Introducción a los computadores

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Colección de ejercicios

Juan Segarra y Alejandro Valero

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

Curso 2021–2022

Esta colección contiene una selección de ejercicios creados por los autores y también versiones de ejercicios recopilados de asignaturas extintas. Los ejercicios están ordenados conforme a la exposición de contenidos en las clases presenciales. Los ejercicios marcados con ⏱ son ejercicios de examen. Si usted considera que alguno de los ejercicios no debería aparecer en esta colección, póngase en contacto con los autores.

Índice general

Tema 1

Álgebra de Boole

A.1. Analizar cada uno de los circuitos combinacionales de la figura 1.1 sintetizando otro con menos puertas con la misma función. Para ello:

a) Obtener la expresión lógica que representa el circuito

b) Simplificar algebraicamente (sin hacer uso de la tabla de verdad) la expresión obtenida

c) Dibujar con puertas lógicas la expresión resultante

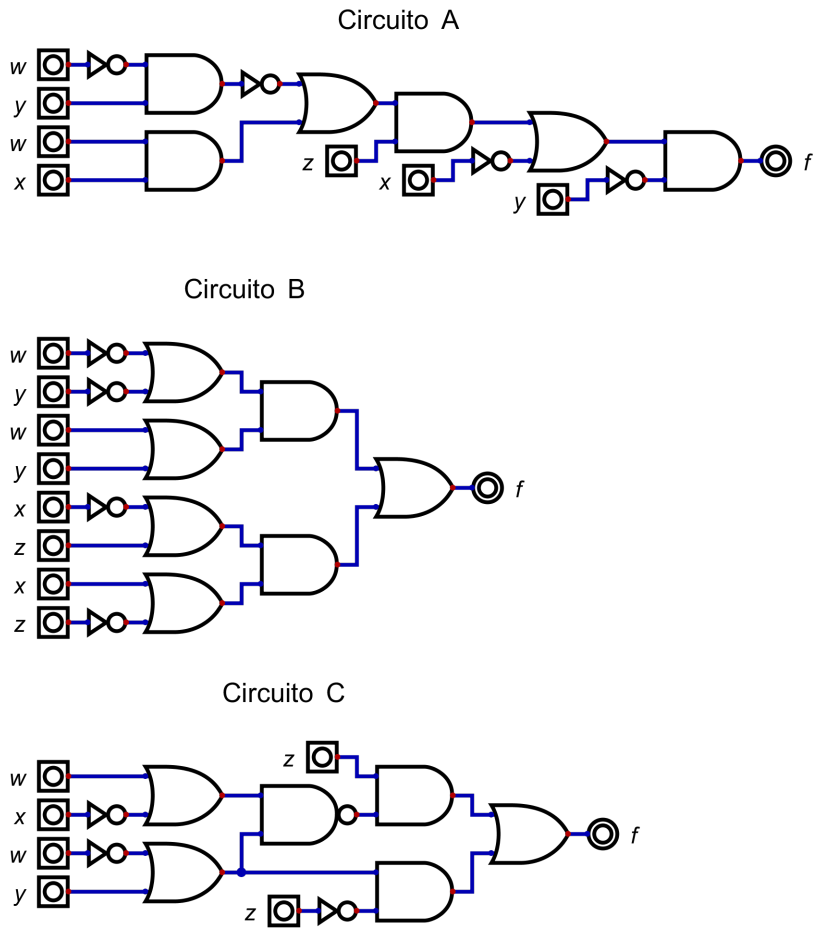


Figura 1.1: Circuitos con varios niveles de puertas lógicas.

A.2. Comprueba si las siguientes expresiones booleanas son equivalentes comparando sus tablas de

verdad. Si lo son, a partir de la tabla de verdad obtén la forma canónica más corta (1ª FC o 2ª FC).

a) ¿$(\overline{A} \cdot \overline{B} + A \cdot \overline{C}) \cdot (A \cdot B + \overline{A} \cdot \overline{B}) \equiv \overline{A} \cdot \overline{B} + A \cdot B \cdot \overline{C}$?

b) ¿$(A \cdot B \cdot \overline{D} + C) \cdot (C + \overline{D}) \equiv A \cdot B \cdot \overline{D} + C$?

c) ¿$\overline{A} \cdot B + \overline{A} \cdot B \cdot C + B \cdot C + B \cdot \overline{C} \equiv B$?

d) ¿$A \cdot B + \overline{A} \cdot C + \overline{B} \cdot C \equiv A \cdot B + C$?

e) ¿$(A + B + \overline{C}) \cdot (\overline{A} + \overline{C}) \cdot (\overline{B} + C) \equiv \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C$?

A.3. Simplifica las siguientes expresiones booleanas aplicando las propiedades vistas en clase:

a) $\overline{(A + B) \cdot (\overline{A} + C) + \big((\overline{A} + \overline{B} + \overline{A} \cdot B \cdot C) \cdot (A + \overline{A} \cdot B) \cdot (\overline{A} + \overline{B})\big)}$

b) $(A \cdot B + C + D) \cdot (C + \overline{D}) \cdot (C + \overline{D} + E)$

c) $\overline{A} + C \cdot D + \overline{\overline{\overline{A} \cdot D} \cdot \overline{\overline{A} \cdot B \cdot C}} + \overline{A} \cdot \overline{D} + C \cdot \overline{D}$

Tema 2

Representación numérica

N.1. Realiza las siguientes operaciones *complemento a 1* con 4 bits, indicando si hay desbordamiento e interpretando el resultado final.

a) $7 - 0$
b) $8 - 1$
c) $-1 + (-8)$
d) $-7 - 1$
e) $-5 + (-1)$
f) $-1 + (-2)$

N.2. Realiza las siguientes operaciones *complemento a 2* con 4 bits, indicando si hay desbordamiento e interpretando el resultado final.

a) $7 - 0$
b) $8 - 1$
c) $-1 + (-8)$
d) $-7 - 1$
e) $-5 + (-1)$
f) $-1 + (-2)$

N.3. ¿Cuál es la representación decimal de los siguientes valores binarios? Represéntalos en 8 bits codificados en coma fija, con 4 bits para la parte entera.

a) 11,01
b) 110,101
c) 1011,011

N.4. ¿Cuál es la representación binaria en coma fija de 16 bits, con 12 bits para la parte entera, de los siguientes números decimales? Indica también cuál es el error de redondeo cometido con cada uno de ellos.

a) 27,325
b) 271,5625
c) 1239,66

N.5. Interpreta los siguientes vectores de bits según el estándar IEEE-754 de 32 bits

a)

0	1000 0000	0000 0000 0000 0000 0000 000

b)

1	1000 0001	1111 0000 0000 0000 0000 000

c)

0	0111 1110	0000 0000 0000 0000 0000 000

N.6. Representa los siguientes números según el estándar IEEE-754 de 32 bits e indica cuál es el error de redondeo cometido en cada uno de ellos.

a) 27,625

b) $-271{,}5625$

c) 28,15

Tema 3

Circuitos combinacionales

3.1. Circuitos combinacionales con puertas lógicas

P.1. El incrementador-decrementador de la figura 3.1 tiene dos entradas A y C y una salida F. La entrada A es un número natural de tres bits (a_2, a_1, a_0) y la entrada C es una señal de control de un bit. La salida F también tiene tres bits (f_2, f_1, f_0). Cuando $C = 1$, el sistema incrementa el número A, y cuando $C = 0$ lo decrementa. En ambos casos, la salida es la aplicación de la función «módulo 8» sobre el número A modificado ($F = (A \pm 1) \bmod 8$). La función *módulo* se define como el menor resto positivo resultante de la división entera (e.g. $3 \bmod 8 = 3$, $15 \bmod 8 = 7$, $-3 \bmod 8 = 5$). Realiza la tabla de verdad y obtén las expresiones que definen a los tres bits de F.

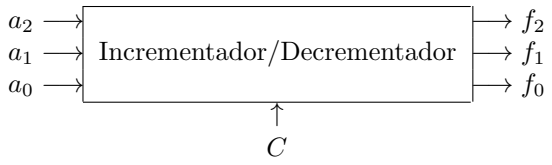


Figura 3.1: Incrementador/decrementador módulo 8.

P.2. ⏱ [Segarra 2019, UZ] Dado un número entero A representado con 4 bits en complemento a 2, se desea implementar un circuito con las siguientes salidas. Una salida P, que deberá ponerse a 1 cuando A sea positivo (el 0 *no* se considera positivo). Una salida Z, que deberá ponerse a 1 cuando A sea cero. Una salida M, que deberá ponerse a 1 cuando el *valor absoluto* de A sea mayor o igual que 3.

a) Obtén la tabla de verdad del circuito.

b) Obtén una expresión de Z.

c) Obtén la expresión correspondiente a la $2^{\underline{a}}$ *forma canónica* de P.

d) Obtén una expresión minimizada de M.

P.3. El circuito de la figura 3.2 tiene como entrada un número A de 4 bits (a_3, a_2, a_1, a_0) codificado en BCD, y genera tres funciones aritméticas f_2, f_1, f_0. *Binary-Coded Decimal* es un código de cuatro bits para los números naturales entre el 0 y el 9, codificados como binario natural. Proponed una síntesis mínima de las tres funciones f_i.

$$f_2 = \begin{cases} 1 & \text{si } A \bmod 4 = 0 \\ 0 & \text{en caso contrario} \end{cases} \qquad f_1 = \begin{cases} 1 & \text{si } 1 \leq A \leq 4 \\ 0 & \text{en caso contrario} \end{cases} \qquad f_0 = \begin{cases} 1 & \text{si } A \bmod 2 = 0 \\ 0 & \text{en caso contrario} \end{cases}$$

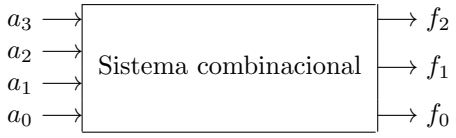


Figura 3.2: Detector de propiedades de un número codificado en BCD.

P.4. Un proceso químico posee tres indicadores para controlar su temperatura (figura 3.3). Cada indicador tiene una constante de referencia $T_1 < T_2 < T_3$. Para cada indicador, su salida adopta dos niveles de tensión eléctrica dependiendo de si la temperatura detectada es menor o igual, o mayor que su constante de referencia. Es decir, $t_1 = 0$ cuando la temperatura es menor o igual que T_1 y $t_1 = 1$ cuando es mayor que T_1. Lo mismo ocurre con las salidas t_2 y t_3 en relación con las constantes T_2 y T_3. Se desea sintetizar un circuito combinacional que genere una salida $S = 1$ si la temperatura del proceso químico está comprendida entre T_1 y T_2, o bien si es superior a T_3, y $S = 0$ en caso contrario. Se desea una implementación con solo puertas NAND.

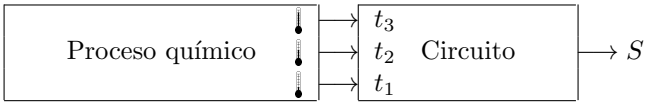


Figura 3.3: Diagrama para un control de temperaturas.

P.5. Se desea diseñar un sistema para controlar el motor eléctrico de una estación de bombeo. Las entradas de este sistema son las señales lógicas NM, NS, IN y CM. La señal NM (salida del sensor de *Nivel Máximo* del depósito) vale 1 si el agua supera el nivel máximo y 0 en caso contrario. La señal NS (salida del sensor de *Nivel de Seguridad* del depósito, situado por debajo del nivel máximo) vale 1 si el agua supera el nivel de seguridad y 0 en caso contrario. La señal IN (*Indicador de Noche*, generada a partir de un reloj) vale 1 durante la noche y 0 durante el día. La señal CM (*Control Manual*, controlada por un interruptor) vale 1 cuanto el sistema está en modo manual y 0 cuando está en modo automático. El motor de la bomba se activa cuando la salida B toma el valor 1 y se para en caso contrario.

El sistema debe cumplir las siguientes especificaciones:

- La bomba debe funcionar durante la noche si el depósito está por debajo del nivel máximo.
- La bomba debe funcionar de día cuando el agua esté por debajo del nivel de seguridad.
- El control manual pondrá en marcha la bomba si el depósito está por debajo del nivel máximo.

Se pide:

a) Obtener la tabla de verdad de la función B.

b) Minimizar la función y realizar una síntesis en suma de productos.

c) Idem, pero como producto de sumas.

d) Realizar el circuito sólo con puertas NOR.

e) Realizar el circuito sólo con puertas NAND.

3.2. Circuitos combinacionales con celdas encadenables

C.1. Realizar un circuito encadenable para comparar números naturales que examine en primer lugar los bits más significativos (de mayor peso). Hay que generar las relaciones *mayor*, *menor* e *igual*.

C.2. Dado un vector de 2^n bits, diseñar un bloque combinacional con n salidas que indiquen la posición (el índice) del «primer cero a la izquierda». Por ejemplo, para $n = 3$ y el vector 00100110, la salida de tres bits debe ser 6. Debe diseñarse una salida adicional que indique si tal cero no existe. Realizad este módulo mediante un circuito encadenable y un codificador.

3.3. Circuitos combinacionales con bloques

B.1. El decodificador decimal de la figura 3.4 corresponde al circuito integrado de mediana escala de integración (MSI) 7442 de 16 *pines*. Observa que tiene sus salidas negadas. Si la entrada X no pertenece al código BCD ($X > 9$) todas las salidas z_i se ponen a 1. Propón una implementación con 4 inversores (puertas NOT) y 10 puertas NAND de 4 entradas.

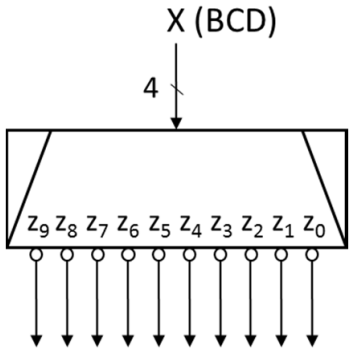


Figura 3.4: Decodificador BCD 7442.

B.2. Analizar el circuito de la figura 3.5. Obtener una expresión aritmético-lógica de la salida C en función de las entradas A y B, considerando el siguiente funcionamiento para los bloques M:

$$P = \begin{cases} 1 & \text{si } R > Q \\ 0 & \text{si } R \leq Q \end{cases} \qquad U = \begin{cases} W & \text{si } S = 1 \\ V & \text{si } S = 0 \end{cases} \qquad Z = X - Y$$

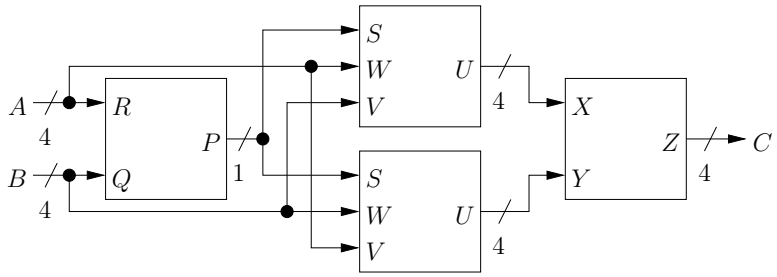


Figura 3.5: Análisis de bloques aritméticos y lógicos.

B.3. El código *Exceso*-3 consiste en representar el valor deseado desplazado en tres unidades con respecto al original. Por ejemplo, el número 4 se representaría como 0111 ($4 + 3 = 7_{10} = 0111_2$). En caso de querer *interpretar* un número codificado en exceso 3 habría que realizar la operación inversa: la representación 0010 correspondería al número -1 ($0010_2 = 2_{10}$; $2 - 3 = -1$).

a) Diseñar un conversor de código BCD Exceso-3 a código BCD utilizando un sumador de 4 bits.

b) Diseñar un conversor de código BCD a código BCD Exceso-3 mediante el decodificador del ejercicio B.1 y 4 puertas NAND.

B.4. Diseñar una Unidad Aritmética de dos operandos A, B utilizando un sumador de dos números de cuatro bits y multiplexores 4:1. Dependiendo de los bits de control t_1, t_0, la ALU debe ser capaz de generar como resultado R (4 bits) todo ceros, el operando A, el operando B, o la suma de ambos ($(A + B)$ mód 16) y su acarreo de salida C_{out}, tal y como se indica en la tabla 3.1.

B.5. Existen sistemas de representación de números que utilizan aritmética modular. Vamos a considerar dos sistemas para representar números naturales entre el 0 y el 15. El sistema A representa el natural

t_1	t_0	R	C
0	0	0	0
0	1	A	0
1	0	B	0
1	1	S	C_{out}

Tabla 3.1: Especificación de la ALU.

n como $p = 3n$ mód 16 y p se codifica en binario. El sistema B representa n como $q = 7n$ mód 16 y también q se codifica en binario.

Ejemplo sistema A: $n = 3 \rightarrow p = 9$ mód $16 = 9_{10} = 1001_2$

Ejemplo sistema B: $n = 6 \rightarrow q = 42$ mód $16 = 10_{10} = 1010_2$

Para operar con ambos sistemas queremos un circuito combinacional de interfaz que convierta la representación A a la representación B.

a) Diseñar el conversor utilizando multiplexores de 4:1 para q_3 y puertas para q_2.

b) Diseñar el conversor utilizando un decodificador de 4:16 y un codificador de 16:4.

B.6. Diseñar un multiplicador de números naturales. El multiplicando A tiene 4 bits A_3, A_2, A_1, A_0 y el multiplicador B tiene 3 bits B_2, B_1, B_0. El producto C tiene 7 bits $C_6, C_5, C_4, C_3, C_2, C_1, C_0$. La figura 3.6 muestra visualmente la multiplicación deseada. Puede hacerse mediante 12 puertas y dos sumadores de 4 bits.

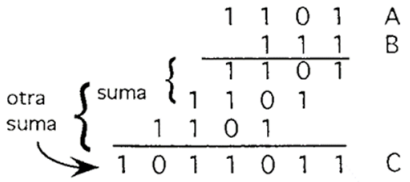


Figura 3.6: Multiplicador de dos números naturales.

B.7. El circuito de la figura 3.7 sintetiza cierta función F con demasiados componentes.

a) Encontrar una expresión de F.

b) Sintetizar la misma función con un decodificador 2:4 y el mínimo número de puertas adicionales.

3.4. Circuitos combinacionales con ROMs y PLAs

R.1. Un circuito debe realizar la conversión de un número natural codificado en binario de 6 bits a un número equivalente codificado con dos dígitos BCD. Por ejemplo, el número 101101 se traduce a la cadena BCD 0100 0101. Diseñar el circuito usando una ROM de 32×6 (32 palabras de 6 bits), indicando el contenido de la memoria y las conexiones adicionales.

R.2. Utilizar un PLA para convertir de código BCD al código progresivo indicado en la tabla 3.2.

R.3. Diseñar un circuito aritmético que produce el cuadrado de un número natural de 3 bits codificado en binario natural.

a) Mediante PLA y con el mínimo número de términos producto.

b) Mediante ROM.

c) Comparar los diseños en velocidad y complejidad.

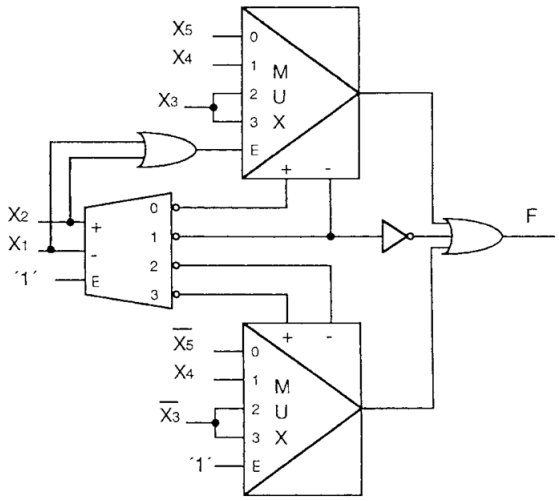


Figura 3.7: Circuito confuso.

Número	BCD				Progresivo			
	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	0	0	0

Tabla 3.2: Tabla de conversión de BCD a progresivo.

R.4. Diseñar un circuito aritmético que produce una salida Z, de 10 bits, con el cuadrado de un número natural X de 5 bits codificado en binario mediante una ROM de 32×8 bits y las conexiones necesarias. Dibujar el diagrama del circuito y listar las 10 primeras y 10 últimas posiciones de la memoria.

Tema 4

Circuitos secuenciales

4.1. Análisis de circuitos secuenciales

S.1. Analizar el circuito lógico de la figura 4.1 asumiendo como salida un número natural de 3 bits (Q_3, Q_2, Q_1).

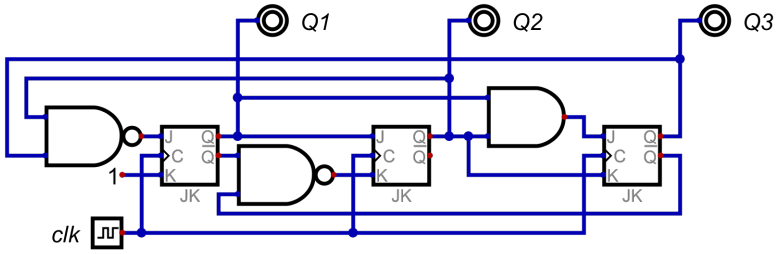


Figura 4.1: Circuito secuencial con flip-flops JK.

S.2. Estudiar el comportamiento del flip-flop AB que muestra la figura 4.2.

a) Determinar la tabla de transición y la ecuación $Q^+ = f(Q, A, B)$

b) Determinar la tabla de excitación

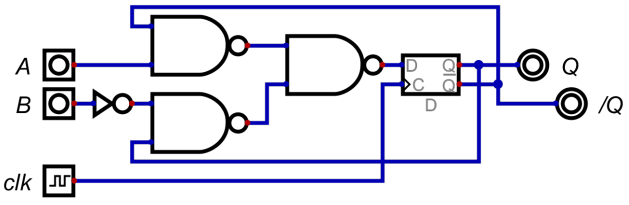


Figura 4.2: Flip-flop AB.

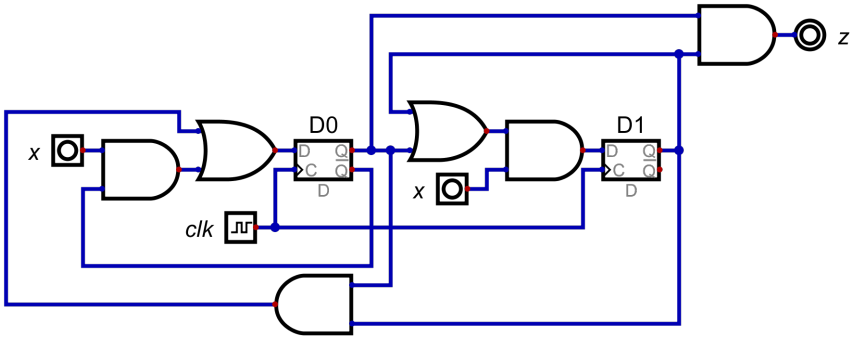
S.3. ⏱ [Viñals 2019, UZ] Dado el siguiente circuito con una entrada x y una salida z:

a) ¿El autómata que lo describe será de tipo Moore o de Mealy? ¿Por qué?

b) Obtén las tablas de transición y salida.

c) Dibuja el diagrama de estados.

d) Calcula el tiempo de ciclo mínimo y la frecuencia de operación máxima asumiendo flip-flops con retardo $t_d = 400$ ps, *set-up* $t_{su} = 150$ ps y puertas con retardo 50 ps.



S.4. Dado el sistema secuencial de la figura 4.3 con un valor inicial $Z = 0$.

a) Obtén el diagrama de estados para $n = 2$.

b) Explica qué hace el circuito

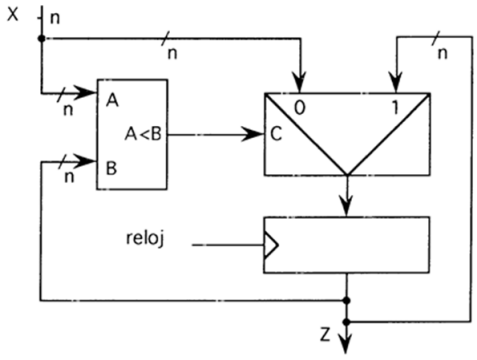


Figura 4.3: Circuito con bloques combinacionales y registro.

S.5. Dibujar un cronograma de las señales A, B, C, D, E, F y G del circuito de la figura 4.4. Asumir Q_i, con i de menor a mayor peso, de izquierda a derecha.

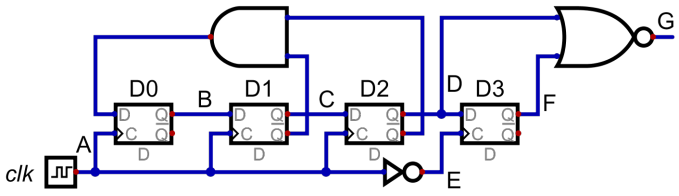


Figura 4.4: Circuito secuencial con flip-flops D.

4.2. Diseño de circuitos secuenciales

D.1. Se desea diseñar un reconocedor de secuencia que tiene una salida $Z = 1$ cuando en su entrada X se han suministrado exactamente tres 1 durante los tres ciclos *anteriores* de reloj. Si durante cuatro o más ciclos hay 1s consecutivos, $Z = 0$.

a) Realizarlo mediante el menor número posible de biestables D y puertas.

b) Realizarlo mediante un registro de desplazamiento de longitud conveniente y las puertas necesarias.

c) Comparar ambas soluciones teniendo en cuenta: facilidad de diseño, facilidad de modificación y coste.

D.2. Especificar mediante un diagrama de estados el reconocimiento de los códigos de longitud variable A, B, C, D, E tal y como se indica en la figura 4.5. Es decir, la entrada del reconocedor es de un bit, y el código va entrando en serie, bit a bit. La salida tiene el número de bits necesario para codificar los cinco símbolos de la tabla, y uno más, el símbolo r, que se da en los ciclos dónde el reconocimiento está en progreso.

X	Código
0	A
10	B
110	C
1110	D
1111	E

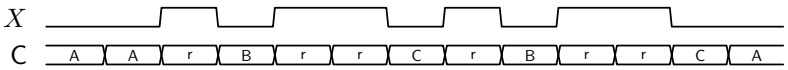


Figura 4.5: Tabla de códigos y ejemplo de reconocimiento de secuencia.

D.3. Tras una observación del comportamiento de un perro que se presentó voluntario, se obtuvieron las siguientes conclusiones. Pluto, que así vamos a llamarlo para preservar su intimidad, podía estar en uno de los siguientes cuatro estados de ánimo: *tranquilo*, *irritado*, *asustado* o *asustado e irritado* en cuyo caso muerde. Se ha observado que si le damos un hueso se queda tranquilo, pero si se lo quitamos se irrita. Por otra parte si lo amenazamos se asusta. Del cuaderno de campo leemos que solo se realizó una acción a la vez para cambiar el estado de ánimo de Pluto. Queremos modelar el comportamiento de Pluto con un autómata de Moore. Vamos a considerar tres entradas binarias: DH (Dar Hueso), A (Asustar) y QH (Quitar Hueso). Diseñad el autómata, identificando estados y salidas.

D.4. Sea un circuito con una entrada X y una salida Z, ambas de 2 bits. X codifica un número natural. Si el valor de X en el ciclo actual es mayor que el valor en el ciclo anterior, entonces la salida en el ciclo actual será $Z = 10_2$. Si dicho valor es menor, entonces $Z = 01_2$. En cualquier otro caso, $Z = 00_2$. Obtener el diagrama de estados del autómata correspondiente.

D.5. Se desea diseñar un circuito secuencial con una entrada x de 1 bit y una salida y de 1 bit tal que la salida actual sea igual a la entrada de hace dos ciclos. Hay que hacerlo con flip-flops tipo D.

D.6. Un reconocedor de secuencias con solapamiento tiene una entrada $X \in a, b, c, l$ que codifica cuatro símbolos posibles y una salida Z binaria. $Z = 1$ cuando los símbolos de entrada en los cuatro ciclos anteriores corresponden a la secuencia *alba*. $Z = 0$ en cualquier otro caso.

a) Determinar la secuencia de salidas para la entrada: b a c a l a l b a l b a b l a

b) Diagrama de estados del sistema secuencial

c) Repetir el ejercicio pero sin solapamiento

D.7. Se quiere cifrar un mensaje binario (secuencia de ceros y unos) mediante la función:

$$Z = \begin{cases} X & \text{si } X^- = 0 \\ \overline{X} & \text{si } X^- = 1 \end{cases}$$

Para ello se desea diseñar un circuito que se comporte como en el ejemplo de la figura 4.6.

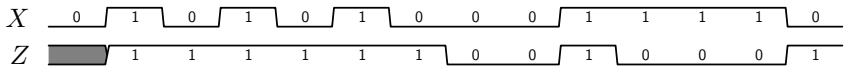


Figura 4.6: Ejemplo de funcionamiento de un circuito para cifrar.

a) Realizar el diagrama de estados del autómata apropiado. Sintetizar el circuito secuencial con el mínimo número de flip-flops D y puertas

b) Realizar el diagrama de estados correspondiente al receptor, que devuelve la secuencia a su estado original

c) Realizar el diagrama de estados para la siguiente función de cifrado:

$$Z^+ = \begin{cases} X & \text{si } X^{2-} = 0 \\ \overline{X} & \text{si } X^{2-} = 1 \end{cases}$$

D.8. Se desea diseñar el sistema de control de un coche de juguete. El coche puede estar parado ($M = 00$) o avanzar. Si avanza, puede ir en línea recta ($M = 11$), avanzar girando hacia la izquierda ($M = 10$) o avanzar girando hacia la derecha ($M = 01$). Para controlar el coche se dispone de un mando a distancia con dos botones independientes. El botón *izquierda* puede estar pulsado ($I = 1$) o sin pulsar ($I = 0$), y el botón *derecha* puede estar pulsado ($D = 1$) o sin pulsar ($D = 0$). El controlador debe proporcionar el siguiente comportamiento:

- Cuando se conecta el juguete, el coche debe empezar parado
- Al pulsar los dos botones a la vez, el coche debe avanzar recto si estaba parado, y parar si estaba en movimiento
- Al pulsar *izquierda*, el coche debe ir a la izquierda si estaba yendo recto o ya estaba girando a la izquierda, y debe ir recto si estaba girando a la derecha
- Al pulsar *derecha*, el coche debe ir a la derecha si estaba yendo recto o ya estaba girando a la derecha, y debe ir recto si estaba girando a la izquierda

a) Realizad este sistema secuencial mediante un autómata de Moore con flip-flops tipo D

b) ¿Tiene alguna influencia el periodo de reloj en el comportamiento del coche?

c) ¿Cuál sería una frecuencia de reloj razonable?

D.9. ⏱ [Segarra 2004b, UZ] 1º de «Defensa de las artes mágicas» en Hogwarts. Tenemos 3 posibles posiciones de la varita mágica: *adelante* (00), *intermedia* (01) y *atrás* (11). Para cambiar de posición tenemos un movimiento o de *avanzar varita* o de *retroceder varita*, de forma que, por ejemplo, estando en posición *adelante* y haciendo un movimiento de *retroceder varita* pasamos a posición *intermedia*, y estando en posición *adelante* el movimiento *avanzar varita* es imposible. Además sabemos que no podemos quedarnos quietos sin hacer movimientos ni tampoco hacer ambos movimientos (*avanzar* y *retroceder*) al mismo tiempo. Finalmente, cada movimiento podemos hacerlo de dos formas: o *rápido* o *lento*. Usando el método anterior de mover la varita mágica podemos hacer varios hechizos (identificados con un código de salida).

Expelliarmus (001): Desde posición *intermedia* hasta posición *adelante* con movimiento *rápido*.

Petrificus totalis (010): Desde posición *intermedia* hasta *adelante* con movimiento *lento*.

Wingardium leviosa (011): Desde posición *adelante* hasta *intermedia* con movimiento *lento*.

Accio (100): Desde posición *intermedia* hasta posición *atrás* con movimiento *rápido*.

Alohomora (101): Desde posición *intermedia* hasta posición *atrás* con movimiento *lento*.

Expecto patronum (110): Desde posición *atrás* hasta posición *intermedia* con movimiento *rápido* (sin efectos) y seguidamente desde posición *intermedia* hasta posición *adelante* con movimiento *rápido* (ejecución del hechizo).

Sin efectos (000): Cualquiera de los movimientos restantes.

Nota: Como la parte final del hechizo *expecto patronum* coincide con el *expelliarmus*, si los movimientos nos permiten hacer un *patronum* el hechizo resultante será un *patronum*, mientras que en caso contrario será un *expelliarmus*.

Nota 2: Se ha de tener en cuenta que los hechizos se ejecutan en el instante en el que se dan los movimientos requeridos.

a) Obtener el diagrama de estados

b) Obtener la función de salida del bit de menor peso (minimizada)

Tema 5

La Máquina Sencilla

5.1. Decodificación y ejecución

E.1. ⏱ [Segarra 2004a, UZ] Responde de forma breve a las siguientes preguntas:

a) ¿Dónde apunta el contador de programa (PC)?

b) ¿Para qué se usa el *flag* de cero en la máquina sencilla?

c) Si los multiplexores siempre tienen una única salida, ¿cómo es posible que el que hay en el esquema de la máquina sencilla tenga 7?

d) Cuando la línea de lectura/escritura de la memoria está puesta a escritura, qué hay en su línea de direcciones? ¿Y si dicha línea está puesta a lectura?

e) ¿Por qué al cargar una instrucción almacenamos la información de los operandos en el registro de instrucción (IR) en vez de cargar directamente los datos en los registros de la ALU?

f) Si el PC sólo puede ser actualizado sumando 1, ¿cómo es posible que en los saltos el PC pueda actualizarse a cualquier dirección de destino?

E.2. ⏱ [Ayuso, Ramos y Segarra 2004, UZ] Dado el contenido para la memoria de la Máquina Sencilla de la tabla 5.1 y suponiendo que el programa empieza a ejecutarse en la dirección 0:

a) Decodifica el programa indicando instrucción, operando fuente y operando destino.

b) Ejecuta 5 instrucciones e indica el contenido final de la memoria

c) Si seguimos ejecutando indefinidamente, ¿cuántas veces salta la instrucción de la dirección 3? ¿Y la instrucción de la dirección 5?

Dir.	Contenido previo	Ensamblador / Contenido tras ejecutar 5 inst.
@0	0xA042	
@1	0x20C2	
@2	0x6140	
@3	0xC001	
@4	0x6040	
@5	0xC005	
...	...	...
@64	0x0	
@65	0x1	
@66	0x2	

Tabla 5.1: Contenido de memoria para la Máquina Sencilla, en hexadecimal, para decodificar y ejecutar.

E.3. ⏱ [Ayuso, Ramos y Segarra 2011, UZ] Disponemos de la Máquina Sencilla, con una unidad de control cableada optimizada, con el contenido de memoria de la tabla 5.2.

a) Obtener el programa ensamblador codificado en las posiciones de memoria 0 a 9

b) Obtener el contenido de las posiciones de memoria modificadas tras la ejecución de 10 instrucciones, suponiendo que el valor inicial de PC es cero

c) ¿Cuántas instrucciones de cada tipo se han ejecutado?

d) ¿Cuántos ciclos ha costado ejecutar esas 10 instrucciones?

Dir.	Contenido previo	Ensamblador / Contenido tras ejecutar 10 inst.
@0	10 0001010 0001101	
@1	00 0001101 0001101	
@2	00 0001011 0001101	
@3	00 0001100 0001101	
@4	01 0001100 0001100	
@5	11 0001100 0000101	
@6	00 0001010 0001011	
@7	00 0000000 0000001	
@8	00 0000000 0000000	
@9	00 0000000 0000000	
@10	0000 0000 0000 0111	
@11	0000 0000 0000 0101	
@12	0000 0000 0000 0001	
@13	0000 0000 0000 0000	

Tabla 5.2: Contenido de memoria para la Máquina Sencilla, en binario, para decodificar y ejecutar.

E.4. ⏱ [Ayuso, Ramos y Segarra 2012, UZ] Dado el contenido en la memoria de una Máquina Sencilla original de la tabla 5.3:

a) Obtener las instrucciones codificadas en las direcciones de memoria 0 a 9

b) Asumiendo que el valor inicial del registro PC es 0 y se ejecutan 11 instrucciones, obtener el contenido final de memoria

c) ¿Cual es el valor final del registro PC?

d) ¿Cuántas instrucciones de cada tipo se han ejecutado?

e) Suponiendo que esta máquina sencilla utiliza la unidad de control cableada optimizada, ¿cuántos ciclos ha costado ejecutar las 11 instrucciones?

5.2. Modificación de instrucciones y de la unidad de control

I.1. ⏱ [Ayuso, Ramos y Segarra 2005, UZ] Se desea sustituir la instrucción CMP de la Máquina Sencilla por la instrucción CMP2. Dicha instrucción debe comparar el contenido de la dirección de memoria F, el contenido de la dirección de memoria D, y el operando inmediato K, y guardar en FZ si los tres valores son iguales o no. El operando K viene dado en la propia instrucción (direccionamiento inmediato), concretamente en la dirección de memoria sucesiva respecto a la dirección donde se encuentra el código de operación y las direcciones de los operandos F y D. Para dar soporte al nuevo repertorio de instrucciones no se ha modificado la unidad de proceso (UP) pero es necesario modificar la UC.

a) Asumiendo la Máquina Sencilla original, haz el grafo de estados (incluyendo las fases de búsqueda y decodificación) de la instrucción CMP2 y dibuja la tabla de vectores de salida de cada estado para la instrucción CMP2.

b) Asumiendo la Máquina Sencilla microprogramada, obtén el microprograma para el nuevo repertorio de instrucciones. Si la frecuencia de reloj del procesador es de 1 MHz, indica el tiempo necesario para ejecutar cada una de las cuatro instrucciones de la nueva Máquina Sencilla.

Dir.	Contenido previo	Ensamblador / Contenido tras ejecutar 11 inst.
@0	10 0001101 0001111	
@1	00 0001010 0001111	
@2	00 0001110 0010000	
@3	11 0000000 0000110	
@4	01 0001101 0001101	
@5	11 0000000 0000001	
@6	00 0001011 0001111	
@7	00 0001100 0001111	
@8	00 0000000 0000000	
@9	00 0000000 0000000	
@10	0000 0000 0000 0100	
@11	0000 0000 0000 0011	
@12	0000 0000 0000 0000	
@13	0000 0000 0000 0000	
@14	0000 0000 0000 0001	
@15	0000 0000 0000 0010	
@16	1111 1111 1111 1110	

Tabla 5.3: Contenido de memoria para la Máquina Sencilla, en binario, para decodificar y ejecutar.

I.2. ⏱ [Segarra 2006, UZ] Optimiza el microprograma de la Máquina Sencilla rellenando los huecos correspondientes al contenido de la memoria del microprograma en la tabla 5.4. Se debe conseguir un microprograma correcto que sea capaz de hacer:

- ADD en 4 ciclos.
- MOV en 4 ciclos.
- CMP en 4 ciclos, ahorrando 1 ciclo a la siguiente instrucción.
- BEQ en 4 ciclos si no se salta y 5 si se salta. Si se salta, ahorrando 1 ciclo a la siguiente instrucción.

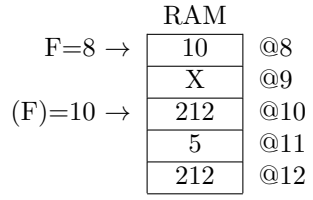
Pista: Ha de tenerse en cuenta que en las decodificaciones la UP puede adelantar trabajo buscando posibles operandos y que si la fase de ejecución no accede a memoria podemos aprovechar para buscar la siguiente instrucción.

@	Salto	TEST	MX ALU $\overline{\text{L}}$/E PC IR A B FZ	
0				fetch
1 2				si $CO_0 = 1\ldots$ si $CO_1 = 1\ldots$
3				ADD…
4				MOV…
5				si $CO_1 = 1\ldots$
6				CMP…
7 8				BEQ…

Tabla 5.4: Esquema para optimizar el microprograma de la Máquina Sencilla.

I.3. ⏱ [Ayuso, Ramos y Segarra 2006, UZ] Se desea modificar la Máquina Sencilla básica con un nuevo modo de direccionamiento: *modo postdecremento*. Para ello, la ALU se modifica de forma que el código de operación 11 (sin utilizar en la máquina original), decrementa en 1 el dato proveniente del registro de entrada B. Este modo sólo se aplicará al campo fuente de la instrucción CMP (ahora *CPD*) que es la única instrucción de la que nos vamos a ocupar:

CPD F-,D ; si ((F) mod 2^7)=(D) entonces Z←1
; si no Z←0
; (F)←(F)-1



CPD 8-, 11 compara el contenido de @10 y @11, es decir, 212 y 5
⇒ Z←0; @8 ← 10 − 1 = 9

CPD 8-, 12 compara el contenido de @10 y @12, es decir, 212 y 212
⇒ Z←1; @8 ← 10 − 1 = 9

Figura 5.1: Ejemplo de uso de CMP.

La figura 5.1 muestra un ejemplo del comportamiento de CPD.

a) Modifica la Unidad de Proceso (UP) de la Máquina Sencilla original para dar soporte a la nueva instrucción variando o añadiendo conexiones, pero sin añadir a la arquitectura ningún bloque. Pista: para poder acceder al operando fuente es preciso poder direccionar con el dato que se encontraba en la dirección F de la instrucción, el cual debe haberse traído de la memoria a la CPU previamente.

b) Obtén el diagrama de estados de la Unidad de Control cableada correspondiente a CPD. Indica explícitamente los movimientos entre registros que se efectúan en cada estado (también denominados RTL).

c) Partiendo del contenido de la memoria de la tabla 5.5 rellena las direcciones @0 a @5 con el programa ensamblador correspondiente y las direcciones @64 a @67 con los valores resultado de ejecutar 6 instrucciones. Nota: PC = 0.

d) ¿Cuál es el valor final de PC?

Dir.	Contenido previo	Ensamblador / Contenido tras ejecutar 6 inst.
@0	1010000001000010	
@1	0010000011000010	
@2	1010000101000000	
@3	1100000000000001	
@4	0110000111000000	
@5	1100000000000100	
...	...	...
@64	0000000000000000	
@65	0000000000000001	
@66	0000000000000010	
@67	0000000001000001	

Tabla 5.5: Contenido de memoria para la Máquina Sencilla, en binario, para decodificar y ejecutar.

I.4. ⏱ [Ayuso, Ramos y Segarra 2007, UZ] En la Máquina Sencilla cableada se desea *sustituir* la instrucción BEQ por la instrucción IBEQ (salto condicional *indirecto*), donde la dirección de destino de salto viene indicada en los 7 bits menos significativos del contenido de memoria de la dirección D.

BEQ D ; si $FZ = 1$: $PC \leftarrow D + 1$, $IR \leftarrow (D)$
IBEQ D ; si $FZ = 1$: $PC \leftarrow (D) \bmod 2^7 + 1$, $IR \leftarrow ((D) \bmod 2^7)$

a) Indica qué cambios serían necesarios en la Unidad de Proceso

b) Dibuja el grafo de estados correspondiente a IBEQ para la UC cableada de la Máquina Sencilla

c) Realiza la tabla de vectores de salida de cada estado para IBEQ

I.5. ⏱ [Ayuso, Ramos y Segarra 2008a, UZ] Dadas las microinstrucciones de la tabla 5.6:

a) Describe lo que hace cada una de las microinstrucciones si se ejecuta en la Máquina Sencilla, tal y como se ha visto en clase (movimientos entre registros o RTL)

b) Indica la secuencia de microinstrucciones para cada instrucción, e.g. CO_{10}=0,1,2,3

c) Describe lo que hace cada una de las instrucciones

@	Salto	Test	MX	ALU	$\overline{L}$/E	PC	IR	A	B	FZ
0	xxxx	111	00	xx	0	1	1	0	0	0
1	0101	100	10	xx	0	0	0	0	1	0
2	0100	101	11	xx	0	0	0	1	0	0
3	0000	011	11	10	1	0	0	0	0	1
4	0000	011	11	00	1	0	0	0	0	1
5	0111	101	00	xx	0	1	0	1	0	0
6	0000	011	xx	01	0	0	0	0	0	1
7	0000	011	11	00	1	0	0	0	0	1

Tabla 5.6: Microprograma para la Máquina Sencilla.

I.6. ⏱ [Ayuso, Ramos y Segarra 2008b, UZ] Usando la unidad de proceso de la Máquina Sencilla y conservando su funcionamiento básico, se quiere implementar un microprograma que realice las siguientes instrucciones:

00: Sumar fuente con inmediato y guardar en destino (ADD F K D)

01: Mover fuente a destino (MOV F D)

10: Compara fuente con inmediato (CMP F K)

11: Salto condicional a destino (BEQ D)

El formato de instrucción y modo de direccionamiento es el habitual en la MS y los operandos inmediatos están en los 16 bits siguientes a la instrucción que los requiera. Para ello se dispone de una plantilla de ROM (tabla 5.7) con ciertos valores ya predefinidos, a los que ha de adaptarse la programación. Rellena dicha plantilla con el microprograma, describe qué hace cada línea de forma concreta e indica a qué instrucción pertenece.

@	Salto	Test	MX	ALU	$\overline{L}$/E	PC	IR	A	B	FZ
0		111								
1	1000	001								
2	0101	000								
3										
4										
5										
6										
7										
8	1100	100								
9										
10										
11										
12										
13										

Tabla 5.7: Microprograma para la Máquina Sencilla.

I.7. ⏱ [Segarra 2012, UZ] Se desea sustituir la instrucción MOV de la máquina sencilla por una nueva instrucción MOVC de movimiento condicional. La nueva instrucción MOVC F,D debe mover el

contenido de la dirección de memoria F a la dirección de memoria D si FZ=1. Con FZ=0, esta instrucción no debe realizar ninguna acción. En caso de realizar el movimiento de datos, se deberá actualizar FZ indicando si el valor movido ha sido cero (FZ=1) o no (FZ=0). Si no ha habido movimiento, no se debe actualizar FZ.

a) Especifica los estados (con su correspondiente tabla de salidas) relativos a esta nueva instrucción (incluyendo *fetch* y decodificación) asumiendo que en el estado de decodificación se trae el primer operando (Máquina Sencilla cableada *optimizada*)

b) ¿Cuántos ciclos tardará en ejecutarse esta instrucción?

c) Reescribe el siguiente código ensamblador de la Máquina Sencilla original a un código ensamblador equivalente para un repertorio que incluya MOVC en lugar de MOV con el mínimo número de instrucciones

MS original		MS con MOVC	
Etiqueta	Instrucción	Etiqueta	Instrucción
	CMP a,b		
	BEQ mover		
	CMP a,a		
	BEQ fin		
mover:	MOV d,c		
fin:			

d) Indica el tiempo de ejecución del código original y del nuevo, para cualquier par de valores A y B, asumiendo una frecuencia de reloj de 100 MHz

5.3. Modificación de la unidad de proceso

M.1. La Máquina Sencilla estudiada en clase no permite la transferencia de datos con el exterior. Se llama bloque de entrada/salida (E/S) al conjunto de circuitos que permiten la transferencia de información entre CPU/memoria y los periféricos dedicados (teclado, pantalla, etc.). El bloque E/S más sencillo consiste en registros con los los que se puede leer y escribir información. La figura 5.2 muestra las modificaciones realizadas en la unidad de proceso (UP) de la máquina. Las dos instrucciones básicas que permitirán esta transferencia son:

OUT D: el contenido de la posición de memoria D se escribe en el registro de salida RS.

IN D: el dato a la entrada del registro de entrada RE se escribe en la dirección de memoria D.

Teniendo en cuenta todo lo anterior:

a) La instrucción IN necesita dos ciclos de reloj en su fase de búsqueda de operandos: en el primero se carga RE con el dato en su entrada y en el segundo se carga en B. ¿Por qué no se pueden realizar ambas cargas en el mismo ciclo?

b) Asumiendo que se *sustituyen* las instrucciones ADD y CMP por las instrucciones OUT e IN, escribe el microprograma correspondiente a estas dos nuevas instrucciones.

c) Asumiendo que hay que *añadir* las nuevas instrucciones OUT e IN al repertorio, indica un formato de instrucción de 16 bits que no sacrifique elementos existentes y escribe el microprograma correspondiente.

d) Si se modifican las dos instrucciones de E/S anteriores para acceder a más de un periférico («OUT N, D» y «IN N, D» donde N indica el número del periférico de entrada o de salida al que se quiere acceder), ¿cuál es el número máximo de periféricos al que podemos acceder?

M.2. Se quiere modificar la Máquina Sencilla de manera que permita almacenar información dentro de la unidad de proceso. Para ello se añade un banco de 4 registros de 16 bits de propósito general. Las instrucciones podrán tener cada operando o bien en memoria o bien en uno de los registros añadidos. La figura 5.3 muestra las modificaciones realizadas en la unidad de proceso. También se modifica el formato de instrucción según se muestra a continuación:

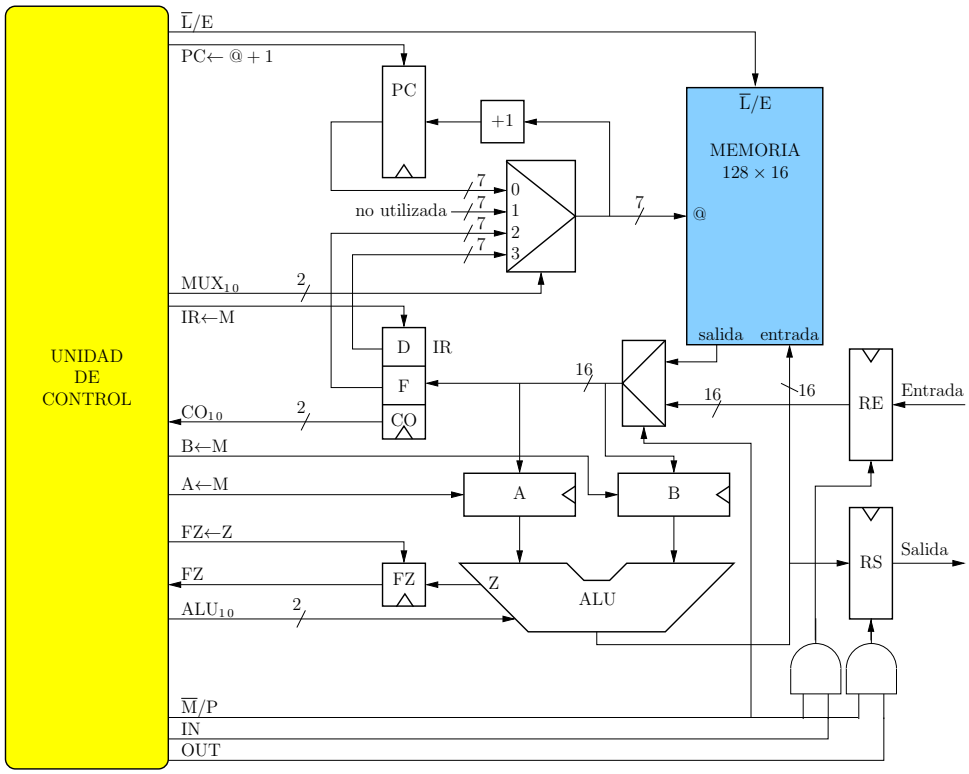


Figura 5.2: Máquina Sencilla modificada con entrada/salida.

2	1	6	1	6
CO	RMF	F	RMD	D

Los bits RMF y RMD controlan si el operando fuente o destino se encuentra en el banco de registros o en memoria, respectivamente. Si RMF = 0 querrá decir que el operando se encuentra en la dirección de memoria especificada en los 6 bits del campo F. Si RMF = 1 entonces el operando fuente se encuentra en el registro indicado por los 2 bits de menor peso del campo F. RMD indica lo mismo para el operando destino. Hay que recordar que sobre el operando destino se pueden hacer accesos tanto para lectura como para escritura.

a) ¿Qué finalidad tienen los multiplexores MXA y MXB de la figura? ¿Por qué no están controlados desde la UC como el resto de componentes de la UP?

b) Indica cuál es la misión de los dos bits de menor peso del bus de direcciones que están conectados al multiplexor MXR.

c) ¿Cuál es la finalidad de la señal WR procedente de la UC y los dos bits de menor peso del bus de direcciones que están conectados al codificador C1?

d) El bit RMD se envía hacia la UC. ¿En qué fase de la instrucción necesita conocerlo la UC?

e) Detalla la tabla de microprogramación para la instrucción ADD, teniendo en cuenta que cualquier operando se puede encontrar tanto en memoria como en uno de los registros internos.

M.3. [Segarra 2007, UZ] Se necesita que la máquina sencilla realice operaciones con números de 32 bits. Para ello se modifica la unidad de proceso como se muestra en la figura 5.4. Se sustituyen los registros A y B por los registros AH, AL, BH y BL, con H y L identificando los bits de mayor y menor peso de los números respectivamente. A su vez, la entrada de datos a memoria estará regulada por un multiplexor, que seleccionará entre la parte de mayor y menor peso del resultado de 32 bits proveniente de la ALU (la ALU ahora realiza operaciónes de 32 bits. ¡Cuidado con qué se selecciona en cada caso!). En la parte de direccionamiento a memoria también se añade circuitería para obtener de forma simple la dirección siguiente a una dirección dada cuando se requiera.

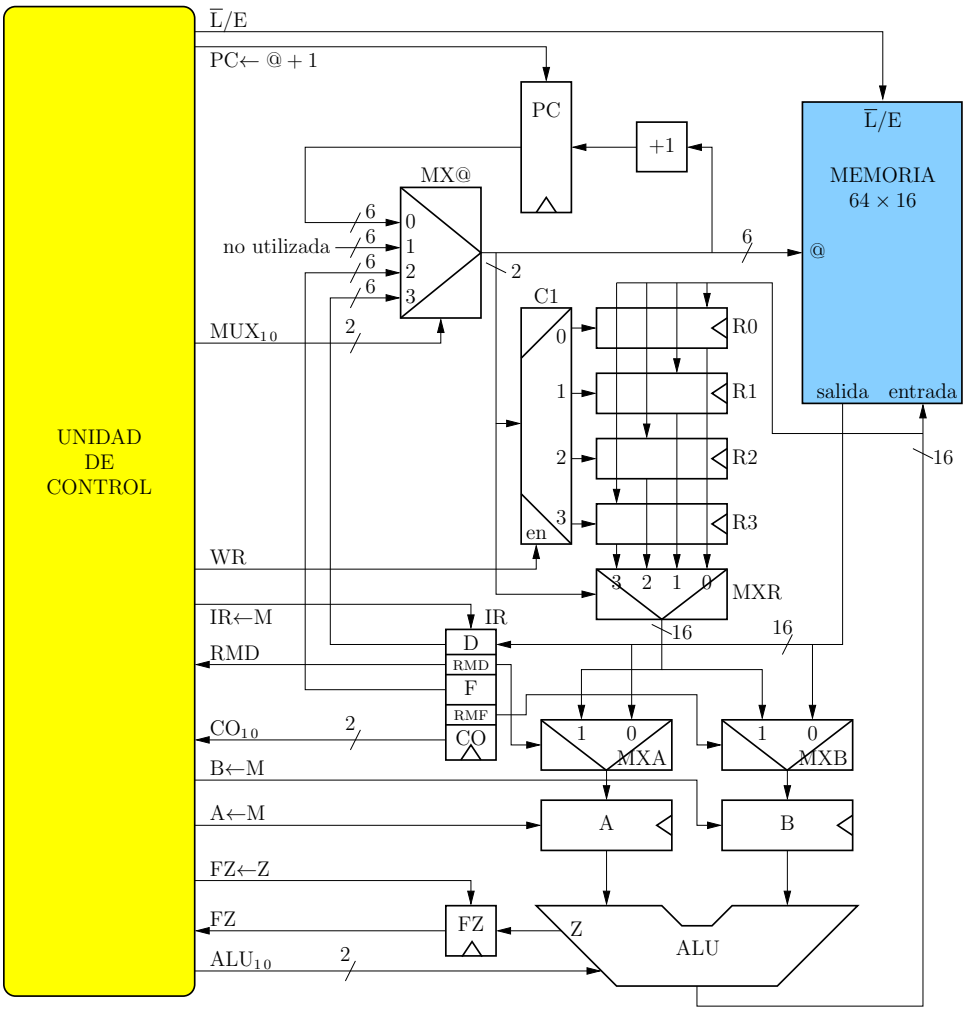


Figura 5.3: Máquina Sencilla modificada con registros de propósito general.

La nueva máquina utiliza el sistema *big-endian*, es decir, los números se almacenan en memoria con sus bits ordenados de mayor a menor peso.

Realiza el nuevo diagrama de estados de la operación ADD para dos operandos de 32 bits y rellena la tabla de salidas de la unidad de control para cada estado.

M.4. ⏱ [Segarra 2009, UZ] Se quiere ampliar la memoria de la Máquina Sencilla para poder ejecutar programas muy grandes. Esto implica que necesitamos un rango de direcciones de memoria mucho mayor. Para evitar otros cambios, se decide añadir un *registro de segmento* SR de 16 bits (figura 5.5). El contenido de dicho registro se concatenará a las direcciones de 7 bits originales para formar direcciones de 23 bits. Así, el contenido de SR serán los 16 bits de mayor peso de la nueva dirección de 23 bits, y los 7 bits de la dirección original serán los 7 bits de menor peso de la nueva dirección. Además, SR dispone de una línea de activación de carga paralela, igual que los demás registros.

Para que todo funcione necesitaremos *sustituir* la instrucción ADD por una nueva instrucción en el repertorio original que actualice SR:

MOVSR K ; SR←K

K es un dato inmediato, y el formato de instrucción es el visto en clase. Los bits de código de operación son: 00-MOVSR, 01-MOV, 10-CMP, 11-BEQ.

a) Responde a las siguientes preguntas:

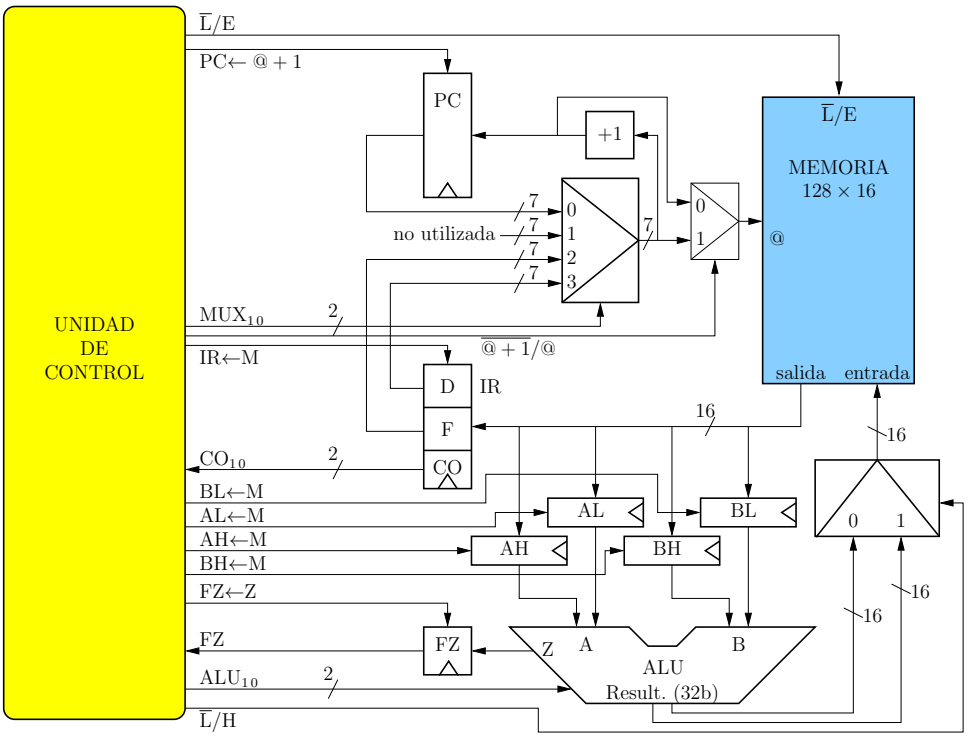


Figura 5.4: Máquina Sencilla modificada para operar con 32 bits.

1) ¿Cuántas direcciones tendrá esta nueva Máquina Sencilla?
2) ¿Cuál será la capacidad de la memoria expresada en bits? ¿Y en bytes?
3) El rango de direcciones de memoria se podría aumentar simplemente haciendo que las direcciones fueran de 23 bits en lugar de 7. De esta forma no haría falta complicarnos con el registro de segmento. ¿Qué inconvenientes tendría esta solución?

b) Para la unidad de control cableada, indica la tabla de estados con las salidas de cada uno de los estados que necesites, sólo para la instrucción MOVSR, asumiendo que el primer estado de la secuencia corresponde a la fase de búsqueda de la instrucción

c) Dada la tabla 5.8 con contenidos de memoria, indica para las instrucciones la descripción de su contenido y para los datos los cambios que se producirían en su ejecución, empezando la ejecución en la dirección 56 y terminando en la primera dirección para la cual no se muestra contenido en la tabla. Considerar que inicialmente el contenido de SR es 0.

M.5. ⏱ [Segarra 2011, UZ] Se necesita modificar la Máquina Sencilla para que el procesador (UC y UP) funcione a una frecuencia de reloj mucho más rápida que la memoria. Para ello se modifica la Máquina Sencilla tal y como se muestra en la figura 5.6. Se decide optar por una implementación en la que el procesador solicite una acción (lectura/escritura) a la memoria y ésta responda en cuanto pueda. La petición se realizará poniendo la línea REQ a 1, especificando también el tipo de petición (lectura/escritura), la dirección que corresponda y, en escrituras, el dato a escribir. Todas estas señales han de mantenerse estables mientras la memoria esté ocupada con dicha petición. Cuando la memoria detecte la petición, pondrá inmediatamente (en mucho menos de un ciclo de CPU) la línea BUSY a 1 y la mantendrá activa mientras esté llevando a cabo la petición solicitada. En escrituras, cuando la memoria haya completado la escritura, pondrá BUSY a 0. En lecturas, cuando la memoria tenga listo el dato a leer lo pondrá en las líneas de datos durante al menos un ciclo de CPU, y después pondrá BUSY a 0. Es decir, en una lectura, cuando BUSY pase a valer 0, el dato ya no estará en la línea de datos. En cualquier caso, cuando BUSY pase a valer 0 se debe desactivar la línea REQ (a no ser que se quiera realizar otra petición, que implicaría poner REQ a 1) y ya se pueden alterar las líneas implicadas.

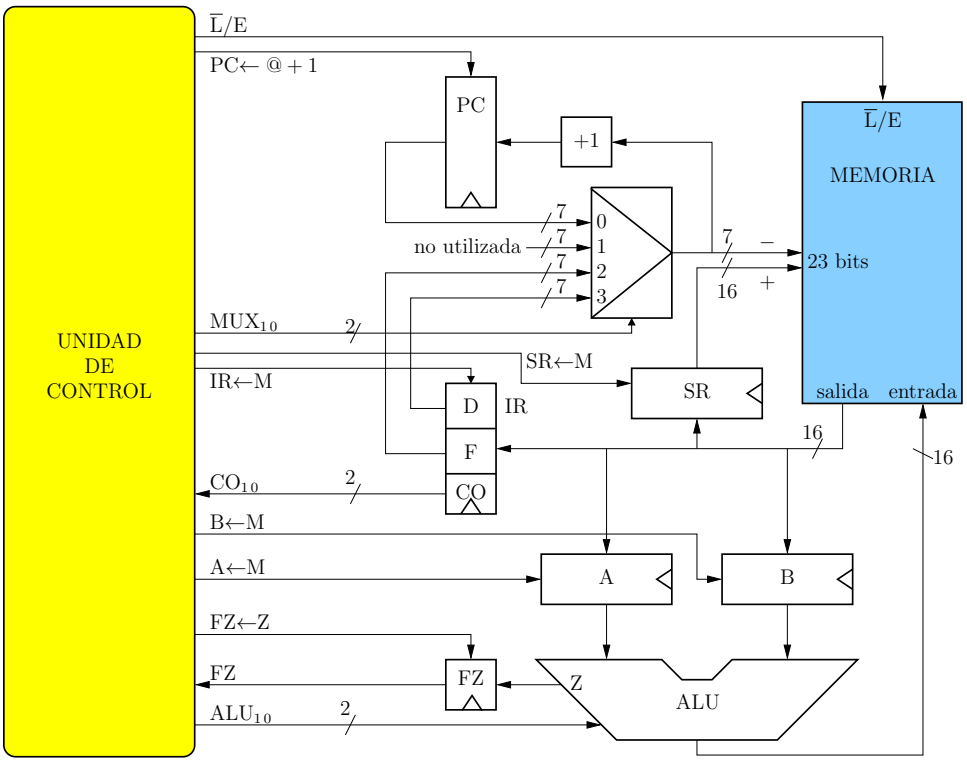


Figura 5.5: Máquina Sencilla modificada con registro de segmento.

Asumiendo una implementación microprogramada:

a) ¿Cuántos bits de test se necesitarán si se usa la misma filosofía de formato de microinstrucción vista en clase?

b) Especifica el contenido del microprograma para poder ejecutar correctamente la instrucción ADD, incluyendo el fetch y la decodificación

c) ¿Cuántos ciclos durará la ejecución de la instrucción ADD como máximo?

Dir.	Contenido previo	Ensamblador / Contenido tras ejecutar
...	...	...
56	0111001001100101	
57	0000000000000000	
58	0000000000000100	
...	...	...
100	0000000000000011	
101	0000000000000100	
...	...	...
571	0111001001100101	
...	...	...
612	0000000000000101	
613	0000000000000110	
...	...	...

Tabla 5.8: Contenido de memoria de la Máquina Sencilla con registro de segmento, en binario, para decodificar y ejecutar.

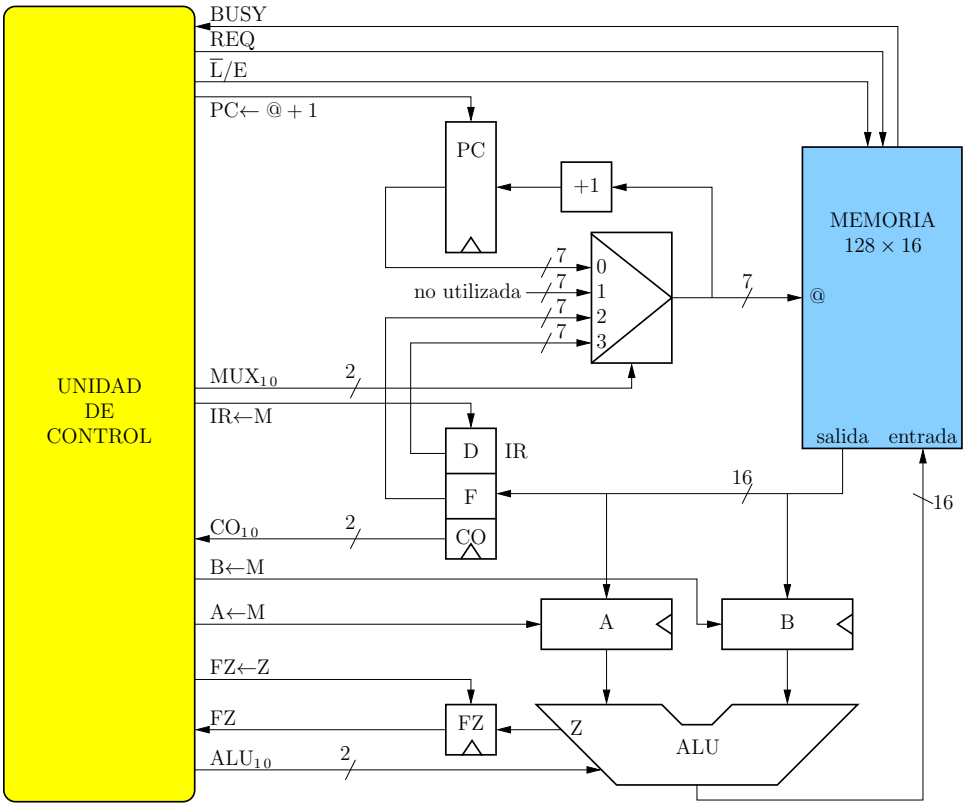


Figura 5.6: Máquina Sencilla modificada para memoria lenta.

Bibliografía

Ayuso, Natalia, Luis M. Ramos y Juan Segarra, eds. (jul. de 2004). *Examen de Sistemas Lógicos 03-04, 2ª conv.* Universidad de Zaragoza.

— eds. (sep. de 2005). *Examen de Sistemas Lógicos 04-05, 2ª conv.* Universidad de Zaragoza.

— eds. (sep. de 2006). *Examen de Sistemas Lógicos 05-06, 2ª conv.* Universidad de Zaragoza.

— eds. (sep. de 2007). *Examen de Sistemas Lógicos 06-07, 2ª conv.* Universidad de Zaragoza.

— eds. (feb. de 2008a). *Examen de Sistemas Lógicos 07-08, 1ª conv.* Universidad de Zaragoza.

— eds. (sep. de 2008b). *Examen de Sistemas Lógicos 07-08, 2ª conv.* Universidad de Zaragoza.

— eds. (sep. de 2011). *Examen de Introducción a los Computadores 10-11, 2ª conv.* Universidad de Zaragoza.

— eds. (sep. de 2012). *Examen de Introducción a los Computadores 11-12, 2ª conv.* Universidad de Zaragoza.

Segarra, Juan (sep. de 2004a). "Ejercicio 4". En: *Examen de Sistemas Lógicos 03-04, 3ª conv.* Universidad de Zaragoza.

— (ene. de 2004b). "Ejercicio 6". En: *Examen de Sistemas Lógicos 03-04, 1ª conv.* Universidad de Zaragoza.

— (feb. de 2006). "Ejercicio 5". En: *Examen de Sistemas Lógicos 05-06, 1ª conv.* Universidad de Zaragoza.

— (feb. de 2007). "Ejercicio 5". En: *Examen de Sistemas Lógicos 06-07, 1ª conv.* Universidad de Zaragoza.

— (sep. de 2009). "Ejercicio 4". En: *Examen de Sistemas Lógicos 08-09, 2ª conv.* Universidad de Zaragoza.

— (feb. de 2011). "Ejercicio 4". En: *Examen de Introducción a los Computadores 10-11, 1ª conv.* Universidad de Zaragoza.

— (feb. de 2012). "Ejercicio 4". En: *Examen de Introducción a los Computadores 11-12, 1ª conv.* Universidad de Zaragoza.

— (jun. de 2019). "Ejercicio 1". En: *Examen de Introducción a los Computadores 18-19 GITST, 1ª conv.* Universidad de Zaragoza.

Viñals, Víctor (sep. de 2019). "Ejercicio 1". En: *Examen de Introducción a los Computadores 18-19, 2ª conv.* Universidad de Zaragoza.