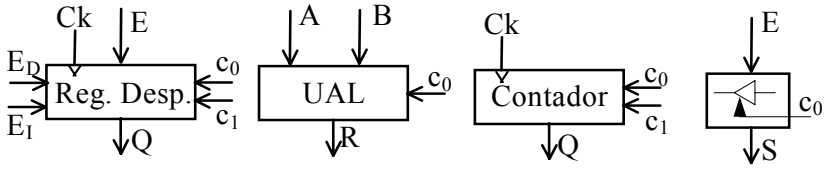
```
6:              B ← 0;
7:            endif;
8:            Despl.CerradoDcha(A);
9:            Despl.CerradoDcha(A);
10:         endfor;
11:         Bus ← B;
12:         Parar;
```



c_1c_0	Q(t+1)
00	Q(t)
01	Des. Dcha.
10	Des. Izq.
11	Carga

c_0	R
0	A+B
1	A-B

c_1c_0	Q(t+1)
00	Q(t)
01	0
10	Q(t)+1 mod 8
11	Q(t)

c_0	S
0	-
1	E

Figura 1: Módulos secuenciales del problema con sus tablas de funcionamiento.

Solución

a) Del análisis de las operaciones del algoritmo propuesto se puede concluir que:

- Es necesario un registro de 8 bits llamado A. Este registro debe poder realizar dos operaciones:
  - Carga de 8 bits de datos desde el bus, correspondiente a la operación *A ← Bus*. El tipo de registro disponible es capaz de cargar en paralelo, y puede realizar esta operación.
  - Desplazamiento **cerrado** a la derecha, correspondiente a la operación *Despl.CerradoDcha(A)*. En este caso el tipo de registro sólo realiza la operación Despl.Dcha, con lo que se hace necesario conectar el bit menos significativo de A (llamado A_0), a la entrada E_D del registro.

En la Figura 2 se puede observar el esquema de conexiones para este registro:

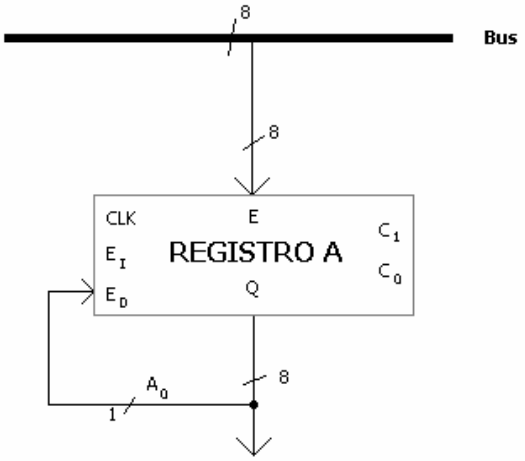


Figura 2: Diagramas de conexiones para el registro A.

- Es necesario generar una señal de condición, correspondiente a la operación *if* $A_1A_0 \neq 01$ *then*, que se llamará s_1. Para ello se puede usar la expresión equivalente $NOT(\bar{A}_1\ A_0) = A_1 + \bar{A}_0$ de tal forma que se generará dicha condición usando una puerta OR y un inversor, como se muestra en la Figura 3.

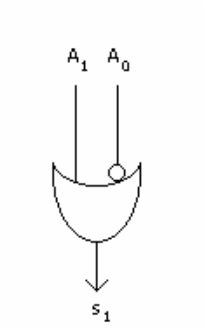


Figura 3: Generación de la señal de condición s_1.

Otra opción podría ser generar la condición s_1 como $\bar{A}_1\ AND\ A_0$ y tenerlo en cuenta a la hora de realizar el diagrama de estados considerando la operación *if* $A_1A_0 \neq 01$ *then* verdadera cuando no se cumple s_1 (es decir, NOT s_1).

- Es necesario un registro de 8 bits llamado B. Este registro debe ser capaz de cargar dos valores constantes "0" y "1" (Operaciones $B \leftarrow 0$, $B \leftarrow 1$) y volcar su contenido al bus (Operación $Bus \leftarrow B$).
  - Para volcar el contenido del registro al bus es necesario usar una puerta triestado que controle la operación de escritura en el bus.
  - Para distinguir el valor a cargar se podría usar un multiplexor con dos entradas constantes 0 y 1 y usar el valor de salida del multiplexor 2 a 1 como bit menos significativo de la entrada E del registro B, poniendo al valor constante 0 el resto de bits de la entrada E del registro. Pero en este caso se puede eliminar dicho multiplexor y usar una señal de control (generada por la unidad de control) que determina el valor del bit menos significativo de la entrada. Si llamamos a esta señal de control C_{LOADB}, cuando sea igual a 1 se cargará el valor "1" en el registro B y si es igual a 0 se cargara "0" en B. En la Figura 4 se muestra el diagrama de conexiones para el registro B.

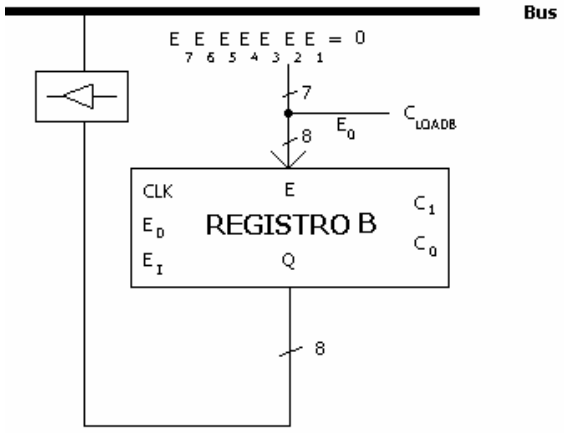


Figura 4: Diagramas de conexiones para el registro A.

- Finalmente es necesario implementar la lógica de la operación *for Cont=0 to 3 do* usando un contador de tamaño adecuado. En este caso se debe contar de 0 a 3 y se dispone de un contador módulo 8 (con tres estados $Cont_2Cont_1Cont_0$) con lo que se hará necesario detectar el fin de cuenta (Cont=3) usando una señal de condición s_0. Esta señal de condición se puede generar usando la condición $Cont_1Cont_0$=11 o bien usar la señal $Cont_2$ como señal de fin de cuenta (si Cont2=1 entonces Cont>3). Se va a usar esta segunda aproximación porque de esta forma no es necesario usar una puerta AND de las entradas $Cont_1$ y $Cont_0$. El diagrama de conexiones quedaría tan simple como el que se muestra en la Figura 5.

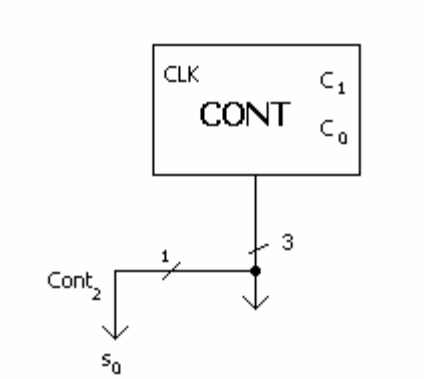


Figura 5: Diagramas de conexiones para el registro A.

Una vez definidos los recursos necesarios para implementar la unidad de procesamiento y las señales de condición que debe generar, sólo quedan por definir las señales de control que la unidad de control debe generar. En la Tabla 1 se muestran las definiciones de cada una de las señales de control, la operación que realizan y el valor de las señales de control para dicha operación.

Señal de control	**Operación a realizar**	**Valor**
C_{A0}, C_{A1}	Cargar registro A	11
C_{A0}, C_{A1}	Desplazamiento cerrado a la derecha de A	10
C_B,	Cargar registro B con 0	1
C_B, C_{LOADB}	Cargar registro B con 1	11
C_{C0}, C_{C1}	Resetear contador	10
C_{C0}, C_{C1}	Incrementar módulo 8 una unidad el contador.	01
C_{BUS}	Volcar el contenido del registro B en el bus	1

Tabla 1: Señales de control que debe generar la Unidad de Control para cada operación.

De esta tabla se puede deducir que las variables de control del registro A y el contador se corresponden con las entradas c_1 y c_0 de los mismos, pero en el caso del registro B solo es necesario usar una señal de control, c_B, asociada a la carga (si c_B=1 → c_1c_0=11) y a la conservación el estado (si c_B=0 → c_1c_0=00). La señal C_{BUS} es la señal de control de la puerta triestado que controla la escritura en el bus.

Con todo esto, el diagrama de la unidad de procesamiento queda como se muestra en la Figura 6.

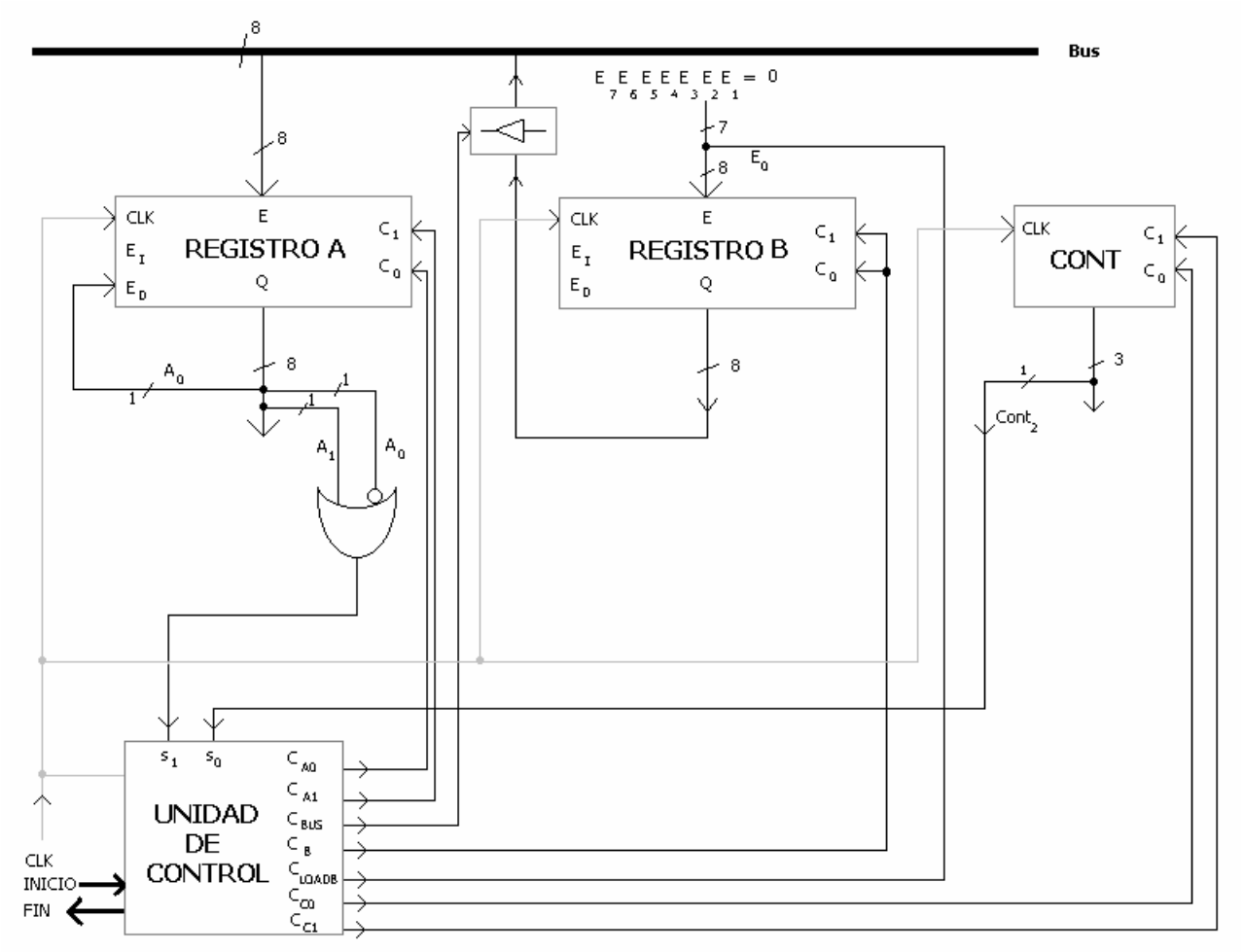


Figura 6: Diseño de la Unidad de Procesamiento.

b) Para describir el funcionamiento de la Unidad de Control solicitada en este apartado, se ha diseñado el diagrama de transición de estados de la Figura 7, donde el significado detallado de cada uno de los estados propuestos viene dado por la Tabla 2. Obsérvese que esta Unidad de Control se ha diseñado como una máquina de Moore. (Dada la sencillez del algoritmo propuesto, se ha obtenido directamente el diagrama de estados a partir de éste, sin necesidad de pasar por el diagrama ASM.)

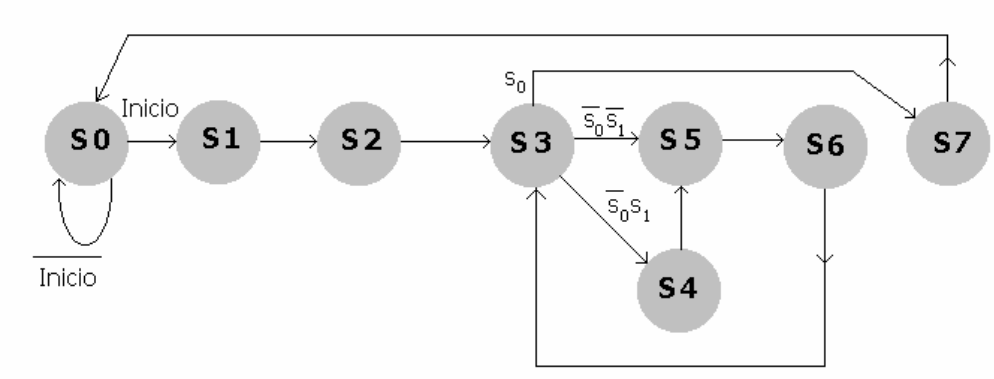


Figura 7: Diagrama de estados de la unidad de control.

Estado de la unidad de control	Microoperaciones efectuadas	Señales de control a activar
S_0	*Ninguna (Estado inicial)*	*Ninguna*
S_1	A ←Bus, Cont ←0	C_{A0}, C_{A1}, C_{C0}
S_2	B ←1	C_B, C_{LOADB}
S_3	*Ninguna (Comprobamos condiciones s_1 y s_0)*	*Ninguna*
S_4	B←0	C_B
S_5	Despl.CerradoDcha(A)	C_{A0}
S_6	Despl.CerradoDcha(A), Cont ←Cont+1 (mod 8)	C_{A0}, C_{C1}
S_7	Bus ← B	C_{BUS}

Tabla 2: Acciones tomadas por la Unidad de Control en cada estado.

Este diagrama de transición de estados cumple todos los requisitos para ejecutar el algoritmo propuesto en el enunciado del problema utilizando la Unidad de Procesamiento diseñada en el apartado anterior. Se puede
comprobar, por ejemplo, que se ejecuta el bucle *for* cuatro veces, ya que se incrementa el contador en la última microoperación que se ejecuta dentro del bucle, de tal forma que cuando se llega al valor Cont=4 se activa la señal s_0 y se termina el bucle. Obsérvese que siempre se deben ejecutar las dos microoperaciones de desplazamiento cerrado del registro A independientemente de la condición $A_1A_0 \neq 01$ y que además se debe hacer en dos estados consecutivos.

La Unidad de Control se realiza utilizando la técnica de los *elementos de retardo*, según pide el enunciado,
asignando un biestable tipo D a cada estado, tal como se muestra en la Figura 8. También se muestra cómo se forman las distintas señales de control a partir de las salidas de los biestables tipo D [ver las páginas 304 a 307 del texto base de teoría]. Obsérvese que los bloques de decisión corresponden a demultiplexores, según se muestra en la Figura 8 [ver las páginas 304 a 307 del texto base de teoría].

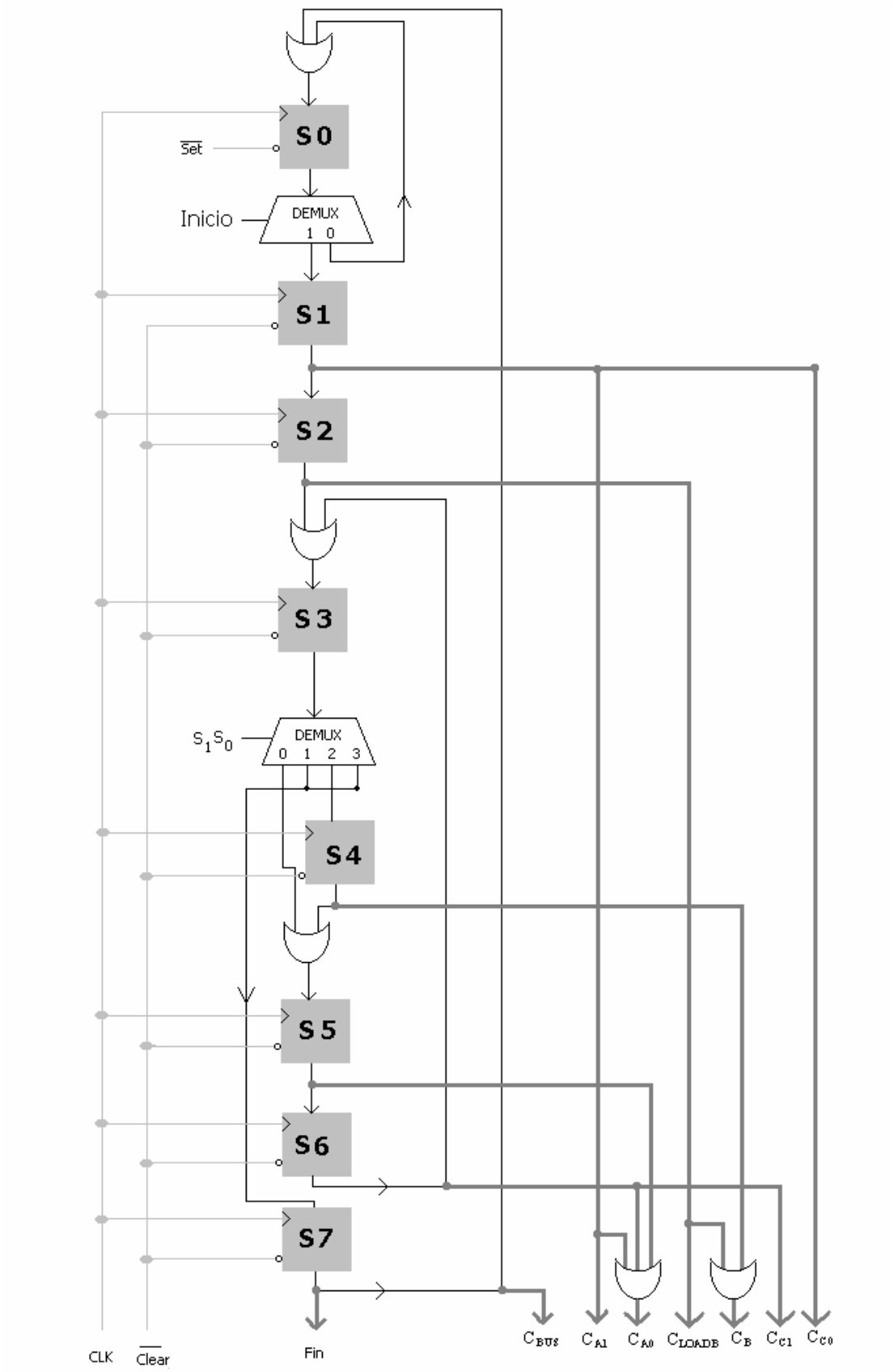


Figura 8: Unidad de Control mediante *elementos de retardo*.