Síntesis programada

```
! MEMORIA
! [ENTRADA]: (MSB)...RESET,HAY_AGUA,N_MAX,N_MIN (LSB)
! [SALIDA]: (MSB)...MOTOR,LUZ_ROJA,LUZ_AMARILLA (LSB)

! REGISTROS
! %r1 = Entrada.
! %r2 = Auxiliar para la entrada.
! %r3 = Salida.
! %r4 = Auxiliar para la salida.
! %r5 = Variable para el contador.
! %r6 = Constante de 8_MIN.
! %r7 = Posición de memoria de la entrada.
! %r8 = Posición de memoria de la salida.


                    .begin

                    .macro save_32 valor,registro
                          sethi  %hi(valor),registro
                          or     registro,%lo(valor),registro
                    .endmacro

                    .org 2048

ENTRADA             .equ 0x05000000
SALIDA              .equ 0x06000000
8_MIN               .equ 0x4F790D55
! Micro de 50 MHz, se realizan 8 operaciones y 1 acceso a memoria (18 ciclos).
! 1 operación se hace en 1/50 * 10-6 segundos.
! Un ciclo del contador, en 18/50 * 10-6 segundos.
! Entonces, 8 min = 480 segundos equivalen a (480*50 / 18) * 10⁶ ciclos

                    ba     INICIALIZACION

LECTURA:            ld     %r7,%r1             ! Guardo en %r1 lo que leo de la
entrada.
                    andcc  %r1,8,%r2           ! %r2 <- (%r1 AND 8), enmascaro el bit
del botón RESET. Si da 1, es porque está apretado.
                    be     COMPARO_N_MIN       ! Si no está apretado el RESET, sigo
viendo los demás bits de la entrada.
                    ba     RESET               ! Si está apretado, reseteo.

COMPARO_N_MIN:      andcc  %r1,1,%r2           ! Enmascaro el bit del nivel mínimo.
```

```
Si da 1, es porque el agua se encuentra por debajo del nivel mínimo.
                    be     COMPARO_N_MAX       ! Si el agua no está por debajo del
nivel mínimo, comparo el nivel máximo,
                    ba     N_MIN               ! sino, actúo de acuerdo a este evento.

COMPARO_N_MAX:      andcc  %r1,2,%r2           ! Enmascaro el bit del nivel máximo.
Si da 1, es porque se superó el nivel máximo.
                    bne    APAGO_MOTOR         ! Se superó el nivel máximo, entonces
apago el motor.
                    ba     LECTURA             ! Si no se superó, vuelvo a leer
esperando cambios.

RESET:              st     %r0,%r8             ! Apago el motor y todas las luces.
                    ba     LECTURA             ! Vuelvo a comenzar.

N_MIN:              andcc  %r1,4,%r2           ! Enmascaro el bit de HAY_AGUA. Si da
1, es porque no se pasó el límite inferior del agua en el reservorio.
                    be     PRENDO_LUZ_ROJA     ! Si no hay agua, prendo la luz roja.
                    ba     PRENDO_MOTOR        ! Si hay agua, y se alcanzó el nivel
mínimo, prendo el motor.

PRENDO_MOTOR:       andcc  %r3,4,%r4           ! Enmascaro el bit correspondiente al
motor.
                    bne    CONTADOR            ! Si ya estaba prendido, salto al
contador para evitar reiniciar %r5.
                    or     %r0,4,%r3           ! Sino, armo %r3,
                    st     %r3,%r8             ! y luego lo mando a la salida,
prendiendo el motor.
                    and    %r5,%r0,%r5         ! Inicializo el registro que voy a
utilizar como contador.
CONTADOR:           subcc  %r6,%r5,%r0         ! Comparo el estado del contador, con
el número equivalente a 8 minutos.
                    be     PRENDO_LUZ_AMARILLA ! Si se cumplieron los 8 minutos,
prendo la luz amarilla.
                    ld     %r7,%r1             ! Sino, cargo la entrada,
                    andcc  %r1,8,%r2           ! y enmascaro el bit de RESET.
                    bne    RESET               ! Si se apretó el RESET, reseteo.
                    andcc  %r1,2,%r2           ! Sino, enmascaro el bit del nivel
máximo.
                    bne    APAGO_MOTOR         ! Si se alcanzó el nivel máximo, apago
el motor.
                    add    %r5,1,%r5           ! Si no pasó ninguna de esas cosas,
incremento el contador,
                    ba     CONTADOR            ! y vuelvo a comenzar el ciclo.

APAGO_MOTOR:        st     %r0,%r8             ! Apago el motor.
                    ba     LECTURA
```

```
PRENDO_LUZ_ROJA:    add    %r0,2,%r3          ! Prendo la luz roja.
                    st     %r3,%r8            ! Y luego lo mando por la salida.
                    ba     LECTURA

PRENDO_LUZ_AMARILLA:      add    %r0,1,%r3          ! Prendo la luz amarilla (y
apago el motor).
                    st     %r3,%r8            ! Y luego lo mando por la salida.
ESPERO_RESET:       ld     %r7,%r1            ! Cargo la entrada,
                    andcc  %r1,8,%r2          ! y enmascaro el bit de RESET.
                    be     RESET              ! Si se apretó, reseteo todo.
                    ba     ESPERO_RESET       ! Sino, espero. Ya que de esta
situación sólo se sale al apretar RESET.

INICIALIZACION:     and    %r0,%r1,%r1        ! Inicializo en 0 el registro %r1.
                    and    %r0,%r2,%r2        ! Inicializo en 0 el registro %r2.
                    and    %r0,%r3,%r3        ! Inicializo en 0 el registro %r3.
                    and    %r0,%r4,%r4        ! Inicializo en 0 el registro %r4.
                    and    %r0,%r5,%r5        ! Inicializo en 0 el registro %r5.
                    save_32 8_MIN, %r6        ! Inicializo %r6 con la constante
8_MIN.
                    save_32 ENTRADA, %r7            ! Inicializo %r7 con la posición
de memoria de la entrada.
                    save_32 SALIDA, %r8       ! Inicializo %r8 con la posición de
memoria de la salida.
                    ba     LECTURA

                    .end
```