 = 5 \cdot 10 - 10 - 10 - 3 = 27\text{ ns}$$

Como se requieren 70ns, habrá que prolongar el ciclo de bus en $70 - 27 = 43\text{ ns}$, que implica introducir **3 ciclos de espera** ($3 \cdot 20 = 60\text{ ns}$).

Este libro recoge una colección de ejercicios y casos prácticos que apoyan la adquisición de conceptos básicos del estudio teórico de Sistemas Electrónicos Digitales Programables. Recopila una serie de ejercicios organizados por capítulos que van avanzando desde conocimientos teóricos básicos de forma secuencial, progresiva y ordenada avanzando a conceptos más complejos para, finalmente, proponer sistemas de conjunto. Su contenido y ordenación son acordes a los temas teóricos impartidos en teoría de la asignatura de Sistemas Electrónicos Digitales impartida en los Grados TIC de la Escuela Politécnica Superior de la Universidad de Alcalá: introducción a los sistemas electrónicos digitales, arquitectura y modelo de programación de microcontroladores, sistema de excepciones y organización y gestión de memoria. El microcontrolador utilizado para el aprendizaje de conceptos es Cortex-M3 de ARM.

No obstante, esta publicación es válida para estudiantes de cualquier otro grado, o cualquier lector que quiera llevar a cabo la aplicación práctica de conceptos básicos adquiridos de Cortex-M3.

Esta publicación tiene como único objetivo facilitar a los estudiantes el desarrollo de ejercicios que refuerzen y reafirmen los conceptos teóricos estudiados. Se ha pretendido generar un documento claro y ordenado de los aspectos fundamentales de los sistemas electrónicos digitales.



Universidad
de Alcalá