

Apunte sobre Modelo de Procesador

(borrador inicial del libro Circuitos Lógicos y Principios de Procesadores
a editar próximamente por la Universidad Abierta Interamericana)

Ing. Enrique E. Douce

Introducción

Se desarrolla en este documento el Modelo Didáctico de Procesador incluido en el programa de la materia Sistemas de Computación II de la carrera Ingeniería en Sistemas Informáticos de la Universidad Abierta Interamericana.

Este modelo tiene capacidad de ejecutar todo tipo de instrucciones en lenguaje máquina e integra todos los contenidos de compuertas lógicas de la materia en un circuito que incluye gran parte de la funcionalidad de un procesador real.

- Permite analizar paso a paso la ejecución de distintas instrucciones, desde su búsqueda en la memoria hasta concretar su acción específica y comprender así, detalladamente, la interacción entre el Software y el Hardware.
- Puede ejecutar instrucciones en distintos modos de direccionamiento (Directo, Inmediato, Registro, Indexado y Relativo), explicando los distintos procedimientos asociados.
- Facilita el abordaje a la programación en Assembler al describir precisamente el funcionamiento interno de los procesadores.
- Mejora la comprensión de eficiencia en programación en lenguajes de alto nivel, al demostrar la necesidad de utilizar varios ciclos de reloj, para la concreción de una particular tarea.

Este modelo no solo sirve para explicar la organización interna de los procesadores, sino que aporta adicionalmente el aprendizaje que permite al alumno entender, que todo sistema complejo, puede abordarse, a partir de la comprensión de sus elementos constitutivos, que a su vez pueden subdividirse en otros elementos más sencillos para ser comprendidos.

Este aprendizaje, es fundamental para comprender todo sistema informático, que en definitiva está constituido por una enorme cantidad de elementos sencillos, relacionados entre sí mediante numerosas estructuras, que al final le permiten desarrollar funciones extremadamente complejas.

Propuesta didáctica

Este libro describe inicialmente, todos los elementos necesarios para comprender el Modelo de Procesador.

Se realiza primero un repaso de las compuertas lógicas utilizadas

Se describen luego, circuitos que se desarrollan con esas compuertas, que permiten construir los distintos módulos del modelo. Se repasa el funcionamiento de Decodificadores, Multiplexores y la Unidad Aritmético Lógica con la generación de sus flags de Signo, Zero, Carry y Overflow.

En tercer lugar, se plantea la función del Clock del sistema.

Es necesario luego, analizar el funcionamiento de los elementos de almacenamiento transitorio, es decir los registros, contruidos a partir de Flip Flops Maestro Esclavo.

El repaso de temas previos concluye con la descripción del Modelo, la interacción entre sus áreas funcionales y la particular interacción entre código de operación de la instrucción y la generación de secuencias de unos y ceros, por las líneas de control.

Concluida esta descripción del Hardware, se describe el formato de las instrucciones y sus modos de direccionamiento. Se presenta un pequeño programa que permitirá ejercitar instrucciones que pertenecen a distintos Modos de Direccionamiento.

Luego se presentan Ilustraciones, que representan los movimientos de información que ocurren durante cada ciclo de ejecución de cada instrucción del programa.

La ejecución de cada instrucción incluye un total de tres o cuatro ilustraciones. Dos ilustraciones para describir los dos ciclos que desarrollan la fase de pedido y una o dos ilustraciones que desarrollan la fase de ejecución.

Es decir que es posible visualizar, como cuadros de una película o video, el flujo de información interna del procesador, el intercambio entre registros, la acción de las líneas de control y en definitiva el estado final de cada registro, luego de que las tareas activadas en cada ciclo se concretan.

Con el propósito de que el alumno recree lo que ocurre en cada ciclo, la secuencia de Ilustraciones incluye varias imágenes en blanco, correspondientes a distintas fases de pedido de instrucción, para que sean completadas describiendo toda la secuencia de ejecución.

Si bien no ha sido tratado en este texto, el modelo permitiría analizar el desarrollo de instrucciones complejas, como las interrupciones o los llamados a subrutinas, para lo que se ha dispuesto un registro genérico, que se grafica en gris, que podría funcionar como Stack Pointer.

Descripción General del Modelo

El procesador que armaremos posee registros de datos de 4 bits y un bus de datos que puede funcionar en 4 y 8 bits.

El bus de direcciones posee, solo 4 líneas, por lo que solo podremos direccionar 2^4 , es decir 16, posiciones de memoria.

Su set de instrucciones incluye 9 instrucciones. Todas las instrucciones poseen el mismo formato: 4 bits para su código de operación, 4 bits para su dato asociado (referencia a los operandos).

Todos los elementos físicos del procesador y la memoria pueden ser construidos a partir de los conocimientos previos que se desarrollan en el curso, por eso, para abordar el Modelo realizaremos previamente un repaso de todos los contenidos necesarios.

Contenidos Previos

Para comprender el Modelo Didáctico de Procesador, es necesario incorporar los contenidos de las 36 primeras páginas y el apéndice del libro *De la compuerta al computador* de Mario Ginzburg.

Se plantea aquí un breve resumen de los mismos.

Se tratarán los siguientes elementos:

- Compuertas Lógicas
- Decodificador
- Multiplexor
- Unidad Aritmético Lógica
- Clock del Sistema
- Memoria
- Registros

Se sugiere relacionar cada concepto aquí descripto, con el Modelo de Procesador en blanco de la **Ilustración 1**, tratando de encontrar en qué posición se encuentra, cómo está interconectado y qué función cumple.

El Modelo se comprende a partir de la comprensión precisa de cada módulo. La comprensión del funcionamiento del conjunto se comprende, a partir del funcionamiento de cada módulo y su forma de interrelacionarse con los demás.

Compuertas Lógicas

Las compuertas lógicas, implementan electrónicamente las operaciones del Álgebra de Boole. Toda la construcción del procesador se realizará con las 4 compuertas básicas: OR, AND, NOT y XOR.

A estas cuatro compuertas le agregamos la compuerta Tri-State, que si bien no tiene significado dentro del Álgebra de Boole, permite resolver problemáticas de la electrónica interna de los procesadores.

En la **Tabla 1**, se describen para cada una de las compuertas, su símbolo, su tabla de verdad y su ecuación correspondiente en el Álgebra de Boole.






Compuerta:	OR			AND			NOT		XOR			Tri-State			
Gráfico:															
Tabla de Verdad:	A B		Z	A B		Z	A		Z	A B		Z	A B		Z
	0	0	0	0	0	0	0	1	0	0	0	0	0	Sin Conexión	
	0	1	1	0	1	0	1	0	0	1	1	0	1	0	
	1	0	1	1	0	0			1	0	1	1	0	Sin Conexión	
	1	1	1	1	1	1			1	1	0	1	1	1	
Expresión:	Z=A+B			Z=A.B			Z=¬A		Z=A⊕B						

Tabla 1

Decodificador

Como se ha desarrollado en el punto 1.11 del libro *De la compuerta al computador*, un decodificador es un circuito que posee n entradas y 2^n salidas, de forma tal que, a cada combinación de las variables de entrada, una y solo una de las salidas se encuentra en uno y todas las demás en cero.

En la **figura 1**, vemos un decodificador de 4 entradas. La entrada está recibiendo el valor 1000. Sólo la novena salida presenta un uno, permaneciendo en cero las 15 restantes.

Utilizaremos decodificadores para resolver distintas funciones de nuestro modelo.

El primero, conectado al Bus de Direcciones, a la entrada de la memoria, para decodificar la dirección que escribe el procesador en el Bus el procesador (ubicarlo en la *Ilustración 2* y analizar su función).

El segundo, como entrada a la ROM de control (dentro de la Unidad de Control del Procesador), para decodificar los códigos de operación de las instrucciones, y generar

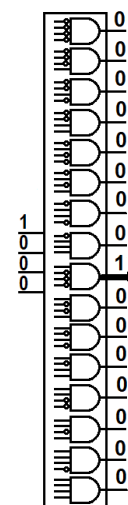


figura 1

los μ códigos (microcódigos) necesarios para que el procesador logre concretar su ejecución (ubicarlo en la **Ilustración 1** y analizar su función).

En ambos casos utilizaremos decodificadores de 4 entradas y 16 salidas. Esto es así, porque la memoria RAM de este sistema será solo de 16 posiciones de 4 bits cada una, y la ROM de control incluirá solo 16 μ códigos que permitirán ejecutar tanto la fase de pedido de instrucción (fetch) como un set de 9 instrucciones.

Multiplexor

El multiplexor es un circuito, que en su configuración básica, posee dos entradas, una salida, y una línea de control. De acuerdo al valor de la línea de control, es posible copiar alternativamente en la salida el valor de una u otra entrada.

En la **figura 2**, Z será igual al valor de A o B, dependiendo si L.C. (la línea de control) es 0 o 1 respectivamente.

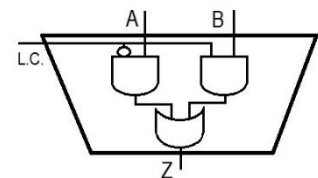


figura 2

Si la estructura descrita, se repitiera 4 veces y se unieran entre sí las líneas de control, podríamos tener un Multiplexor con 2 entradas de 4 bits cada una, y una salida de 4 bits. Podemos observar en la **figura 3**, un multiplexor de 2 entradas de 4 bits. Según sea E igual a 0 o a 1, el dato que aparecerá en la salida, será el de la entrada de la izquierda o el de la entrada de la derecha, respectivamente.

En el modelo, utilizaremos conjuntos de 4 multiplexores básicos para resolver dos necesidades. El primero a la entrada del Registro RL, lo que permitirá seleccionar su entrada entre la salida de la ROM de control o del registro RI. Otro, a la entrada del registro IP, para poder seleccionar si al registro IP la información llega desde el sumador de dos, que se conecta a su salida, o desde la salida de la UAL que alimenta todos los registros. Se sugiere ubicar en la **Ilustración 1** estos dos multiplexores de dos entradas de cuatro bits cada uno y analizar cómo pueden seleccionar el origen de la información que se pasa a la etapa siguiente.

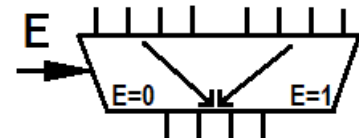


figura 3

Unidad Aritmético Lógica (UAL)

En el punto 1.14 de **De la compuerta al computador**, hemos aprendido a construir una UAL de 4 bits, con la capacidad de sumar, restar y calcular los flags Signo, Zero, Carry y Overflow.

Utilizaremos esa misma UAL en el Modelo de procesador. Para simbolizarla utilizaremos el típico trapecio. Dos entradas de 4 bits. Una salida de 4 bits. Una línea de control para controlar si suma o resta y 4 salidas para los Flags.

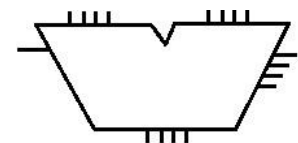


figura 4

Se sugiere repasar, los conceptos desarrollados en Sistemas de Computación I, respecto a cómo la UAL realiza sumas, restas y genera los flags.

Clock del Sistema

El Modelo del Procesador funciona, recibiendo un clock de una frecuencia genérica.

El Clock se distribuye en diversos órganos del procesador. Simplemente se denomina **Ck** cada punto del circuito que recibe el clock del sistema.

El Clock genera su señal desde el momento en que el sistema se enciende, hasta el momento en que se apaga. Es precisamente esta secuencia, la que va generando transiciones entre estados previamente diseñados, para conformar toda la secuencia que permite ejecutar cada una de las instrucciones.

Memoria

El modelo utiliza dos memorias. Una memoria RAM, que permite almacenar el programa y los datos y una memoria ROM, interna al procesador, que se utiliza para generar los microcódigos, que por medio de las líneas de control, controlan todos los movimientos de información dentro del procesador.

Para comprender la memoria RAM, basta estudiar la memoria RAM que se describe en el Apéndice de *De la Compuerta al Computador*.

Para comprender la memoria ROM, tendríamos que graficar la misma memoria, pero retirándole todos los circuitos que se utilizan para guardar información y reemplazar cada Flip Flop mediante un valor fijo, 0 o 1, de acuerdo a lo que la programación de la ROM requiera.

Debemos tener en cuenta, que la memoria RAM, es una memoria de 16 posiciones de 4 bits cada una. Y que la memoria ROM, es una memoria ROM de 16 posiciones, pero de 28 bits cada una. No confundir esta memoria ROM, interna al procesador, con las memorias Flash-ROM que poseen los sistemas para almacenar los programas de arranque.

Registros

En el punto 1.20 de *De la compuerta al computador*, se han construido registros con Flip Flops Maestro-Esclavo.

En este Modelo, utilizaremos los siguientes registros, todos de 4 bits, salvo el RI.

- Dos Registros acumuladores: A y B (para contener los datos, sus resultados parciales y finales)
- Un Registro IP *Instruction Pointer* (que contiene en todo momento, la dirección de memoria de la próxima instrucción)
- Un Registro RI *Registro de Instrucciones*, único registro de 8 bits. El Registro RI, contiene la instrucción que se está ejecutando.
- Un Registro RDI *Registro del Bus de Direcciones*. El registro que se utiliza para fijar un valor en el Bus de Direcciones. El valor que está guardado en RDI, se replica hacia el exterior del procesador por el Bus de Direcciones.
- Un Registro RE *Registro de Estado*. Es el registro que se utiliza para guardar los flags.
- Un Registro Auxiliar cero. Para utilizar el valor 0000, cuando se requiera mover información entre las salidas de los registros y las entradas, sumándoles cero.
- Y por último, el RL, el registro que direcciona la posición de memoria de la ROM de control, que contiene cada micro-operación que se activa en cada pulso de Clock.

Todos los Registros se construyen con Flip-Flops Maestro-Esclavo.

Todos los registros salvo el RL, reciben el Clock negado en los Maestros y sin negar en los Esclavos. El registro RL, es el único que recibe el Clock sin negar en los Maestros y negado en los Esclavos.

Veamos el efecto de esta particular configuración de los Clocks de los Registros.

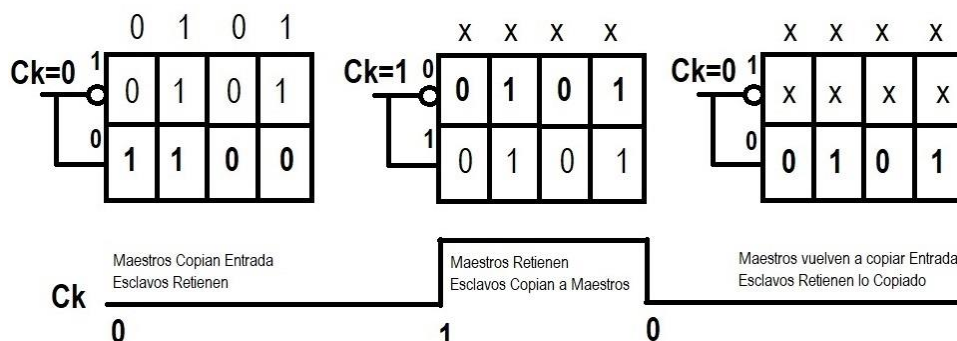


figura 5

En la **figura 5** podemos ver un registro como el A, B, RE, RDI, RI o IP. En esta secuencia vemos como evolucionan Maestros y Esclavos, mientras que el Clock realiza una secuencia pasando de cero a uno y nuevamente a cero.

Mientras el Ck es igual a cero, los Maestros copian una nueva entrada, y los Esclavos retienen. Ésta será su condición natural.

Luego puede observarse que la salida de este registro cambia cuando el Clock pasa de cero a uno. En este momento, los Maestros pasan a retener lo que copiaron y los Esclavos pasan a copiar lo que los Maestros retienen. Es el momento en que los Esclavos cambian de valor. Es decir, es el momento en que el registro se actualiza, que coincide con el pasaje del Clock de cero a uno.

Es decir, los Registros A, B, IP, RDI, RI y RE, al estar contruidos de esta forma, cambiarán su estado justo en el momento en que el Clock pasa de cero a uno.

En cambio, el Registro RL está construido de esta forma:

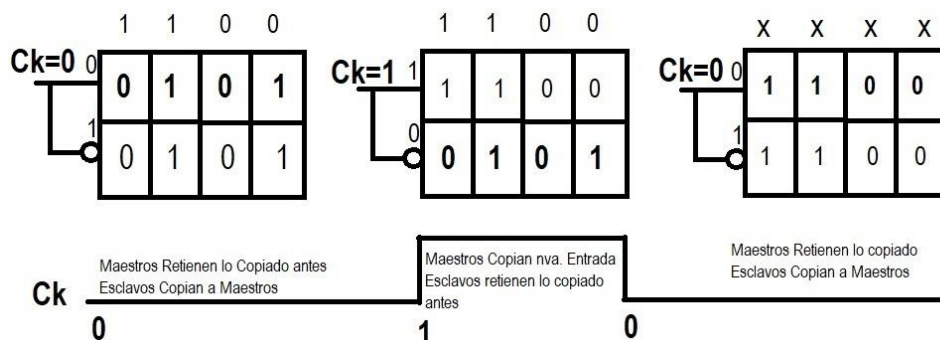


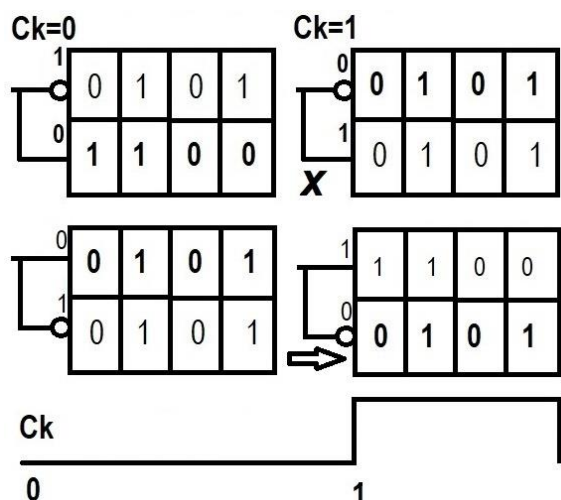
figura 6

En este particular registro de la **figura 6**, cuando comienza el ciclo de Clock, con valor cero, los Maestros están reteniendo, y los Esclavos copiando.

Por lo tanto, cuando el Clock pasa a uno, los Esclavos pasan a retener y mantener su salida.

Recién cuando el clock pasa de uno a cero, cambia el estado de los Esclavos. Es decir, que en el Registro RL, el cambio del contenido del registro, su actualización ocurre cuando el Clock pasa de uno a cero.

Esta diferencia entre todos los demás Registros y RL, nos permitirá armar una secuencia continua de cambios.



Los cambios en un tipo de registros, originarán un cambio en los otros, los que luego volverán a genera un cambio en los primeros.

Basta que el programa y los datos estén cargados en la memoria. Basta que el puntero de instrucciones (IP) esté apuntando a la próxima instrucción, para que la secuencia de ejecución se desarrolle naturalmente a partir del acceso de los pulsos de reloj al sistema.

Analicemos ahora, en forma comparada, los registros A, B, IP, RDI, RE y RI contra el registro RL.

Arriba vemos el comportamiento de un registro del tipo de A, B, IP, RDI, RE o RI, ante la entrada de un pulso de reloj.

figura 7

Abajo vemos el comportamiento de un registro del tipo de RL, ante la entrada de un pulso de reloj.

El RL, mantiene su valor durante todo el ciclo, en cambio los demás, cambiarán cuando el clock pase de cero a uno.

Si logramos alimentar con la salida del RL el decodificador de entrada a la ROM de Control, lograremos que durante todo un ciclo, se active una única posición de la ROM y en consecuencia, las líneas de control que activan todo el procesador mantendrán su valor durante un ciclo completo.

Cuando ingrese el próximo ciclo, se producirá el cambio de la salida del RL, y en consecuencia se activará un nuevo conjunto de valores de las líneas de control, es decir una nueva micro operación.

En nuestro Modelo, para poder visualizar ambas transiciones, representaremos los dos estados de cada Registro en un único diagrama como el que se muestra a continuación.

El registro superior es similar a los registros A, B, IP, RDI, RE y RI. El registro inferior de la figura es similar al registro RL. Ahora, en la **figura 8**, representamos la misma transición que se representa en la **figura 7**.

En una misma imagen podemos ver el valor que toman los Maestros y los Esclavos cuando el Clock es cero, y al lado podemos ver el contenido de esos mismos flip-flops, cuando el clock es 0.

De esta forma graficaremos en cada imagen que dibujemos del procesador, el valor de todos los registros cuando el Clock es cero y el valor de todos los registros cuando el Clock es uno.

Siempre se muestra en **negrita** los valores retenidos por los Flip-Flops y en texto normal, los valores copiados por los Registros.

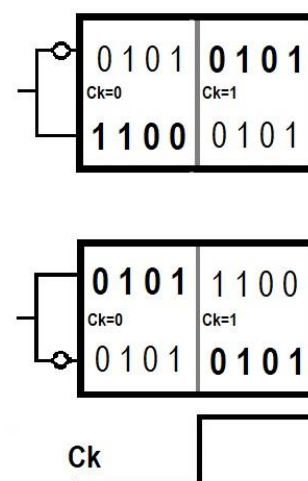


figura 8

Estructura del Modelo de Procesador

Datapath

El Datapath de este procesador es análogo al que planteó en su momento el Modelo de Von Neumann. La UAL es el dispositivo que se utiliza para relacionar las salidas de los registros con las entradas de los registros.

Los contenidos de los registros A, B, parte derecha del RI (que almacena el Dato Asociado de la instrucción) y parte derecha del RDA (que contiene el dato que entrega la memoria) tienen acceso mediante compuertas Tri-State a cualquiera de las dos entradas de la UAL. El IP tiene acceso mediante compuertas Tri-State, solo a la entrada Y de la UAL.

Las compuertas Tri-State son comandadas desde la UC, mediante líneas de control.

Existe un registro auxiliar, que contiene siempre cero (0000), cuya salida también se conecta a las entradas de la UAL mediante compuertas Tri-State.

Si activando la línea de control que corresponda, logramos que el contenido de un registro llegue a una de las entradas de la UAL, activando la línea de control complementaria del registro auxiliar cero, y comandando la UAL para que sume, estamos logrando que la UAL sume al contenido de ese registro, el valor cero, lo que va a determinar que a la salida de la UAL, tendremos exactamente el mismo valor que tiene el registro almacenado.

La salida de la UAL, está conectada con las entradas de los registros A, B, IP (mediante un multiplexor y RDI).

Por lo tanto, como todos esos registros normalmente tienen cero (0) en los clocks, sus Maestros copiarán el dato, pero nunca ingresará ese dato a los Esclavos, pues para ello es necesario que un pulso de clock, entre a la entrada de clock del registro.

Escritura de un Registro

Para controlar el acceso del Ck a la entrada de Ck del Registro, el Ck del sistema se conecta a una AND donde la otra entrada es la línea de control que habilita la escritura del Registro.

De esta manera, cuando la línea de control está en uno, el ciclo del Ck entra al Registro. De esta forma, mientras el Ck está en cero, los Maestros copian y los Esclavos retienen. Justo al momento de que el Ck se pone en uno, los Maestros comienzan a retener, y los Esclavos copian lo que los Maestros retienen, actualizando su valor.

De esta forma, se asegura que solo cuando la línea de control esté en uno, se escribe en el registro.

Se observa en la **figura 9**, la forma de controlar el acceso del Ck al registro. Mientras la línea de control CA permanece en cero, el registro retiene en sus Esclavos el valor que guarda.

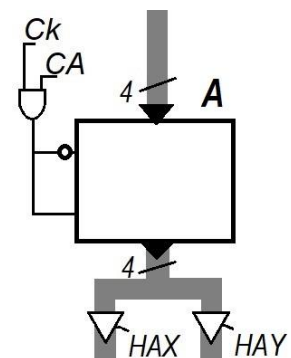


figura 9

Acceso a un Registro

Como se observa en la **Ilustración 1**, la salida de los registros puede acceder tanto a la entrada X como a la entrada Y de la UAL.

El Registro IP, solo accede a la entrada Y de la UAL. Y el registro RDI, accede al Bus de Direcciones.

Actualización del Registro IP

En la **figura 10**, puede observarse el proceso de actualización del registro IP.

La salida del registro, siempre alimenta un sumador, que lo incrementa en dos. Dado que las instrucciones ocupan dos posiciones de memoria, el incremento en dos, permite apuntar a la instrucción siguiente.

Para que el registro IP se actualice, se necesita que ocurran dos hechos. La línea CIP debe valer 1 y la línea M debe valer 0. De esta forma se asegura, que a la entrada del registro IP, aparezca su salida incrementada, y que el pulso de Ck que entra a la compuerta AND, ingrese en el Clock del registro.

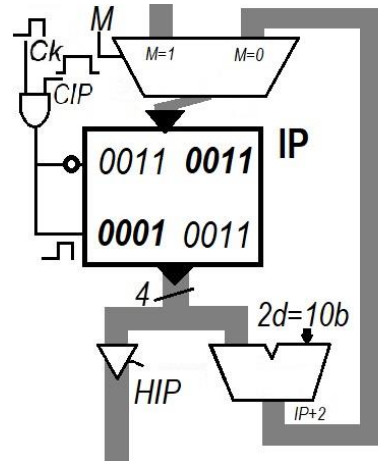


figura 10

Dentro del registro se encuentra un par de valores. A la izquierda el valor que toman los Maestros y los Esclavos cuando el Ck es cero, y a la derecha, el valor que toman los Maestros y los Esclavos cuando el Ck es uno.

Cuando el CK es cero, los Maestros copian el valor incrementado, mientras los Esclavos retienen aún el valor anterior. Cuando el CK es uno, los Maestros retienen el valor incrementado, mientras los Esclavos ahora copian el nuevo valor incrementado.

De esta forma se consigue que el IP apunte a la siguiente instrucción.

Cuando la línea de control del Multiplexor vale 1, se conecta la entrada del IP a la salida de la UAL, habilitando al registro a ser actualizado durante la ejecución de una instrucción, lo que se realiza en las instrucciones de salto. (Modo de direccionamiento Relativo).

Modelo de procesador en Blanco

Vemos a continuación la **Ilustración 1** del Modelo de Procesador.

Puede seguirse en él todo lo descrito sobre el mismo hasta el momento.

Decodificadores, Multiplexores, Unidad Aritmético Lógica, Registros, Datapath, Circuito de actualización del IP, Actualización de cada Registro.

Procederemos ahora a analizar la Unidad de control

Unidad de Control

La Unidad de Control, es la que controla las líneas de control que comandan todo el funcionamiento del procesador.

Todos los órganos del procesador, Registros, UAL, Acceso a los Buses Internos, Acceso a la memoria, Multiplexores, etc. son controlados por los valores que toman las líneas de control.

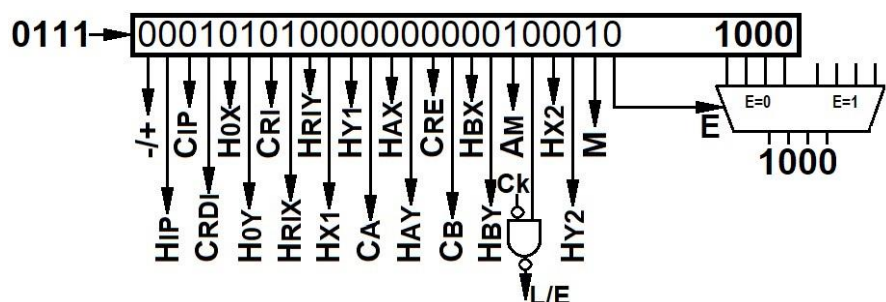


figura 11

Las líneas de control, son la salida de datos de la ROM de la Unidad de Control. De acuerdo al valor que toma el Registro RL, se accede a distintas posiciones y los valores de las líneas cambian.

Puede observarse en la **figura 11**, la salida de la ROM de control, ante la activación de la posición 0111.

Los valores que toman las líneas se ajustan al contenido de esa posición de la ROM.

Todas las líneas que comienzan con H, son las que habilitan o no, el acceso de los datos guardados en los registros. a las entradas de la UAL por los buses internos X o Y. HIP, controla el acceso del dato contenido en el registro IP. H0X y H0Y, controlan el acceso del dato auxiliar cero. HRIX y HRIY, controlan el acceso del dato del RI a la UAL. HAX, HAY, HBX y HBY controlan el acceso de los registros acumuladores a la UAL.

Todas las líneas que comienzan con C, son las que controlan el acceso de los datos a los registros. CIP, CRDI, CRI, CA, CB y CRE permiten al setearse en uno, que un ciclo de reloj ingrese al registro correspondiente, desencadenando el procedimiento de copia y retención del dato.

Luego tenemos líneas específicas: -/+ para controlar la operación de la UAL, M y E, para controlar a los Multiplexores, AM para controlar el modo de acceso de la memoria (una o dos posiciones), L/E para controlar si a la memoria RAM se la lee o escribe.

Por último, los cuatro últimos dígitos de la posición de memoria de la ROM de Control de la Unidad de Control, contienen la posición de ROM que deberá ser accedida en el pulso siguiente del reloj. Al acceder la misma al Multiplexor de entrada al RL, según sea el valor de E, la próxima dirección seleccionada de la ROM surgirá de la misma ROM, o surgirá del RI.

De esta forma, el acceso a la ROM de control estará determinado, ya sea por el código de operación de la instrucción que se acaba de leer, o por la misma ROM de control, indicando la secuencia a seguir.

Instrucciones

Las instrucciones en lenguaje máquina tienen un formato específico.

Según el tipo de procesador, cada instrucción hace referencia a uno o más datos.

Nosotros nos concentraremos en instrucciones que hacen referencia hasta un máximo de dos datos. En el caso de ser dos los datos, uno de ellos residirá siempre en un registro.

Usaremos en nuestro modelo, instrucciones que tienen un código de operación y un dato asociado.

Código de Operación y Dato Asociado tendrán siempre cuatro bits cada uno, es decir, ocuparán dos posiciones de memoria del modelo.

Instrucción en lenguaje máquina:

Código de Operación (Cod-Op)	Dato Asociado al Cod-Op (referencia a los operandos)
Indica al Procesador: Qué hacer?	Indica al Procesador: Con qué?

figura 12

El código de operación, al ser leído y decodificado por el procesador, permite definir qué hará el procesador en la ejecución de la instrucción. El dato asociado, hace referencia a los operandos de la instrucción, por lo que permite definir qué datos utilizará el procesador para realizar la tarea.

Según el Modo de Direccionamiento, el Dato Asociado hará referencia a distintos operandos.

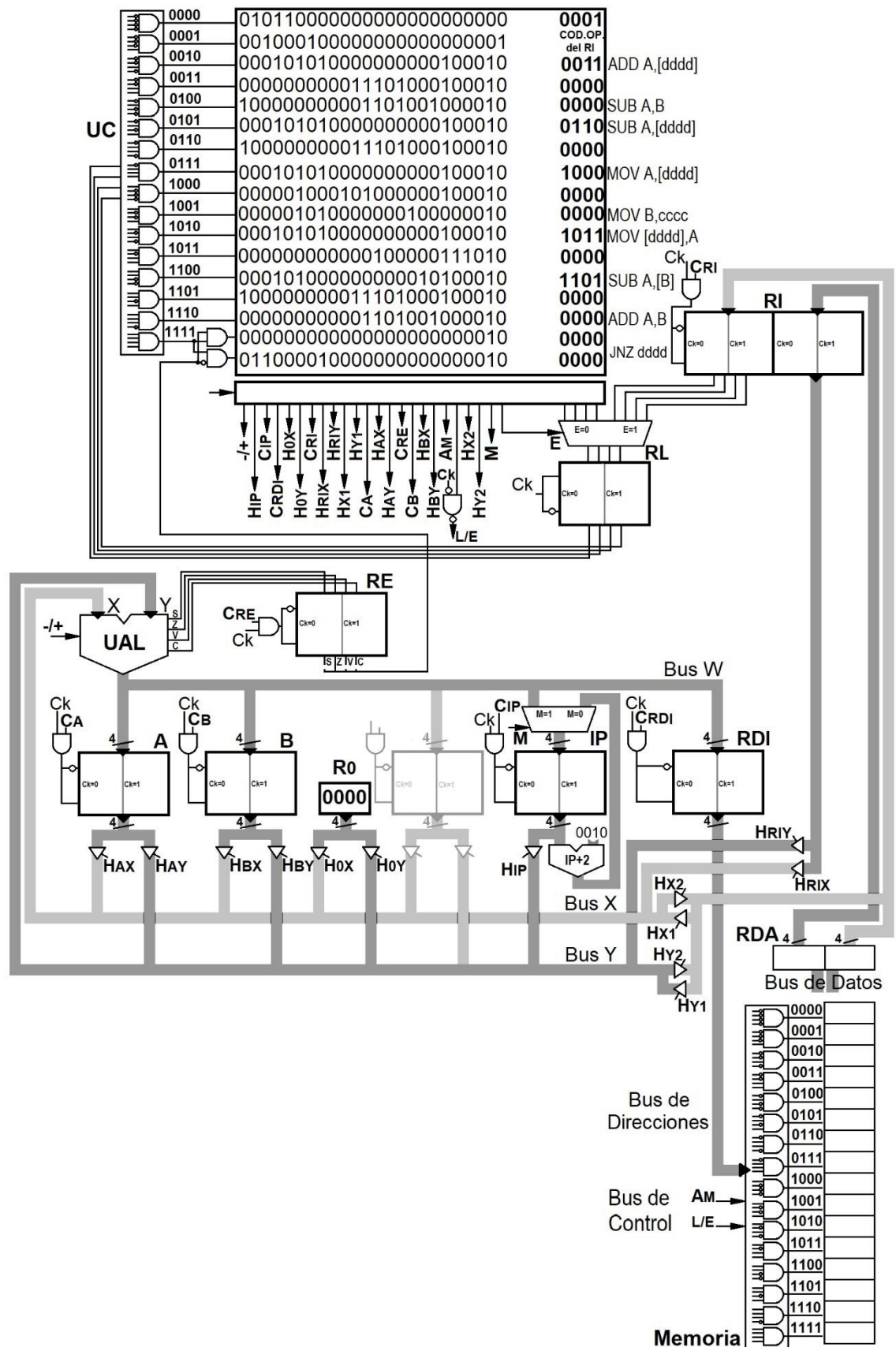


Ilustración 1

En códigos mnemónicos (Assembler) describimos las instrucciones con dos, tres o cuatro letras que indican qué hace la instrucción y luego dos, uno o ningún dato. Cuando son dos datos, se separan con coma. Se entiende que cuando una coma divide los dos datos, la acción de la instrucción se inicia en el dato de la derecha y repercute sobre el dato de la izquierda.

SUB A,[dddd] significa que el valor contenido en la dirección de memoria dddd, debe restarse de A y guardarse en A.

El Modelo de Procesador puede ejecutar las siguientes 9 instrucciones. Es decir, que armamos un procesador cuyo Set de instrucciones es el siguiente:

El código de operación 0010 ejecuta la instrucción ADD A,[dddd]

El código de operación 0100 ejecuta la instrucción SUB A,B

El código de operación 0101 ejecuta la instrucción SUB A,[dddd]

El código de operación 0111 ejecuta la instrucción MOV A,[dddd]

El código de operación 1001 ejecuta la instrucción MOV B,cccc

El código de operación 1010 ejecuta la instrucción MOV [dddd],A

El código de operación 1100 ejecuta la instrucción SUB A,[B]

El código de operación 1110 ejecuta la instrucción ADD A,B

El código de operación 1111 ejecuta la instrucción JNZ rrrr.

Siendo dddd la dirección donde está el dato. Siendo cccc una constante. Siendo rrrr la dirección donde debe saltar la ejecución en el caso que el flag Z indique que el resultado de la UAL No fue cero.

Pasamos a ver los Modos de Direccionamiento.

Modos de Direccionamiento

Analizaremos cinco modos de direccionamiento.

Directo

En el Modo de Direccionamiento Directo, el Dato Asociado al Código de Operación es la dirección del dato a operar. Es decir que en la instrucción MOV A,[dddd]: dddd es la dirección de memoria donde está el dato. En códigos mnemónicos (Assembler) se identifican las instrucciones en modo de direccionamiento directo con uno de los dos operandos entre corchetes y dentro de los corchetes la dirección del dato. En nuestro modelo tenemos las instrucciones: ADD A,[dddd] - SUB A,[dddd] - MOV A,[dddd] - MOV [dddd],A

Inmediato

En el Modo de Direccionamiento Inmediato, el Dato Asociado al Código de Operación es el dato a operar. Es decir que en la instrucción MOV B,cccc: cccc es el dato que el procesador deberá mover a B. En códigos mnemónicos (Assembler) se identifican las instrucciones en modo de direccionamiento inmediato, con un registro a la izquierda de la coma y un dato a la derecha, sin paréntesis, que es el que debe moverse al registro. En nuestro modelo tenemos la instrucción MOV B,cccc.

Registro

En el Direccionamiento en Modo Registro, el Dato Asociado al Código de Operación es una referencia al registro donde se encuentra el dato. Es decir que en la instrucción SUB A,B, el dato asociado que aparece en el código máquina es una referencia que le indica al procesador que el dato a operar, el dato a restar de A, es el contenido del registro B. En nuestro modelo tenemos las instrucciones SUB A,B y ADD A,B

Indirecto o Indexado

En el Direccionamiento Indirecto o Indexado, el Dato asociado al Código de Operación es una referencia al registro donde se encuentra la dirección de memoria del dato a operar. Es decir que en la instrucción SUB A,[B] el dato asociado que aparece en el código máquina de esa instrucción, es una referencia que indica al procesador que la dirección de memoria del dato que debe restar de A, se encuentra guardada en el registro B. En nuestro modelo tenemos la instrucción SUB A,[B]

Relativo

En el Modo de Direccionamiento Relativo, el Dato Asociado al Código de Operación, es un valor, que en definitiva terminará sumándose al valor de IP. Se utiliza el Modo de Direccionamiento relativo para las instrucciones que determinan saltos en la normal secuencia de ejecución de instrucciones. Existen saltos sin condición, saltos condicionales al estado de algún flag y llamados a subrutinas que emplean el Modo de Direccionamiento Relativo. En nuestro modelo utilizaremos la instrucción JNZ dddd, donde el dato asociado, incluirá siempre el desplazamiento relativo del IP para pasar de la dirección de la próxima instrucción a dddd.

Programa a Ejecutar

Para analizar los efectos de las distintas instrucciones, con sus distintos modos de direccionamiento, utilizaremos el siguiente programa.

El mismo solo realiza dos restas, una en Modo de Direccionamiento Directo otra en Modo de Direccionamiento Indexado, para comprobar si el resultado guardado en A es cero.

Como el resultado aún no es cero, se ejecuta un salto para repetir la resta en Modo de Direccionamiento Indexado.

Cuando el resultado de A, es cero, en este caso se procede a guardar el dato en la memoria mediante una instrucción de Direccionamiento Directo y concluye el programa.

0000	0111	MOV A,[1101]	Este es el programa y los datos a correr en el programa.
0001	1101		
0010	0101	SUB A,[1110]	Para ello, se utilizaran imágenes en blanco como la propuesta en la Ilustración 1, se procederá a escribir programa y datos en todas ellas y se pondrá en la primera de ellas la condición inicial.
0011	1110		
0100	1001	MOV B,1110	
0101	1110		
0110	1100	SUB A,[B]	Condición Inicial:
0111	0000		
1000	1111	JNZ 0110	1. Programa y Datos deben encontrarse en la memoria 2. El contenido de IP, al momento de iniciar el programa (Ck=0) debe ser la dirección de memoria de la primer instrucción (es decir 0000)
1001	1100		
1010	1010	MOV [1111],A	3. El contenido de RL, al momento de iniciar el programa (Ck=0) debe ser 0000.
1011	1111		
1100			4. Debe ROTULARSE la hoja adecuadamente indicando: a. PRIMER CICLO b. FASE PEDIDO c. I1: MOV A,[1101]
1101	0110		
1110	0010		
1111			

Luego de analizar primero, que líneas de control se activaron, y luego el efecto de esas líneas de control mientras el Clock sigue valiendo cero, se completarán los valores de todos los registros (Maestros y Esclavos) para Ck=0.

Luego se pasará el Clock a uno y se analizará su efecto, completando Maestros y Esclavos de todos los registros.

Una vez completo el análisis del primer ciclo de reloj, se procederá a completar la segunda hoja, siguiendo estas pautas para el Inicio de hoja nueva

Inicio de hoja nueva:

Debemos entender que una hoja nueva, representa un nuevo ciclo de reloj entrando al procesador. Por lo tanto para pasar de una hoja completa a una hoja nueva deberemos:

1. Verificar que programa y datos se encuentren en la memoria
2. Completar el contenido de todos los registros salvo el RL, copiando en los Esclavos para $Ck=0$, el valor que contienen los Esclavos para $Ck=1$ de la hoja anterior
3. Calcular el nuevo valor de RL
4. Si en el ciclo anterior se direccionó la memoria, completar en RDA el valor entregado por la memoria
5. ROTULAR la hoja de acuerdo al nuevo ciclo y la fase en la que se encuentra.

Luego de analizar primero, que líneas de control se activaron, y luego el efecto de esas líneas de control mientras el Clock sigue valiendo cero, se completarán los valores de todos los registros (Maestros y Esclavos) para $Ck=0$.

Luego se pasará el Clock a uno y se analizará su efecto, completando Maestros y Esclavos de todos los registros.

Una vez completo el análisis del ciclo de reloj completo, se procederá a iniciar una nueva hoja.

Este procedimiento debe ejecutarse hasta completar el programa