UNIVERSIDAD DE GRANADA.

ESCUELA TECNICA SUPERIOR DE INGENIERIAS INFORMATICA Y DE TELECOMUNICACIÓN.

Departamento de Ingeniería de Computadores, Automática y Robótica.

TECNOLOGÍA Y ORGANIZACIÓN DE COMPUTADORES.

PRÁCTICA 7.

DESCRIPCIÓN A NIVEL RT DE UN COMPUTADOR SENCILLO.

1º GRADO EN INGENIERÍA INFORMÁTICA.

PRÁCTICA 7. INSTRUCCIONES.

DESCRIPCIÓN A NIVEL RT DE UN COMPUTADOR SENCILLO.

Objetivos:

- *Experimentar con el computador sencillo CS1_LogicWorks_versión 1.b en LogicWorks. El diseño de este computador es descrito teóricamente en el Tema 5.*
- *En esta práctica, se muestra cómo realizar e introducir en CS1_LogicWorks un programa, ejecutarlo y analizar su comportamiento. También, sus posibilidades para realizar seguimientos de las microoperaciones que tienen lugar durante la ejecución un programa a nivel de cada ciclo máquina.*

Material necesario para el desarrollo de la práctica:

- *Guión de prácticas disponible en SWAD en el apartado ARCHIVOS>DOCUMENTOS>04.-PRACTICAS>PRACTICA_7 >PRACTICA_7_TOC-INSTRUCCIONES.PDF y documento de apoyo disponible en SWAD en el apartado ARCHIVOS>DOCUMENTOS>04.-PRACTICAS>PRACTICA_7>PRACTICA_7_TOC_EJEMPLO.PDF.*
- *Material del Tema 5º disponible en SWAD en el apartado ARCHIVOS>DOCUMENTOS>01.-TEORIA y PROBLEMAS>TEMA_5> 0.TEMA_5_TOC_SISTEMAS RT.PDF. Apartado 5.4. Ejemplo de un Computador Sencillo a nivel RT.*
- *Seminario 5. Guía de Trabajo Autónomo. PARTE 1: INTRODUCCIÓN AL MANEJO DE UN SIMULADOR LÓGICO, páginas 1-3 a 1-10 (ambas inclusive) disponible en SWAD en el apartado ARCHIVOS>DOCUMENTOS>02.-SEMINARIOS>SEMINARIO_5>05.-SEMINARIO_5_TOC_SIMULADOR_ENTRENADOR_LOGICO_GUIA.*
- *Software Simulador Lógico LogicWorks.*
- *Referencia Bibliográfica*

[DIA09] Díaz Ruiz, S., Romero Ternero, M. C., Molina Cantero. A. J.. *Estructura y Tecnología de Computadores. Teoría y problemas*. McGraw-Hill, 2009. Capítulos 1 y 2 (apartado 2.2).

7.1 Introducción.

En esta sección se describe la implementación y organización de u Computador Sencillo (en adelante CS1) en LogicWorks, mostrando sus diferentes "vistas" para seguir la ejecución de un programa. Primeramente, hay que descargar el fichero ***CS1_VersiónEstudiante_LW4_Vb.zip*** desde el directorio donde está este guión de prácticas en la plataforma SWAD. El fichero está comprimido. Hay que descomprimirlo para obtener el fichero ***CS1_VersiónEstudiante_LW4_Vb.cct***. que es el que acepta el simulador LogicWorks. A partir de ahora se usará la abreviatura **CS1** para referirse al circuito ***CS1_VersiónEstudiante_LW4_Vb.cct***.

Una vez abierto CS1 en el simulador, se observan diferentes vistas de CS1. La ***primera vista*** es general (ver Figura 7.1), en donde se puede observar la organización global de la unidad de procesamiento, UP, con visualizadores que muestran el contenido de los registros en hexadecimal, tales como: contador de programa PC, registro de direcciones MAR, registro de instrucciones IR, registro temporal RT y acumulador AC.

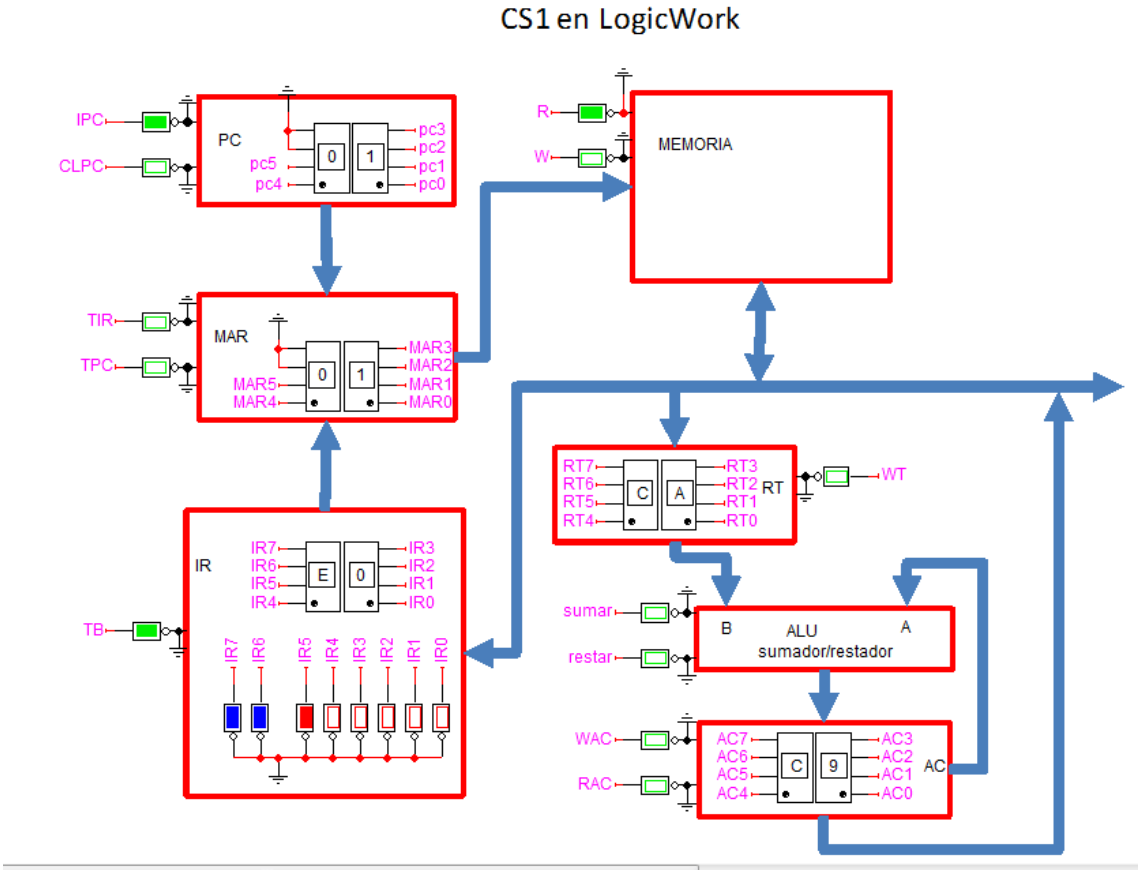

Figura 7.1

También se visualizan con indicadores tipo led las señales de control de cada componente de la unidad de procesamiento. Estas señales las activa la unidad de control. Durante la ejecución de un programa. Más adelante se verá cómo cambian los contenidos de los registros y las señales de control, en cada ciclo de reloj.

La ***segunda vista*** de CS1 (ver Figura 7.2(a)), es la organización real y detallada de los componentes y buses que conforman la unidad de proceso. Si se selecciona un componente y se hace doble "Clic" con el botón izquierdo del ratón, LogicWorks mostrara la realización de dicho componente. Hay que tener en cuenta, que en la "Versión Estudiante" de CS1, el circuito que implementa realmente cada componente ha sido realizado tomando primitivas de la librería "Simulation Logic.clf" del LogicWorks, y no es objetivo de análisis por parte del estudiante. Para dicho análisis, se incluye una "imagen" (no es un circuito) de la realización que debería tener el componente, según lo explicado en teoría en los temas dedicados al análisis y diseño de sistemas digitales. Esto se hace para "invitar" al estudiante (si lo desea y como actividad extra) a su realización y posterior comprobación, mediante la sustitución de dicho componente por el diseñado por el estudiante. En la Figura 7.2(b) se presenta un ejemplo del contenido "imagen" del registro temporal RT.

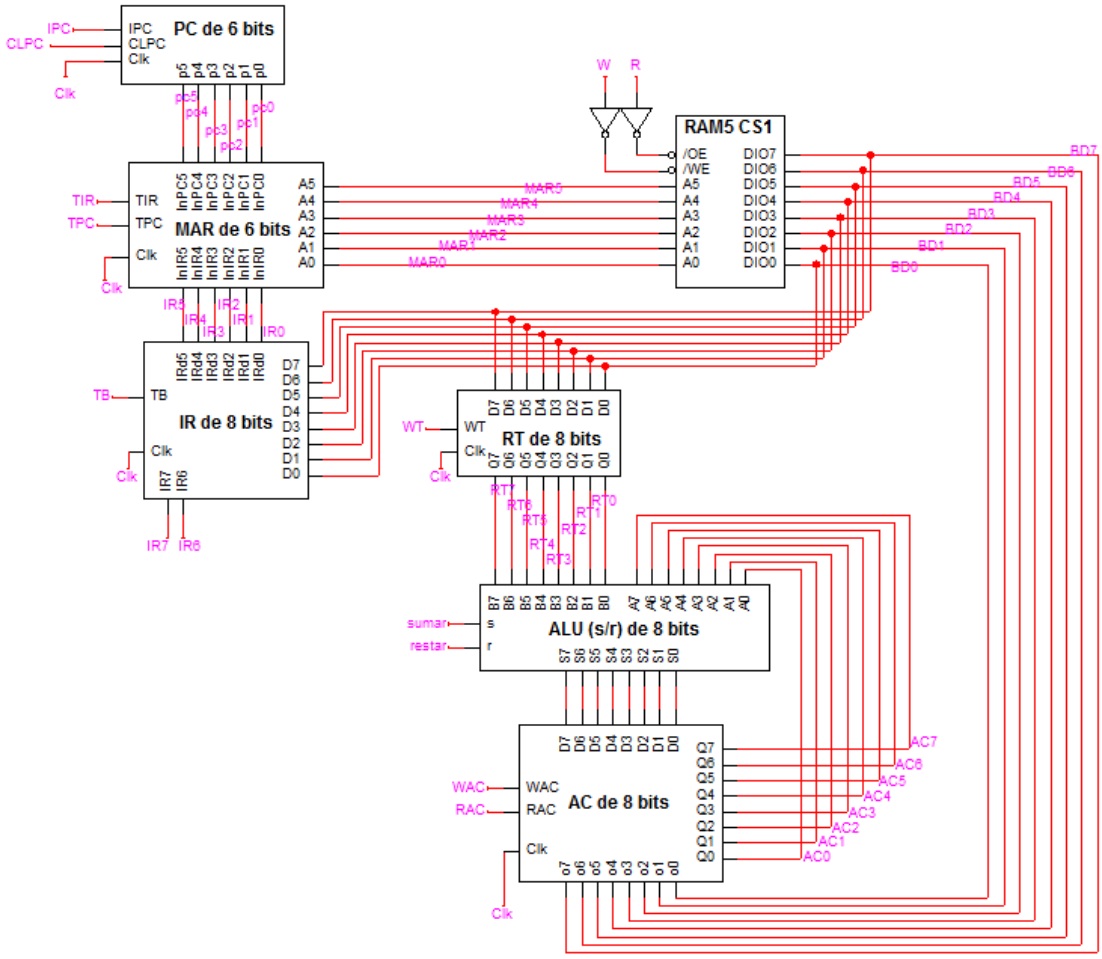

Figura 7.2(a) Unidad de Proceso detallada en LogicWorks.

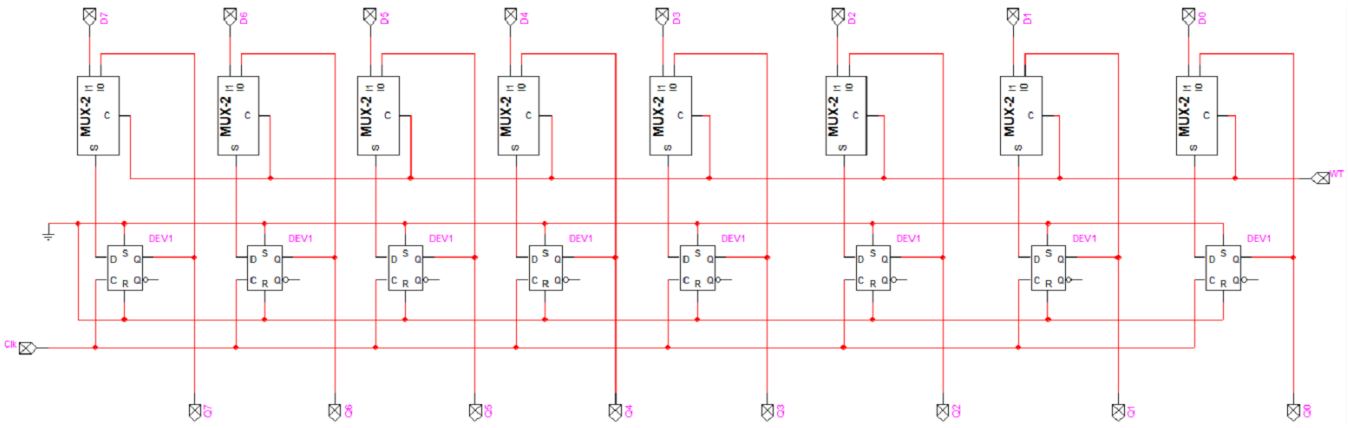

Figura 7.2(b)Vista de la "imagen" que contiene el componente: Registro RT.

Respecto al contenido y edición de la memoria RAM5 CS1, LogicWorks dispone de la siguiente forma de hacerlo. Primero seleccione el módulo RAM5 CS1 con un clic en el botón izquierdo del ratón, después nos vamos al icono que se muestra en la Figura 7.3(a).

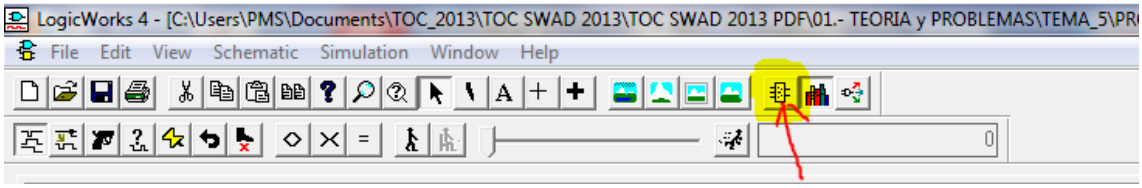

Figura 7.3(a)

Al pulsarlo se obtiene la vista de la Figura 7.3(b). Pulsando en la opción "Siguiente", cuadro de dialogo, como el de la Figura 7.3(c) . Se vuelve a seleccionar "Siguiente", y aparece el cuadro de la Figura 7.3(d), que muestra el contenido en hexadecimal de la memoria.

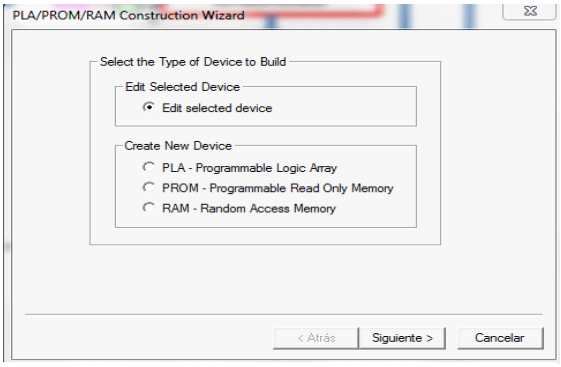

Figura 7.3(b)

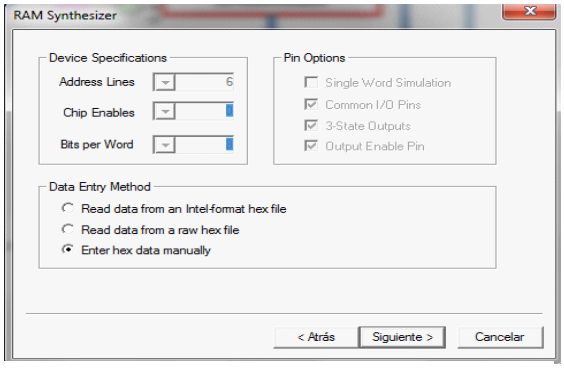

Figura 7.3(c)

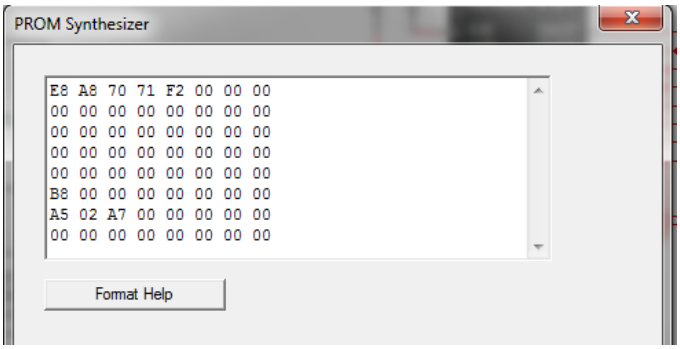

Figura 7.3(d)

Dentro del cuadro de la Figura 7.3(d), se pueden editar los contenidos de la memoria, situándose con el ratón encima del dato que se desea modificar. Las direcciones de memoria son consecutivas, de modo que las dos primeras filas corresponden a direcciones de 00 a 0F, el segundo par de filas, de 10 a 1F, el tercer par de filas, de 20 a 2F, y por último, el cuarto par de filas a direcciones de 30 a 3F. En total, la memoria contiene 64 palabras de 8 bits de longitud. Como ejemplo, se puede ver que el dato B8, que aparece en la Figura 7.3(d), es un dato ubicado en la segunda fila del tercer par y primera columna, por tanto su dirección es la 28 en hexadecimal.

Para facilitar qué posiciones de memoria en hexadecimal corresponden a los contenidos almacenados en la RAM5 CS1 de LogicWorks, se puede utilizar la Figura 7.3(e).

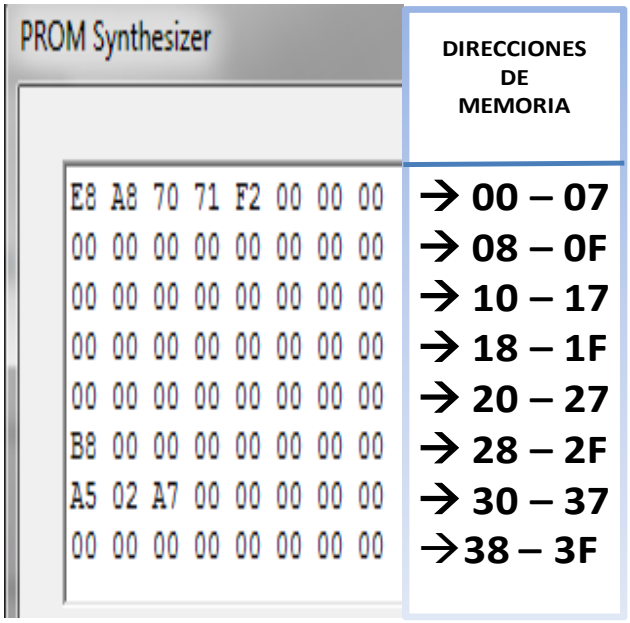

Figura 7.3(e)

La ***tercera vista*** de CS1 (ver Figuras 7.4) se dedica a la unidad de control. **NO ES NECESARIO LEER ESTA PARTE DEL DOCUMENTO Y SE PUEDE PASAR AL APARTADO 7.2**. En la Figura 7.4(a) se ve la realización en LogicWorks de la unidad de control, cuyo diseño se explica en el Tema 5. Junto a esta unidad de control se han añadido diversos elementos (ver Figura 7.4(b)), como visualizadores leds para ver, en cada ciclo de reloj, el estado y bloque ASM en el que se encuentra dicha unidad. También, se proporciona una "imagen" de la Carta ASM de procesado junto con las señales de control que se activan. Mediante visualizadores leds, se pueden ver las señales de control que se activan en cada ciclo de reloj y su efecto en la unidad de procesamiento. Para ello, se adjuntan textos junto con a dichos visualizadores para indicar la/s microoperación/es de transferencia a registros que se van a producir como consecuencia de las señales de control que estén activas en un determinado ciclo de reloj. Se puede, por tanto, realizar un seguimiento minucioso de las fases de captación y ejecución de las instrucciones del programa que se esté ejecutando. Dicho programa tiene que estar almacenado en la memoria RAM5_CS1 a partir de la dirección cero, siendo la última instrucción la de STOP.

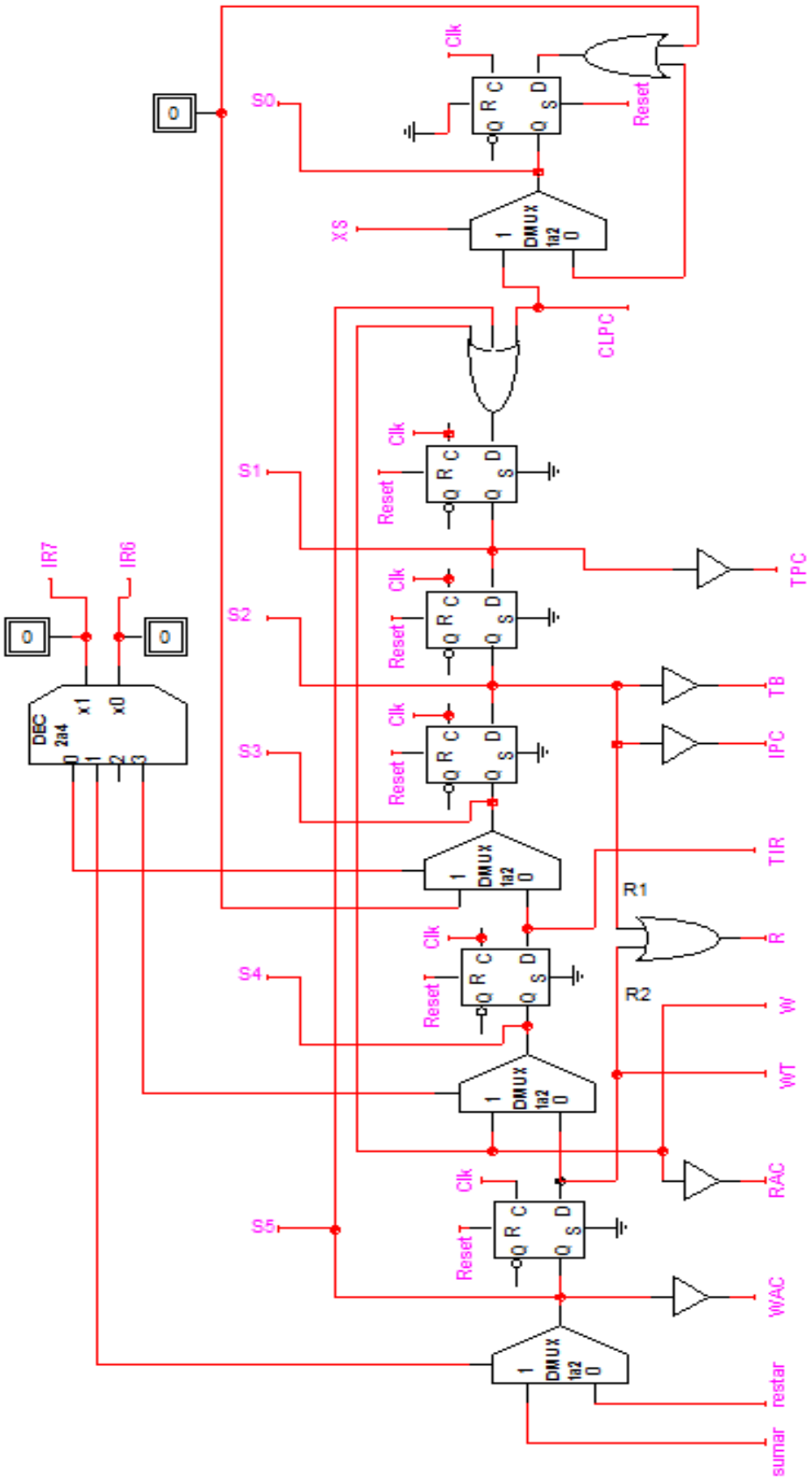

Figura 7.4(a.). Unidad de control de CS1 en LogicWorks 4.1

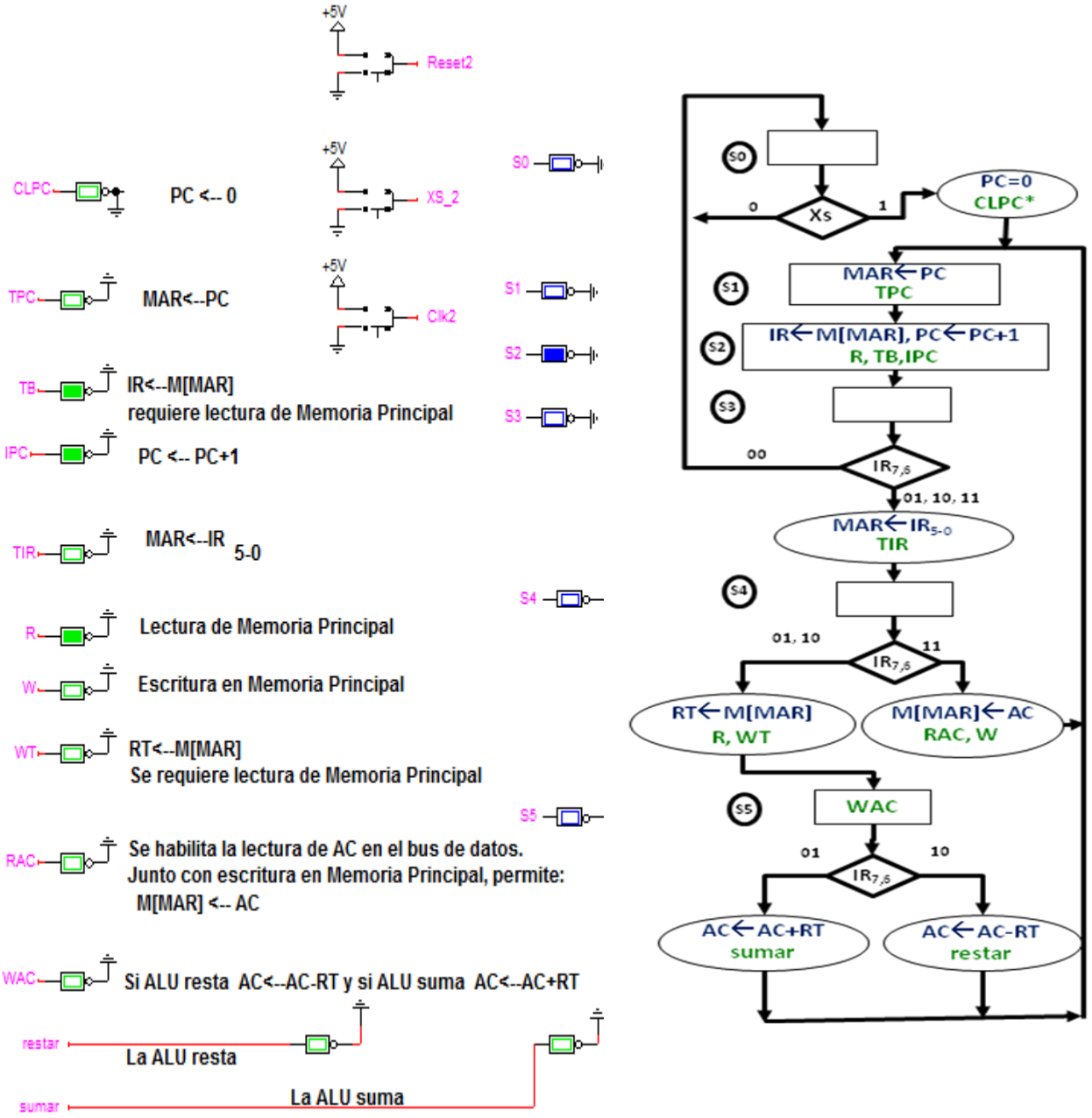

Figura 7.4(b). Unidad de control de CS1 en LogicWorks 4.1

7.2 Cómo realizar un programa y cargarlo en memoria.

Lo primero es definir en ensamblador, con el repertorio de instrucciones de CS1 (ver *Figura 7.4(a)*), el programa que se desea realizar.

Ensamblador ($DirDato en hexadecimal)	**Descripción RT**	**Formato de la instrucción en binario** (Código máquina)	
		C.O.	**Dirección del dato en binario**
STOP	Fin de la ejecución	00	X X X X X X
ADD $DirDato	AC ← AC + M($DirDato)	01	A_5 A_4 A_3 A_2 A_1 A_0
SUB $DirDato	AC ← AC – M($DirDato)	10	A_5 A_4 A_3 A_2 A_1 A_0
STA $DirDato	M($DirDato) ← AC	11	A_5 A_4 A_3 A_2 A_1 A_0

Figura 7.4(a) Repertorio de Instrucciones de CS1

Cada instrucción de CS1 está formada por 8 bits de los cuales, los 2 bits más significativos corresponden al Código de Operación de la Instrucción (C.O.) y los 6 restantes a la dirección de memoria del dato con el que se va a trabajar, a excepción de la instrucción de parada (STOP, C.O. = 00) en que no son relevantes dichos bits.

Por ejemplo, suponga que se desea realizar un programa que llamado **ProgSUMA2** que sume dos datos que están en las direcciones, dadas en hexadecimal, **30 y 31** y que el resultado se almacene en la dirección **32**. Una solución para para el programa **ProgSUMA2** viene dada en la *Figura 7.4(b)*. Obsérvese que las dos primeras instrucciones, sirven para poner el acumulador AC a cero. Para ello, la primera instrucción almacena en la **dirección 28** de memoria el contenido actual del acumulador (**datAux**) y la segunda instrucción resta este valor con el mismo dato que fue almacenado en la **dirección 28**, consiguiendo dejar el acumulador a cero. La tercera instrucción suma el **cero** del acumulador con **dat1** que hay almacenado en memoria en la **dirección 30**, y se carga en el acumulador. La cuarta instrucción suma **dat1** del acumulador con **dat2** que hay almacenado en memoria en la **dirección 31**, por tanto en el acumulador ya se tendría el resultado deseado, **datResul=dat1+dat2**, que se carga en el acumulador. Por último, la quinta instrucción almacena el resultado (**datResul**) en la **dirección** de memoria **32**. La última instrucción (STOP) detiene la ejecución del programa.

Programa en ensamblador ($DirDato en hexadecimal)	**Descripción RT del programa ProgSUMA2**	**Instrucciones en binario**		**Instrucciones en hexadecimal** (Programa en código máquina)
		C.O. 2 bits	**Dirección del dato en binario con 6 bits**	
STA $28	M($28) ← AC	11	10 1000	E8
SUB $28	AC ← AC – M($28)	10	10 1000	A8
ADD $30	AC ← AC + M($30)	01	11 0000	70
ADD $31	AC ← AC + M($31)	01	11 0001	71
STA $32	M($32) ← AC	11	11 0010	F2
STOP	Fin de la ejecución	00	-- ----	00

*Figura 7.4(b) Programa **ProgSUMA2.***

Para cargar el programa en memoria hay que editar la memoria RAM5 CS1, con la utilidad del LogicWorks, descrita en la sección 7.1 (ver Figuras 7.3). El programa en código máquina debe almacenarse en direcciones consecutivas de memoria partiendo de la dirección 00. Por último, hay que introducir en memoria los datos con los que va a operar el programa. En nuestro ejemplo, vamos a suponer que **dat1=A5** (**en la dirección 30**) y **dat2=02** (**en la dirección 31**). La Figura 7.4(c) muestra, cómo deben quedar los contenidos de memoria correspondientes a las direcciones (en hexadecimal): **de la 00 a la 05, donde se almacenan las instrucciones del programa** y en **las direcciones 30 y 31 los datos de entrada dat1=A5 y dat2=02** (no importa el contenido de la memoria en el resto de las direcciones).

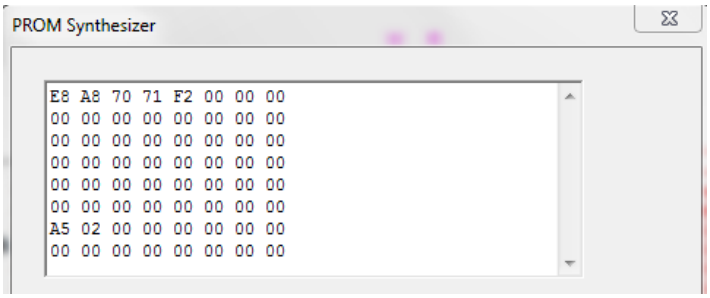

Figura 7.4(c)

Después de ejecutar el programa **ProgSUMA2**, se tendrá que observar el resultado correcto (A5 + 02 = A7), almacenado en la **dirección 32** de la memoria, como el obtenido en la Figura 7.4(d). Se observa también que en la **dirección 28** aparece **B8,** que correspondería a **datAux**, que sería el valor que tenía el acumulador antes de lanzar la ejecución del programa.

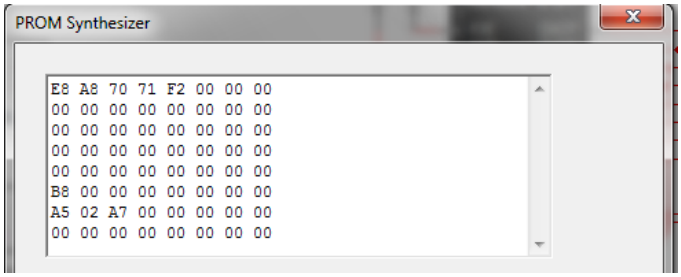

Figura 7.4(d)

7.3 Ejecución del programa y su análisis con CS1 en LogicWorks.

Una vez se tienen en memoria el programa y los datos de entrada, se puede ejecutar. La primera vez que se abre CS1 en LogicWorks, es conveniente activar el pulsador de "Reset1 o Reset2" (ver Figura 7.5), esto lleva a CS1 al estado de **espera S0**. Lo siguiente es definir si se quiere utilizar **modo manual (paso a paso)** o **automático** para generar la señal de reloj. En general, el modo paso a paso interesa para analizar con detalle todos los eventos que se producen en cada ciclo de reloj. El modo automático, está orientado a ejecutar el programa de una manera rápida y también para obtener cronogramas del funcionamiento de CS1. Para elegir entre modo manual paso a paso o modo automático, éste se debe seleccionar mediante el conmutador etiquetado como **Clk Manual con conmutadores Clk1 o Clk2**. Para iniciar la ejecución de un programa, se pulsa **XS_1** o **XS_2**. Este pulso entra en el módulo XS que genera un pulso sincronizado que se envía a CS1 que hace que el sistema, deje de esperar en el estado **S0** y pase a los siguientes estados. Cuando la ejecución del programa termina, CS1 vuelve al **estado de espera S0**, esperando una nueva pulsación en **XS_1** o **XS_2**, para iniciar otra ejecución de un programa.

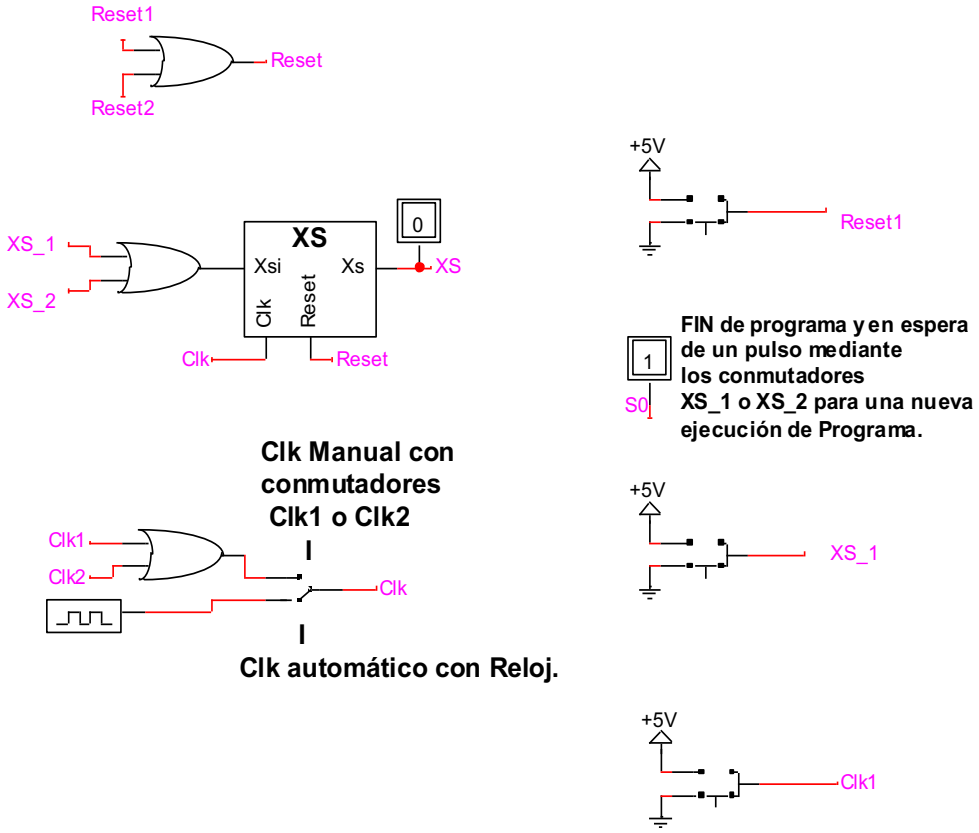

Figura 7.5

El análisis y seguimiento del programa se puede obtener a través de las tres "vistas" de CS1 explicadas en el apartado 7.1, incluyendo la información dada en los cronogramas.

7.4 Realización de la Práctica 7.

7.4.0.- En el computador original CS1 suministrado en SWAD, una vez descomprimido, "CS1_VersiónEstudiante_LW4_Vb.cct", se incluye una memoria RAM5_CS1 que inicialmente almacena ceros en todas las posiciones de memoria. Si se ejecutara CS1, el computador interpreta, como programa almacenado, una sola instrucción con código de operación C.O. = 00, que es la STOP y, por tanto, CS1 Se detendría.

7.4.1.- Edite, como se indica en las Figuras 7.3, la memoria RAM5_CS1 e introduzca en memoria el programa **ProgSUMA2**, descrito en ***el apartado 7.2*** (Figuras 7.4 (b), 7.4(c) y 7.4(d)), con los **datos de entrada dat1=A5 y dat2=02** en **las direcciones M($30) y M($31),** respectivamente**.** Ejecute el programa. Después de esto, compruebe el contenido de la RAM5_CS1, correspondiente a la **dirección 32,** donde se almacena el resultado. ¿Es correcto el resultado obtenido? En el documento

PRACTICA_7_TOC_EJEMPLO.PDF se pueden visualizar las diferentes microoperaciones que componen cada una de las cuatro instrucciones de CS1, así como la secuenciación de las mismas para el programa **ProgSUMA2**.

7.4.2.- Se desea confeccionar una versión del programa ProgSUMA2, que opere con datos de entrada dat1 y dat2 que estén ubicados en direcciones distintas. Concretamente, dat1 en la dirección 38 y dat2 en la dirección 39. Además, se desea que el resultado se almacene en la dirección 3A. Dicho programa se denominará ***ProgSUMA2_vb***.

a) Realice una tabla similar a la indicada en la *Figura 7.4(b)* para el nuevo programa.

b) Edite en la memoria RAM5_CS1, las nuevas instrucciones en hexadecimal, a partir de la dirección 00, así como los **datos de entrada dat1=A5 y dat2=02** en **las direcciones M($38) y M($39),** respectivamente**.**

c) Ejecute el programa ProgSUMA2_vb y compruebe que el resultado almacenado en memoria es correcto comprobando el contenido de la RAM5_CS1, correspondiente a la **dirección 3A,** donde se almacena el resultado.

7.5 Material a desarrollar para la Práctica 7.

El estudiante deberá generar el siguiente material:

7.5.1.- Para el programa **ProgSUMA2.** Fotografías o capturas de pantalla de la memoria RAM5_CS1 tal y como se indica en las Figuras 7.4(c) y 7.4(d) del programa **ProgSUMA2** descrito en el apartado 7.2, antes y después de ser ejecutado (una fotografía o captura de pantalla para cada caso). El programa ha de ser introducido por el estudiante editando la memoria RAM5_CS1 de la forma descrita en las Figuras 7.3. Para la ejecución del programa utilice los **datos de entrada dat1=A5 y dat2=02** en **las posiciones de memoria M($30) y M($31),** respectivamente**.**

7.5.2.- Para el programa **ProgSUMA2_vb** realice:

a) Una tabla similar a la indicada en la *Figura 7.4(b)* para el programa **ProgSUMA2_vb**.

b) Fotografías o capturas de pantalla de la memoria RAM5_CS1 tal y como se indica en las Figuras 7.4(c) y 7.4(d) del programa **ProgSUMA2_vb**, descrito en el apartado 7.4.2, antes y después de ser ejecutado (una fotografía o captura de pantalla para cada caso). El programa ha de ser introducido por el estudiante editando la memoria RAM5_CS1 de

la forma descrita en las Figuras 7.3. Para la ejecución del programa utilice los **datos de entrada dat1=A5 y dat2=02** en **las posiciones de memoria M($38) y M($39),** respectivamente**.**