U.B.A. - FACULTAD DE INGENIERÍA

Departamento de Electrónica

(66.70) Estructura del Computador

"Diseño de la lógica de un sistema de semáforos"

- **Solución cableada VS. Solución por software -**

2º Cuatrimestre 2012

Turno: Viernes 16 – 19 hs.

GRUPO N° 3		
Apellido y Nombre	**Número de padrón**	**Dirección de correo electrónico**
Alonso, Juan Manuel		
Luraschi, Matías		
Rinaldi, Lautaro Ezequiel		

Calificación :
Firma de Aprobación :

Observaciones:

Enunciado Completo del Trabajo Práctico:

"Diseño de la lógica de un sistema de semáforos"

Se le ha solicitado que desarrolle el sistema para el funcionamiento nocturno del par de semáforos que se encuentran en la esquina de una estación de Bomberos. A continuación se describe cómo debe ser el funcionamiento de los mismos:

- Ambos semáforos se encontrarán por defecto en un estado en el cual la luz amarilla se enciende de forma intermitente (1 seg. prendida – 1 seg. apagada).

- Cuando un peatón desea cruzar, debe pulsar el botón que se encuentra debajo del semáforo. En tal caso, los semáforos seguirán la siguiente secuencia:

  - Por 5 segundos se mantendrá encendida la luz amarilla en el semáforo 1, y se encenderá la luz roja en el semáforo 2.
  - Luego, se encenderá la luz verde del semáforo 1 dejando la luz roja en el semáforo 2. Se permanecerá en este estado por una duración de 30 segundos.
  - Transcurrido ese período, se encenderá la luz amarilla del semáforo 1 mientras que el semáforo 2 continúa con la luz roja encendida.
  - Una vez transcurridos 5 segundos, se enciende la luz roja del semáforo 1 mientras que se pone en amarillo el semáforo 2. Se quedará en este estado por 5 segundos.
  - Ahora deberá permanecer el semáforo 1 en rojo mientras que el semáforo 2 prende únicamente la luz verde, quedando en este estado por otros 30 segundos.
  - Cumplido dicho tiempo, se procede a encender la luz amarilla exclusivamente en el semáforo 2, mientras que en el semáforo 1 se mantiene encendida la luz roja.
  - Luego de 5 segundos, se procede a volver al estado por defecto, en el cual ambos semáforos encienden de forma intermitente sus luces amarillas.
  - En caso de volverse a presionar el botón para el cruce de los peatones mientras se ejecuta la secuencia anterior, éste no tiene ningún efecto.
  - Existe también un botón que es utilizado al momento que deben salir los camiones de Bomberos. Cuando este es presionado ambos semáforos deben pasar a encender su luz roja y su luz amarilla simultáneamente. Debido a que el tiempo requerido para la salida de los camiones no es conocido, se debe esperar a que este botón sea pulsado nuevamente para volver al estado por defecto de los semáforos (sin importar en qué estados se encontraban previamente).
  - Para tener referencia temporal, existe una señal de reloj de ***32Hz*** que puede ser utilizada en cada uno de los semáforos.

Diseño

A – Definir cuáles son las entradas y las salidas del sistema a desarrollar.
B – Plantear el diagrama de estados del funcionamiento del sistema a implementar. Debe indicarse claramente los estados presentes y las condiciones de transición entre cada uno de ellos.

Solución cableada

Se dispone de compuertas, Flip-Flops y demás componentes electrónicos cuyas hojas de datos están en el archivo adjunto.

C – Presentar una tabla simplificada que indica el comportamiento del sistema. Completar la tabla de estados, en donde se muestren todos los estados posibles del sistema. (*Sugerencia: Plantear las variables en un orden adecuado para poder reducirla y así tener que evaluar una menor cantidad de estados*)
D – Simplificar la lógica requerida, obteniendo todas las ecuaciones mínimas tanto por 1s como por 0s. Determine cuál de todas ellas es conveniente usar para su implementación y justifique su respuesta.
E – Presentar un diagrama de bloques que indique cómo está compuesta la solución cableada obtenida. Muestre cómo dichos bloques se encuentran conectados entre sí. Detalle qué componentes conforman cada uno de los bloques.
F – Presentar el esquema circuital de la solución obtenida, detallando todos los componentes utilizados en la misma.

Solución por software

Se pide diseñar un código Assembly ARC que implemente la lógica de control descripta.
Los dos botones y las luces de los semáforos se encuentran mapeados a los bits de la 0xD6000020, tal como se indica a continuación:

Bit0: Botón de peatón.
Bit1: Botón para salida de bomberos.

Bit16: Luz verde semáforo 1.
Bit17: Luz amarilla semáforo 1.
Bit18: Luz roja semáforo 1.

Bit24: Luz verde semáforo 2.
Bit25: Luz amarilla semáforo 2.
Bit26: Luz roja semáforo 2.

Suponer que la frecuencia de reloj del procesador es de 1GHz.

G – Presentar el código ARC apropiadamente documentado.
H – Comparación detallada entre las dos soluciones obtenidas.

***Obs*:**

En el enunciado debió corregirse la frecuencia de la señal de reloj de 32 Khz a 32 Hz. También donde detallaba en que bits se encontraban mapeadas las luces en memoria se corrigió bit126 por bit26.

Entradas y Salidas del Sistema a Desarrollar

Entradas:

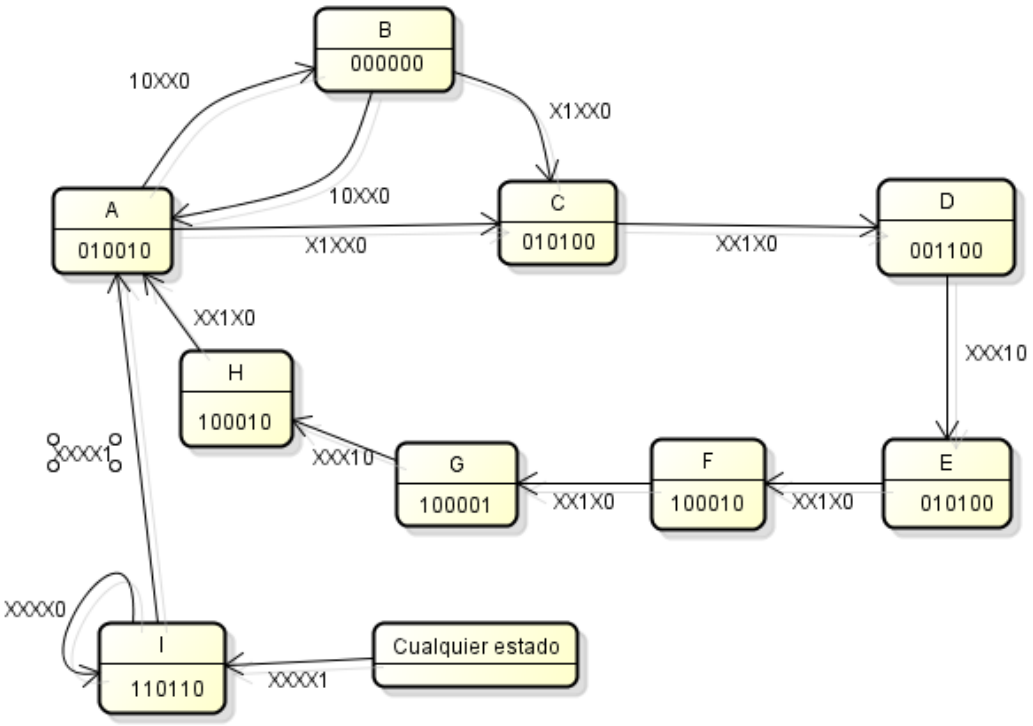
- **T1**: Timer de 1 segundo.
- **BP**: El botón que utilizan los peatones cuando desean cruzar.
- **T5**: Timer de 5 segundos.
- **T30**: Timer de 30 segundos.
- **BB**: El botón que se pulsa cuando sale un camión de Bomberos.

Salidas:

- Luces del semáforo 1:
  - **R1**: Rojo
  - **A1**: Amarillo
  - **V1**: Verde

- Luces del semáforo 2:
  - **R2**: Rojo
  - **A2**: Amarillo
  - **V2**: Verde

Diagrama de estados del funcionamiento del sistema



El orden de las entradas es: T1 BP T5 T30 BB, el significado de cada uno fue aclarado anteriormente.
El orden de los estados es: R1 A1 V1 R2 A2 V2 donde R1 A1 V1 corresponden a las luces roja, amarilla y verde del primer semáforo y R2 A2 V2 a las luces roja, amarilla y verde del segundo.

Como tenemos 9 estados, con 4 Flip-Flops alcanza para resolver la lógica del sistema planteado, quedando 7 estados prohibidos de los 16 posibles.

Además, como tendremos 4 Salidas de Flip-Flops, posteriormente deberemos implementar un circuito decodificador para obtener, a partir de ellas, las 6 salidas de nuestro circuito.

Tabla de estados completa

La tabla de estados se envía en el archivo adjunto, llamado Tabla_estados.xlsx debido a su extensión, para no complicar la lectura del presente informe.

Simplificación de la Lógica Requerida

A la hora de decidir si implementar la función simplificada por 0's ó por 1's, se utilizó el siguiente orden de prioridades:

1. Menor cantidad de niveles.
2. Menor cantidad de términos.
3. Menor cantidad de entradas por compuerta.

A continuación se detallan las ecuaciones correspondientes a la lógica de los Flips-Flops, para que de un determinado estado pase al siguiente estado que nosotros deseamos. También se muestra la elección tomada:

Como los diagramas de Karnaugh que requeríamos realizar son de 9 variables, son imposibles de resolver gráficamente, por lo que debe utilizarse algún método algorítmico como ser Quine-McCluskey. Por este motivo, utilizamos unl software que nos permite simplificar funciones de hasta 16 variables

Planteamos las ecuaciones por 1's:

- J3 = BB
- K3 = BB

- J2 = Q1 Q0 T30 BB'
- K2 = BB + Q1 Q0 T5

- J1 = Q3' Q2' BP BB' + Q3' Q2 Q0 T5 BB'
- K1 = BB + Q2' Q0 T30 + Q2 Q0 T5

- J0 = Q3' Q2' Q1' T1 BP' BB' + Q3' Q2' Q1 T5 BB' + Q3' Q2 Q1' T5 BB' + Q3' Q2 Q1 T30 BB'
- K0 = BB + Q2' Q1' BP + Q2' Q1' T1 + Q2' Q1 T30 + Q2 T5

Planteamos las ecuaciones por 0's:

- J3 = BB
- K3 = BB

- J2 = Q1 Q0 T30 BB'
- K2 = (Q1 + BB) (Q0 + BB) (T5 + BB)

- J1 = (Q2 + BP) BB' (Q2' + Q0) (Q2' + T5) Q3'
- K1 = (Q2 + T30 + BB) (Q0 + BB) (Q2' + T5 + BB)

- J0 = (Q2 + Q1 + T1) BB' (Q2 + Q1 + BP') (Q2 + Q1' + T5) (Q2' + Q1 + T5) (Q2' + Q1' + T30) Q3'
- K0 = (Q2 + Q1 + T1 + BP + BB) (Q2 + Q1' + T30 + BB) (Q2' + T5 + BB)

Análisis de conveniencia de implementación:

Obs: Notación: # Cantidad de compuertas necesaria TIPO DE COMPUERTA (cantidad de entradas de la compuerta)

Función a implementar	1s	0s	Elección
J3	-	-	-
K3	-	-	-
J2	1 AND (4)	1 AND (4)	-
K2	1 OR (2), 1 AND (3)	3 OR (2), 1 AND (3)	1s
J1	2 AND (4), 1 OR (2)	3 OR (2), 1 AND (5)	1s
K1	2 AND (3), 1 OR (3)	2 OR (3), 1 OR (2), 1 AND (3)	1s
J0	1 AND (6), 3 AND (5), 1 OR (4)	5 OR (3), 1 AND (7)	1s
K0	3 AND (3), 1 AND (2), 1 OR (5)	1 OR (5), 1 OR (4), 1 OR (3) 1 AND (3)	1s

A continuación se detallan las ecuaciones correspondientes a la lógica de decodificación de los estados de salida de los Flips-Flops, ya que como el circuito tiene 6 salidas (3 luces por cada semáforo) pero tan sólo 4 salidas de los Flips-Flops, es necesario hacer una decodificación. También se muestra la elección tomada:

Planteamos las ecuaciones por 1's:

- R1 = Q2 Q1 Q0' + Q3 + Q2 Q0
- A1 = Q2' Q0' + Q2 Q1' Q0'

- V1 = Q2' Q1 Q0
- R2 = Q2' Q1 + Q2 Q1' Q0' + Q3

- A2 = Q2' Q1' Q0' + Q2 Q0
- V2 = Q2 Q1 Q0'

Planteamos las ecuaciones por 0's:

- R1 = (Q3 + Q2) (Q3 + Q1 + Q0)
- A1 = Q0' (Q2' + Q1')

- V1 = Q1 Q0 Q2'
- R2 = (Q3 + Q2 + Q1) (Q3 + Q1 + Q0') (Q3 + Q2' + Q1')

- A2 = (Q2 + Q0') (Q2 + Q1') (Q2' + Q0)
- V2 = Q2 Q1 Q0'

Análisis de conveniencia de implementación:

Obs: Notación: # Cantidad de compuertas necesaria TIPO DE COMPUERTA (cantidad de entradas de la compuerta)

Función a implementar	1s	0s	Elección
R1	1 OR (3), 1 AND (3), 1 AND (2)	1 OR (2), 1 OR (3), 1 AND (2)	1s
A1	1 AND (3), 1 AND (2)	1 OR (2), 1 AND (2)	1s
V1	1 AND (3)	1 AND (3)	1s
R2	1 AND (3), 1 OR (3), 1 AND (2)	1 AND (3), 3 OR (3)	1s
A2	1 AND (3), 1 AND (2), 1 OR (2)	1 AND (3), 3 OR (2)	1s
V2	1 AND (3)	1 AND (3)	1s

Diagrama en bloques de la solución cableada

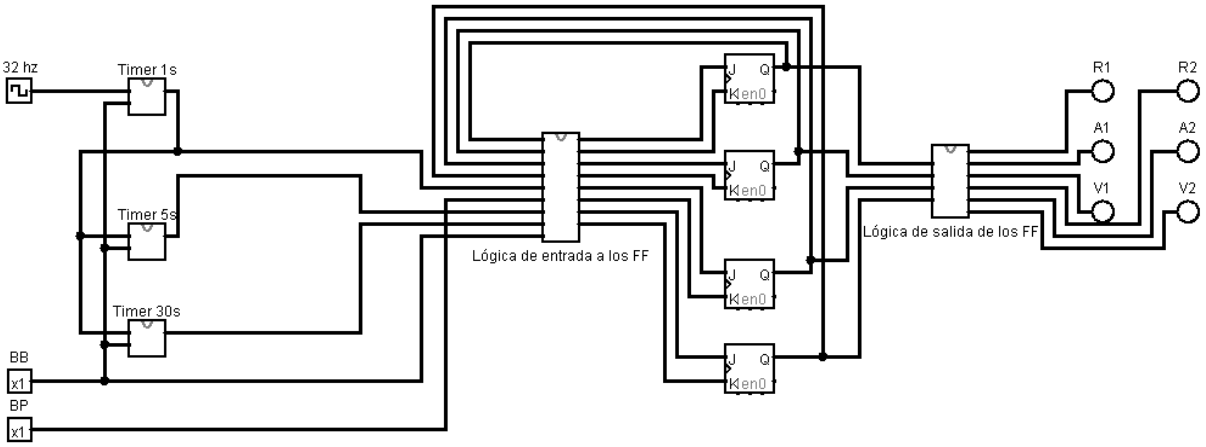
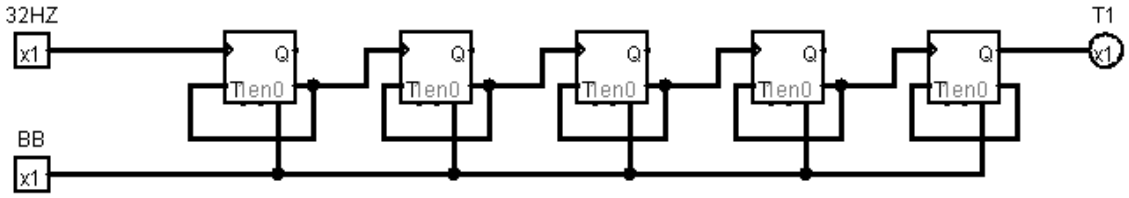


Diagrama Circuital de la solución cableada

Como el diagrama es bastante grande, se muestra en bloques separados, y cada uno de ellos tiene nodos de interconexión con los otros bloques con nombres comunes a todos ellos:

Timer de 1 segundo:

Se utilizan 5 Flip-Flops en cascada, ya que cada uno de ellos divide a la mitad la frecuencia, teniendo en T1 una salida de 1Hz:



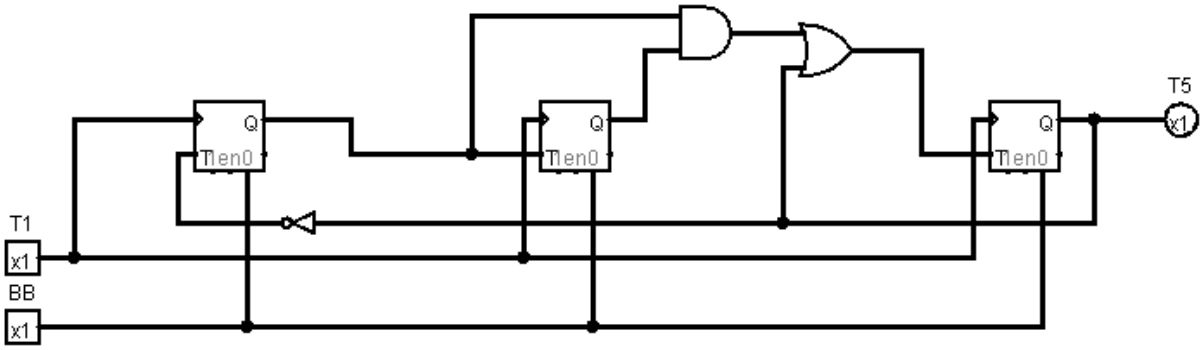
Entradas:

- Clock de 32Hz
- BB (Botón de los bomberos para resetear el conteo)

Salida:

- T1 (timer de 1 segundo)

Timer de 5 segundos:



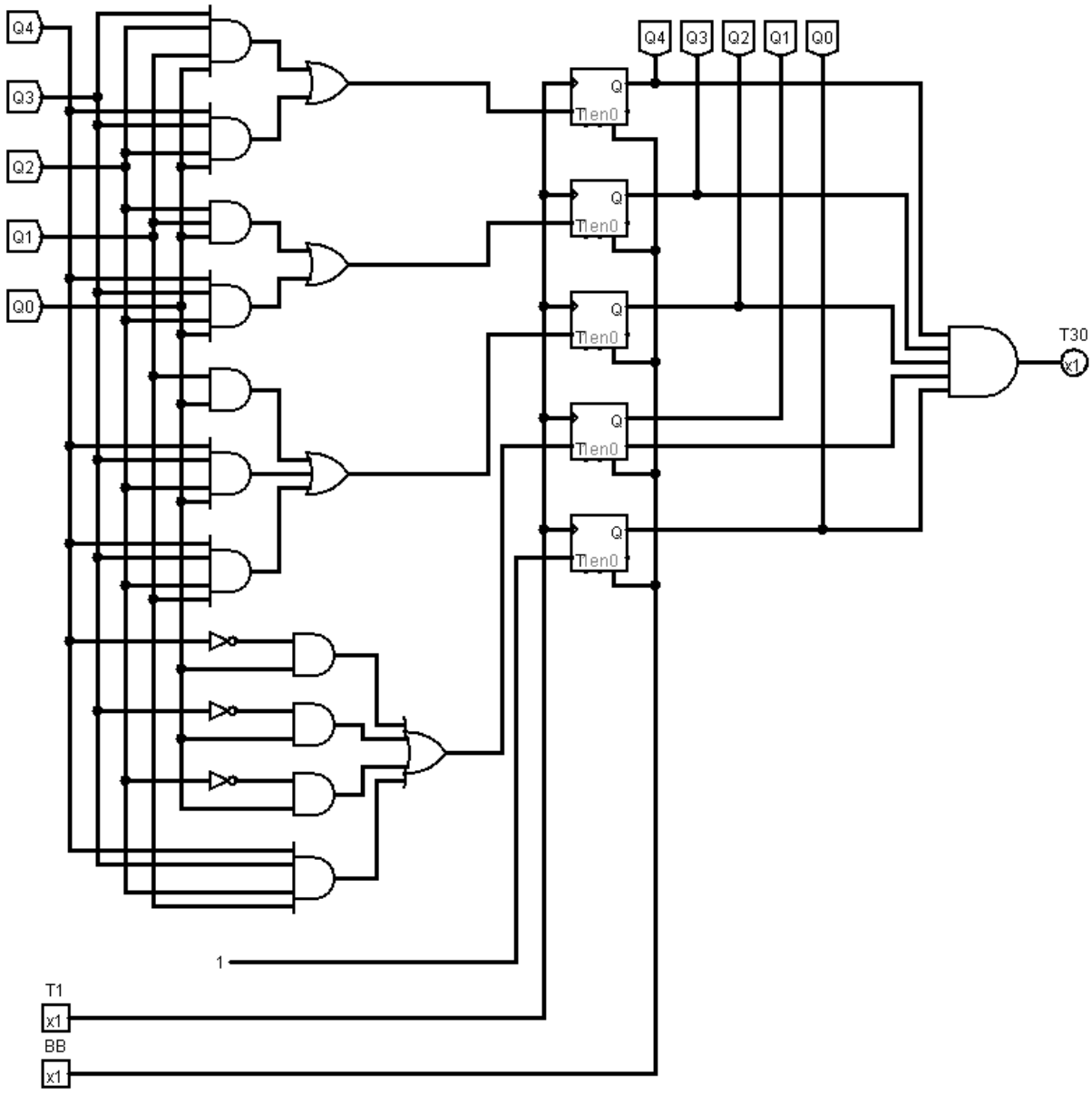
Entradas:

- T1 (timer de 1 segundo)
- BB (botón de los bomberos para resetear el conteo).

Salida:

- T5 (timer de 5 segundos)

Timer de 30 segundos:



Entradas:

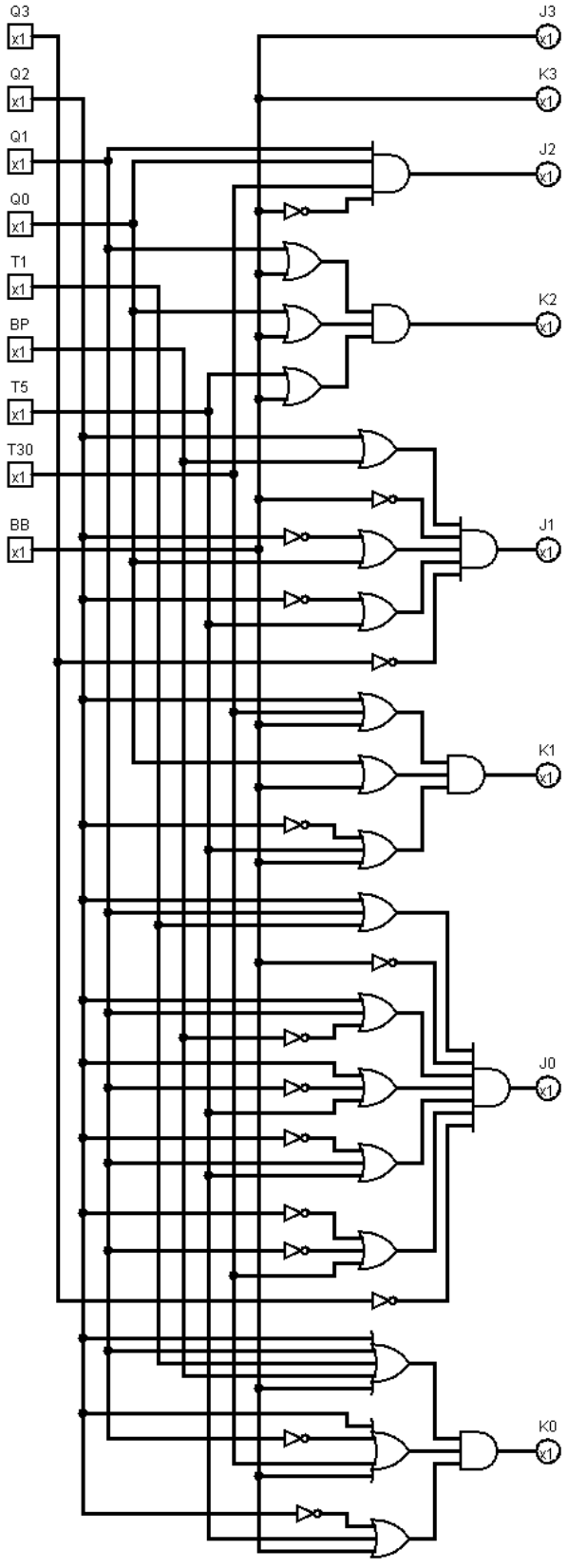
- T1 (timer de 1 segundo)
- BB (botón de los bomberos para resetear el conteo).

Salida:

- T30 (timer de 30 segundos)

Detalle de la lógica de entrada:

Este circuito es el que decodifica los estados, para que obtener la secuencia deseada.



Entradas:

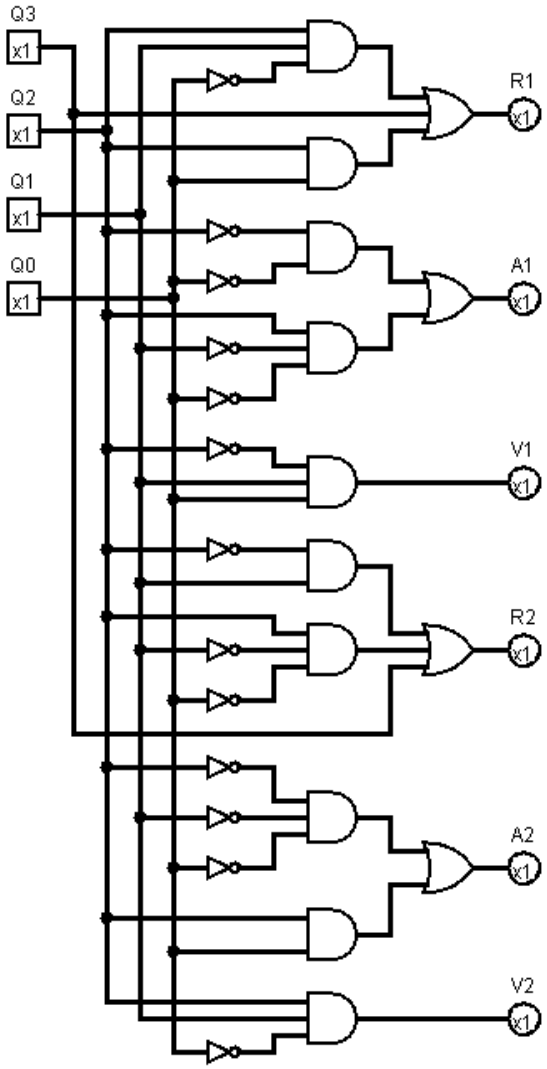
- Q3 (salida del FF JK 3)
- Q2 (salida del FF JK 2)
- Q1 (salida del FF JK 1)
- Q0 (salida del FF JK 0)
- T1 (timer de 1 segundo)
- BP (botón de peatones)
- T5 (timer de 5 segundos)
- T30 (timer de 30 segundos)
- BB (botón de bomberos)

Salidas:

- J3 y K3 (entradas del FF JK3)
- J2 y K2 (entradas del FF JK2)
- J1 y K1 (entradas del FF JK1)
- J0 y K0 (entradas del FF JK0)

Detalle lógica de salida:

Este circuito, es el que a partir de las 4 salidas de los Flip-Flops nos permite obtener las 6 funciones salidas de nuestro circuito que nosotros deseamos implementar.



Entradas:

- Q3 (salida del FF JK 3)
- Q2 (salida del FF JK 2)
- Q1 (salida del FF JK 1)
- Q0 (salida del FF JK 0).

Salidas:

- R1, A1 y V1 (luces del primer semáforo)
- R2, A2, y V2 (luces del segundo semáforo).

Análisis de Costos para la construcción de un prototipo

A continuación, se detalla un presupuesto para la construcción de 1 prototipo de la solución cableada de la lógica del sistema de semáforos.

Se deja constancia que el mismo no incluye la implementación del correspondiente circuito de potencia, para poder manejar luces de semáforo, como tampoco la fuente de alimentación que el mismo requiere.

- El diseño del montaje superficial del circuito a implementar, sobre software tipo KiCAD o similar tiene un costo de $500.
- La construcción de las placas sobre la cuales deben montarse los componentes, sobre dos placas epoxi doble faz de 10cm x 20cm tiene un costo de $150 por placa, $ 300 en total. Son placas vírgenes y el presente importe incluye:
  - 2 placas experimentales epoxi doble faz de 10cm x 20cm vírgenes
  - Percloruro férrico (cantidad necesaria)
  - Impresión sobre transparencia del circuito
  - Traspaso del diseño de las pistas del circuito a las placas vírgenes
  - Ataque del cobre no deseado con el ácido correspondiente
  - Taladrado de las placas
- El montaje de los componentes tiene un costo de $300. Incluye mano de obra, estaño y cable wire wrapping de ser necesario para la construcción de puentes auxiliares.

Circuitos Necesarios para construir las entradas de los FF:

	Cantidad de Entradas	Cantidad Necesaria	Componente Detalle	Componente Número	Cantidad de integrados	Precio U$S (No incluye IVA)	Observaciones
NOT		15	Hex-Inverter	74HCT04	3	0,75	Sobran 3 NOT
OR		17					
	2	7	Quad OR – 2 input	74HCT32	2	0,45	Sobra 1 OR
	3	8	Triple OR – 3 input	74HCT4075	3	0,40	Sobra 1 OR
	4	1	Dual OR – 4 input	HEF4072B	1	0,60	Sobra 1 OR
	5	1					Lo construyo con uno de 4 input y uno de 1 input que me sobraron arriba 1 OR 4 input + 1 OR 2 input
AND		6					
	3	3	Triple AND - 3 input	74HCT11	1	0,40	-----
	4	1	Dual AND – 4 input	HEF4082B	1	0,60	Sobra 1 AND
	5	1	Dual AND – 4 input	HEF4082B	1	0,60	Lo construyo 2 AND de 4 input. Las dos entradas sobrantes las conecto a 1 lógico.
	7	1	Dual AND – 4 input	HEF4082B	1	0,60	------

Total: U$s 4,95

Total en pesos con IVA incluido: $29

Circuitos Necesarios para construir las SALIDAS de los FF:

	Cantidad de Entradas	Cantidad Necesaria	Componente Detalle	Componente Número	Cantidad de integrados	Precio U$S (No incluye IVA)	Observaciones
NOT		13	Hex-Inverter	74HCT04	3	0,75	Sobran 5 NOT
AND		10					
	2	4	Quad AND – 2 input	74HCT08	1	0,25	---
	3	6	Triple AND – 3 input	74HCT11	2	0,40	---
OR		4					
	2	2	Quad OR – 2 input	74HCT32	1	0,45	Sobran 2 OR
	3	2	Triple OR – 3 input	74HCT4075	1	0,40	Sobra 1 OR

Total: u$s 4,15

Total en pesos con IVA incluido: $24

Componentes necesarios para los temporizadores:

	Cantidad Necesaria	Detalle del componente	Número de Componente	Cantidad de integrados	Precio U$S (No incluye IVA)	Observaciones
Cristal	1				1,10	
FF-JK	5		74HCT107	3	0,65	A partir de estos se construyen los FF-T para dividir la frecuencia.

Total: u$s 3,05

Total en pesos con IVA incluido: $ 18

Adicionalmente necesitamos 4 FF-JK (es decir, dos integrados 74HCT107) adicionales para la implementación del circuito lógico.

Total: u$s 1,30

Total en pesos con IVA incluido: $ 8

Pack de Resistencias para utilizarlas como PULL-UP, PULL-DOWN, e inicialización de los FF: $ 18

Pack de Capacitores para inicialización de los FF: $ 12

Precio total de la implementación del prototipo: $1209

Aclaración:

- Es importante destacar, que el costo del diseño del montaje superficial del circuito a implementar, sobre software tipo KiCAD o similar (que es el mayor de todos los costos), sólo debe realizarse una vez, motivo por el cual, si quisiera construirse en serie, este costo debería dividirse entre la cantidad de circuitos a construir, por lo que el precio de construcción en serie del mismo sería mucho menor.

Código ARC de la solución por software (documentado)

```
.begin
.org 2048

!Almaceno datos en memoria, que posteriormente voy a utilizar como mascaras para correr instrucciones internas auxiliares
mascara_BB: 2 !bit 1
mascara_BP: 1 !bit 0
mascara_V1: 65536
mascara_A1: 131072
mascara_R1: 262144
mascara_V2: 16777216
mascara_A2: 33554432
mascara_R2: 67108864


botones_luces .equ 0xFFC !Aca va la direccion a donde estan mapeados los botones, y las luces (la del enunciado del TP era demasiado alta, tuvimos que cambiarla a una mas baja). Idem 4092.


!%r1 Contiene los Botones y luces de salida (los volvemos a leer de memoria cada vez que controlamos el boton de bomberos o el del peaton
!%r3 Lo utilizamos como registro auxiliar
!%r4 contiene el numero de estado actual, siendo A = 1, B = 2, ..., I = 9


and %r0,%r0,%r4 ! r4 <-- 0

!Comienza el programa en el estado A
ba estado_A



estado_A:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria

    ld [mascara_V1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V1
    ld [mascara_A1],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en A1
    ld [mascara_R1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en R1
    ld [mascara_V2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V2
    ld [mascara_A2],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en A2
    ld [mascara_R2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en R2

    st %r1, [botones_luces] ! ACTUALIZO LAS LUCES DEL SEMAFORO EN MEMORIA
    add %r0,1,%r4 !Actualizo el numero de estado

    call contar_tiempo !Hace pasar un segundo
    call check_boton_BB !Controla si presionaron el botón BB (bomberos), y si corresponde salta a la secuencia de bomberos
    call check_boton_BP !Controla si presionaron el botón BP (peatón), y si corresponde empieza con la rutina para que cruce el peatón
    ba estado_B !Si no presionaron BB ni BP, pasa al estado B y continúan los semaforos intermitentes
```

```
estado_B:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria

    ld [mascara_V1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V1
    ld [mascara_A1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en A1
    ld [mascara_R1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en R1
    ld [mascara_V2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V2
    ld [mascara_A2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en A2
    ld [mascara_R2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en R2

    st %r1, [botones_luces] ! ACTUALIZO LAS LUCES DEL SEMAFORO EN MEMORIA
    add %r0,2,%r4 !Actualizo el numero de estado

    call contar_tiempo !Hace pasar un segundo
    call check_boton_BB !Controla si presionaron el botón BB (bomberos), y si corresponde salta a la secuencia de
bomberos
    call check_boton_BP !Controla si presionaron el botón BP (peatón), y si corresponde empieza con la rutina para que
cruce el peatón
    ba estado_A !Si no presionaron BB ni BP, pasa al estado A y continúan los semaforos intermitentes



estado_C:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria

    ld [mascara_BP],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en el botón BP, ya que ya lo leí y entré en la secuencia

    ld [mascara_V1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V1
    ld [mascara_A1],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en A1
    ld [mascara_R1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en R1
    ld [mascara_V2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V2
    ld [mascara_A2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en A2
    ld [mascara_R2],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en R2

    st %r1, [botones_luces] ! ACTUALIZO LAS LUCES DEL SEMAFORO y suelto el botón BP EN MEMORIA
    add %r0,3,%r4!Actualizo el numero de estado
```

```
    andcc %r0, %r0, %r27

    seguir_en_C:
        call contar_tiempo !Hace pasar un segundo
        add %r27, 1, %r27 !sumo cada segundo que transcurre, para saber cuando tengo que cambiar de estado
        call check_boton_BB !verifico que no hayan presionado el boton de bomberos, sino debo pasar al estado I
        addcc %r27, -5, %r0 !cuando la suma llega a 5 significa que pasaron 5 segundos
        be estado_D !paso al estado siguiente si ya pasaron los 30 segundos
        ba seguir_en_C !si no pasaron 5 segundos, sigo en el mismo estado contando hasta que pasen 5 segundos


estado_D:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria

    ld [mascara_R1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en R1
    ld [mascara_A1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en A1
    ld [mascara_V1],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en V1
    ld [mascara_R2],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en R2
    ld [mascara_A2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en A2
    ld [mascara_V2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V2

    st %r1, [botones_luces] !  ACTUALIZO LAS LUCES DEL SEMAFORO
    add %r0,4,%r4!Actualizo el numero de estado

    andcc %r0, %r0, %r27

    seguir_en_D:
        call contar_tiempo !Hace pasar un segundo
        addcc %r27, 1, %r27 !sumo cada segundo que transcurre, para saber cuando tengo que cambiar de estado
        call check_boton_BB !verifico que no hayan presionado el boton de bomberos, sino debo pasar al estado I
        addcc %r27, -30, %r0  !cuando la suma llega a 30 significa que pasaron 30 segundos
        be estado_E !paso al estado siguiente si ya pasaron los 30 segundos
        ba seguir_en_D !si no pasaron 30 segundos, sigo en el mismo estado contando hasta que pasen 30 segundos


estado_E:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria

    ld [mascara_V1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V1
    ld [mascara_A1],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en A1
    ld [mascara_R1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en R1
    ld [mascara_R2],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en R2
    ld [mascara_A2],%r3
    xnor %r3,%r0,%r3 !Not r3
```

```
    and %r3,%r1,%r1 !Fuerza un 0 en A2
    ld [mascara_V2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V2

    st %r1, [botones_luces] !  ACTUALIZO LAS LUCES DEL SEMAFORO
    add %r0,5,%r4!Actualizo el numero de estado

    andcc %r0, %r0, %r27

    seguir_en_E:
        call contar_tiempo !Hace pasar un segundo
        addcc %r27, 1, %r27  !sumo cada segundo que transcurre, para saber cuando tengo que cambiar de estado
        call check_boton_BB !verifico que no hayan presionado el boton de bomberos, sino debo pasar al estado I
        addcc %r27, -5, %r0  !cuando la suma llega a 5 significa que pasaron 5 segundos
        be estado_F !paso al estado siguiente si ya pasaron los 5 segundos
        ba seguir_en_E  !si no pasaron 5 segundos, sigo en el mismo estado contando hasta que pasen 5 segundos


estado_F:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria

    ld [mascara_R1],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en R1
    ld [mascara_A1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en A1
    ld [mascara_V1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V1
    ld [mascara_V2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V2
    ld [mascara_A2],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en A2
    ld [mascara_R2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en R2

    st %r1, [botones_luces] ! ACTUALIZO LAS LUCES DEL SEMAFORO
    add %r0,6,%r4!Actualizo el numero de estado

    andcc %r0, %r0, %r27

    seguir_en_F:
        call contar_tiempo !Hace pasar un segundo
        addcc %r27, 1, %r27 !sumo cada segundo que transcurre, para saber cuando tengo que cambiar de estado
        call check_boton_BB !verifico que no hayan presionado el boton de bomberos, sino debo pasar al estado I
        addcc %r27, -5, %r0 !cuando la suma llega a 5 significa que pasaron 5 segundos
        be estado_G !paso al estado siguiente si ya pasaron los 5 segundos
        ba seguir_en_F !si no pasaron 5 segundos, sigo en el mismo estado contando hasta que pasen 5 segundos


estado_G:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria

    ld [mascara_R1],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en R1
    ld [mascara_A1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en A1
```

```
    ld [mascara_V1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V1
    ld [mascara_R2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en R2
    ld [mascara_A2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en A2
    ld [mascara_V2],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en V2

    st %r1, [botones_luces] !  ACTUALIZO LAS LUCES DEL SEMAFORO
    add %r0,7,%r4!Actualizo el numero de estado

    andcc %r0, %r0, %r27

    seguir_en_G:
        call contar_tiempo !Hace pasar un segundo
        addcc %r27, 1, %r27  !sumo cada segundo que transcurre, para saber cuando tengo que cambiar de estado
        call check_boton_BB  !verifico que no hayan presionado el boton de bomberos, sino debo pasar al estado I
        addcc %r27, -30, %r0  !cuando la suma llega a 30 significa que pasaron 30 segundos
        be estado_H  !paso al estado siguiente si ya pasaron los 30 segundos
        ba seguir_en_G  !si no pasaron 30 segundos, sigo en el mismo estado contando hasta que pasen 30 segundos


estado_H:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria

    ld [mascara_R1],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en R1
    ld [mascara_A1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en A1
    ld [mascara_V1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V1
    ld [mascara_V2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V2
    ld [mascara_A2],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en A2
    ld [mascara_R2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en R2

    st %r1, [botones_luces] !  ACTUALIZO LAS LUCES DEL SEMAFORO
    add %r0,8,%r4 !Actualizo el numero de estado

    andcc %r0, %r0, %r27

    seguir_en_H:
        call contar_tiempo !Hace pasar un segundo
        addcc %r27, 1, %r27  !sumo cada segundo que transcurre, para saber cuando tengo que cambiar de estado
        call check_boton_BB  !verifico que no hayan presionado el boton de bomberos, sino debo pasar al estado I
        addcc %r27, -5, %r0 !cuando la suma llega a 5 significa que pasaron 5 segundos
        be volver_a_estado_A !paso al estado siguiente si ya pasaron los 5 segundos
        ba seguir_en_H  !si no pasaron 5 segundos, sigo en el mismo estado contando hasta que pasen 5 segundos



volver_a_estado_A:
```

```
    ld [botones_luces],%r1 !leo los botones y las luces de memoria

    ld [mascara_BP],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en el botón BP, hasta que no vuelva al estado inicial, el botón no tiene que tener efecto
    st %r1, [botones_luces] ! Suelto el botón BP EN MEMORIA por si lo presiono durante la secuencia
    and %r0, %r0, %r27 !limpio el registro 27 para no confundir en la simulacion
    ba estado_A !vuelvo al estado A (inicial)



estado_I:

    and %r0, %r0, %r27 !limpio el registro 27 para no confundir en la simulacion
    ld [botones_luces],%r1 !leo los botones y las luces de memoria

    ld [mascara_BB],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en el botón BB, ya que ya lo leí y entré en la secuencia, sino de ahora en más simpre va a
estar entrando y saliendo porque se mantiene el botón presionado

    ld [mascara_R1],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en la posicion de R1
    ld [mascara_A1],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en A1
    ld [mascara_R2],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en R2
    ld [mascara_A2],%r3
    or %r3,%r1,%r1 !Fuerza un 1 en A2
    ld [mascara_V1],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V1
    ld [mascara_V2],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en V2

    st %r1, [botones_luces] ! ACTUALIZO LAS LUCES DEL SEMAFORO y suelto el botón BB EN MEMORIA
    add %r0,9,%r4! Actualizo el numero de estado

    seguir_en_I:
        call contar_tiempo !Hace pasar un segundo
        call check_boton_BB_salida !controla si volvieron a presionar el boton de bomberos, en cuyo caso debo volver al
estado A
        ba seguir_en_I !si no presionaron el botón de bomberos, permanezco en el estado actual



check_boton_BB:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria
    ld [mascara_BB],%r3
    and %r3,%r1,%r3 !Me fijo si esta presionado el boton BB
    addcc %r3, -1, %r0
    bpos estado_I !si está presionado el botón BB, debo saltar al estado I sin importar en que estado estoy
    jmpl %r15+4, %r0 !sino presionaron el botón BB, vuelvo a la subrutina que invocó a esta función



check_boton_BB_salida:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria
    ld [mascara_BB],%r3
    and %r3,%r1,%r3 !Me fijo si esta presionado el boton BB
    addcc %r3, -1, %r0
```

```
    bpos reiniciar_botones_BBBP_y_pasar_a_estado_A  !si está presionado el botón BB, debo salir del estado I y volver al estado A (inicial) porque ya salió el camion de bomberos
    jmpl %r15+4, %r0 !sino presionaron el botón BB, vuelvo a la subrutina que invocó a esta función



reiniciar_botones_BBBP_y_pasar_a_estado_A:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria
    ld [mascara_BB],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en el botón BB, sino de ahora en más simpre va a estar entrando y saliendo porque se mantiene el botón presionado
    ld [mascara_BP],%r3
    xnor %r3,%r0,%r3 !Not r3
    and %r3,%r1,%r1 !Fuerza un 0 en el botón BP, para que cuando vuelva al estado A no ingrese directamente a la secuencia del peaton
    st %r1, [botones_luces] ! suelto el botón BB y BP EN MEMORIA
    ba estado_A !regreso al estado A



check_boton_BP:

    ld [botones_luces],%r1 !leo los botones y las luces de memoria
    ld [mascara_BP],%r3
    and %r3,%r1,%r3 !Me fijo si esta presionado el boton BP
    addcc %r3, -1, %r0
    bpos estado_C !si está presionado el boton BP, debo saltar al estado C
    jmpl %r15+4, %r0


ciclosneg .equ -6 ! Esto determina cuantos ciclos internos del timer representan un segundo

contar_tiempo:
    !r16 cuenta la cantidad de ciclos de reloj hasta llegar a 1 segundo

    and %r0, %r0, %r16 !pongo r16 en 0

    seguir_sumando:
        add %r16, 1, %r16
        addcc %r16, ciclosneg , %r0 ! considero que ciclosneg es la cantidad de ciclos de reloj que representa 1 segundo
        be paso_un_segundo
        ba seguir_sumando !si llega aca es porque todavia no conto CICLOSNEG ciclos, es decir no llego a 1 segundo

    paso_un_segundo:
        jmpl %r15+4, %r0


.end
```

Costo de la solución por software

Mano de obra (programación en código Assembler):

El costo de realizar una solución programada como esta ronda en los $800 (si lo realiza gente con experiencia en programación en lenguaje Assembler), según la cantidad de horas de trabajo.

En nuestro caso particular, la cantidad de horas fue mucho más elevada debido a que tuvimos que aprender un lenguaje que nunca antes habíamos utilizado, y gran parte se ha aprendido a medida que se iba programando. Por eso no podemos fijar un precio por hora de trabajo.

Circuito integrado:

Es difícil determinar el costo de un micro-controlador de 1Ghz y 32 bit, ya que normalmente para este tipo de implementaciones, se utilizan (por ser más comunes y más económicos comercialmente) micro-controladores tipo RISC (de 8 bits) o sino hoy en día son muy comunes los PICs, que suelen ser de 8 bits y la frecuencia ronda entre en los 20 Mhz y los 50 Mhz. Si utilizáramos uno de estos últimos (debería modificarse el código ya que utilizamos, por ejemplo, palabras de 4 bytes, entre otras cuestiones) el costo del integrado ronda en los u$s 11 (once dólares), lo que equivale a $50 (cincuenta pesos argentinos).

Costo de grabar el código programado en el integrado:

Llevar a un local especializado el código para que lo graben en el micro-controlador, cuesta $80, suponiendo que nosotros no disponemos de un programador de integrados como el que necesitamos.

Producción en serie:

Como puede verse, el costo de programación se abona una sola vez, por lo que si se desean producir en serie este sistema, el costo de programación antes mencionado se divide entre la cantidad de sistemas a producir, pudiéndose volver muchísimo menor.

Además se cuenta con la ventaja de poder producir este sistema en muy poco tiempo, ya que lo único que debería hacerse es programar una y otra vez el código en integrados diferentes.

Comparación de las dos soluciones obtenidas:

Ventajas de la solución por software:

- En el sistema funcionando, la solución de fallas consiste en reemplazar el micro-controlador programado por uno nuevo, sin necesidad de detenerse a buscar cuál es el componente que está fallando, por lo que es más rápido.
- El costo total del sistema resulta ser menor.
- El tiempo y trabajo para la producción en serie es menor, ya que solo deben programarse nuevos integrados.
- El espacio físico ocupado por la solución por software es el tamaño de un integrado, frente a varias placas que requiere la solución cableada
- Si se desea modificar alguna de las condiciones del sistema, por ejemplo el tiempo de espera de un estado pasarlo de 30 segundos a 60, si el código está apropiadamente documentado, el esfuerzo que requiere es mínimo, y pueden volver a programarse los micro-controladores que ya disponemos, reutilizando los sistemas ya implementados, por lo que el costo es menor.

Desventajas de la solución por software:

- Dificultad y costo de conseguir micro-controladores RISC de 1 Ghz y 32 bits.

Ventajas de la solución cableada:

- Ante una falla, si puede determinarse a priori cuál es el componente que está fallando, el costo de reemplazarlo suele ser prácticamente nulo (la mayoría de los componentes cuestan menos de $5)

Desventajas de la solución cableada:

- Detectar una falla en un sistema que cuenta con dos o tres placas resulta ser un trabajo tedioso y complicado, ya que la falla puede estar en cualquiera de los integrados (flip-flops, compuestas, otros) como también en resistencias, capacitores o incluso en la misma placa. Motivo por el cual ante una falla, conviene reemplazar todo el sistema debido al tiempo que insumiría detectar la falla, por lo que también el costo es mayor.
- El espacio físico ocupado es mucho mayor que el de la solución programada, ya que son dos placas de 10 x 20.
- A la hora de producción en serie, se requiere muchas más horas hombre construyendo los PCB, o soldando los componentes, con los consecuentes errores humanos que estos pueden conllevar (por ejemplo, soldar un componente en el lugar equivocado, o una construcción defectuosa del PCB, lo que implica que dicho sistema sea más propenso a una falla, o el costo de reemplazarlo)
- Si se desea modificar alguna de las condiciones del sistema, por ejemplo el tiempo de espera de un estado pasarlo de 30 segundos a 60, debe reemplazarse todo el sistema, ya que deben agregarse o quitarse circuitos y cambiar conexiones de lugar, lo que implica reemplazar toda la placa. Además muchos de los componentes de estas no son reutilizables.