TEMA 2 : LA UNIDAD CENTRAL DE PROCESO (C.P.U.)

- ***Introducción.***

El procesador es el mecanismo de ejecución de instrucciones. Éste, a su vez, está formado por distintos mecanismos tales como la ruta de control (con depuración de problemas que aparecen al diseñar la unidad de control), repertorio de instrucciones y modos de direccionamiento. Ahora bien ¿qué módulos necesito para pasar los datos? La ruta de datos ya ha sido definida por lo que paso directamente al diseño de la U.C..

- ***Repertorio de instrucciones elegido.***

Dependiendo de los distintos tipos de instrucciones elegidas para nuestro repertorio de instrucciones y de su tamaño en bytes tendremos tres formatos tipo: 1, 2 ó 3 bytes.

- ***Formato tipo 1.***

7 0

8 bits

En la siguiente *tabla* se muestran las *instrucciones* pertenecientes a nuestro repertorio y que se corresponden con el *formato tipo 1*.

Operación	**Sintaxis**	**Descripción**	**Cód. Operación (COP)**
Suma	ADD r1	A ! A + r1	30h, 31h, 32h, 33h, 45h
Resta	SUB r1	A ! A − r1	18h, 19h, 1Ah, 1Bh, 46h
And	AND r1	A ! A and r1	20h, 21h, 22h, 23h, 48h
Or	ORA r1	A ! A or r1	24h, 25h, 26h, 27h, 49h

IMPORTANTE

- ***Nota 1, sobre C.O.:*** Vendrá en hexadecimal y donde 1 dígito o nibble equivale a 4 bits.
- El conjunto formado por la A.L.U., la U.C., el bus de datos, el SP, el multiplexor, los registros de estado, DEC y registros restantes forman el *microprocesador (P).*
- El microprocesador junto con la memoria y los dispositivos de E/S forman un *microcontrolador (Controlador).*
- El microcontrolador junto con una A.L.U. 2 para operaciones que se repiten mucho forman un *procesador digital de señales (DSP).*

- ***Formato tipo 2.***

15 8 7 0

8 bits 8 bits

En la siguiente *tabla* se muestran las *instrucciones* pertenecientes a nuestro repertorio y que se corresponden con el *formato tipo 2*.

Operación	**Sintaxis**	**Descripción**	**Cód. Operación (COP)**

Suma Inmediata	ADI dato	A ! A + dato	35h
Resta Inmediata	SUI dato	A ! A − dato	36h
And Inmediata	ANI dato	A ! A and dato	68h
Or Inmediata	ORI dato	A ! A or dato	69h
Xor Inmediata	XRI dato	A ! A xor dato	6Ah

- ***Formato tipo 3.***

23 16 15 8 7 0

8 bits 8 bits 8 bits

En la siguiente *tabla* se muestran las *instrucciones* pertenecientes a nuestro repertorio y que se corresponden con el *formato tipo 3*.

Operación	**Sintaxis**	**Descripción**	**Cód. Operación (COP)**
Cargar	LDA dir	A ! M (dir)	70h
Almacenar	STA dir	M (dir) ! A	71h
Salto Incondicional	JMP dir	PC ! dir	74h
Salto si FZ = 1	JZ dir	Si FZ = 1 ! PC ! dir	72h

IMPORTANTE

Nota 2, sobre instrucción que falta: *MVI dato16 , SP* es un movimiento inmediato de 3 bytes porque mueve 16 bits a SP; lo mismo ocurre con *MVI dato16 , HL* que NO está implementada pero que la ruta de datos me lo permite.

Ej. : *MVI 3000h , SP*; SP por defecto en 0000h. Gráficamente sería:

MEMORIA

SP

8 8 8 8

8 8

- ***Empleo de buses.***

Dependiendo de los distintos tipos de instrucciones elegidas para nuestro repertorio de instrucciones y de su tamaño emplearemos rutas de datos de 1, 2 ó 3 buses.

- ***Ruta de datos con bus único.***

Las características de la ruta de datos de bus único son las siguientes:

- Existirá un único bus de datos de 8 bits.
- Tendrá un bus de direcciones de 16 bits.

- Necesitará registros temporales para liberar al bus. Esto será necesario en el diseño de la U.C. para cargar en este registro TEMP el primer operando y así ir a por el segundo y poder realizar la operación pertinente.

- ***Ruta de datos con dos buses.***

Las características de la ruta de datos de dos buses son las siguientes:

- Existirán dos buses de datos de 8 bits.
- Tendrá comunicación con la memoria a través de HL

- Necesitará un enlace de bus.

- ***Ruta de datos con tres buses.***

Las características de la ruta de datos de dos buses son las siguientes:

- Existirán tres buses de datos de 8 bits.
- Tendrá comunicación con la memoria a través de HL

- Necesitará un enlace de bus.

- ***Fases de ejecución de las instrucciones.***

4.1. Introducción.

Las distintas fases de ejecución de cada instrucción de nuestro repertorio se pueden resumir de forma genérica en el organigrama siguiente:

La primera fase es la de buscar instrucciones para saber cuál ejecutar; la segunda es la de decodificación de instrucción porque he de saber qué instrucción es y evaluar los registros de estado; el resto de fases no son más que lo mostrado antes en el organigrama.

En cuanto a la forma de almacenar la información en este dispositivo electrónico ha sido diseñado con la de disparo por flanco.

IMPORTANTE

Nota 3, sobre la forma en que los dispositivos electrónicos almacenan la información. : Existen cuatro formas de hacerlo:

- Disparado por nivel:

Ej. :

Información en E Señal de control c que recibirá nivel bajo `0' (entrada negada: _0) o alto `1' (tal como muestra la figura).

El tipo de nivel nos lleva a dos tipos distintos:

- Por nivel bajo o activado por cero lógico.
- Por nivel alto o activado por uno lógico.

Estos tipos de disparo provienen de un cronograma como el que se muestra a continuación:

Nivel alto (t0 = 20 ns)

Nivel bajo (t0 = 15 ns)

- Disparado por flanco:

Ej. :

Información en E Señal de control c que recibirá flanco de bajada (entrada negada: _0) o de subida (tal como muestra la figura).

El tipo de flanco nos lleva a dos tipos distintos:

- Por flanco de bajada.
- Por flanco de subida.

Estos tipos de disparo provienen de un cronograma como el que se muestra a continuación:

Flanco de subida Flanco de bajada

Tc =1/ fc

t0 t0 t0 t0

Ahora bien, cada fase de ejecución durará un ciclo de reloj, esto hace que nos aparezca el concepto de periodo o ciclo de reloj que es el periodo de la señal que utilizo para sincronizar las instrucciones y que se denota, como muestra la figura de arriba, por Tc. La instrucción NOP sirve para ajustar bucles de tiempo de ejecución de programa. La duración del ciclo de reloj dependerá de la duración de la acción más lenta. No se puede realizar una L/E en el banco de registros en el mismo ciclo de reloj.

4.2. Consideraciones.

Debemos tener en cuenta las siguientes:

- Las funciones básicas de la MaNoTaS, que nos ayudarán a diseñar la U.C., son:

- Acceso al banco de registros y realizar las operaciones que se puedan realizar con él.
- Acceso a memoria con lo que se puede leer y escribir en ella.
- Operaciones en la A.L.U..

- Suposiciones:

- El tiempo de cada una de estas funciones es igual a un ciclo o periodo de reloj.
- El coste del resto de los elementos es cero; esto implica que al NO realizar funciones básicas, como *LOAD RT* ! carga registro temporal, el coste temporal es despreciable.

- Las acciones asociadas a una fase ocurren en paralelo y así reducir el tiempo de ejecución como, por ejemplo, en la *fase de búsqueda de instrucción: RI ! M (PC) y PC ! PC + 1.*

- Las acciones asociadas a fases sucesivas ocurren en serie.

4.3. Establecimiento de las fases.

Antes de mencionar cada una de las distintas fases de ejecución y analizarlas por separado, en la siguiente figura se muestra un organigrama resumen de todas ellas:

Fase 1

Fase 2

Fase 3

ADD,SUB,ANA,ORA ADI,SUI,ANI,ORI LDA, STA, JMP JZ

No

Si

Fase 4

Fase 5 LDA STA JMP

Fase 6

- ***Fase 1: Búsqueda de la instrucción.***

Consta de dos pasos:

- RI ! M[PC]
- PC ! PC +1

- ***Fase 2: Decodificación y evaluación.***

Tanto esta fase como la anterior son comunes a todas las instrucciones.

- ***Fase 3: Obtención de operandos y evaluación del código de condición Z.***

- ***Caso I.*** Instrucciones Aritmético − Lógicas.

- Modo de direccionamiento directo a registro.

- TEMP ! r1

- Modo de direccionamiento inmediato.

- TEMP ! M[PC] (TEMP ! dato)
- PC ! PC +1

- ***Caso II.*** Instrucciones de referencia a memoria y salto incondicional.

- L ! M[PC] (L ! DirL)
- PC ! PC +1

- ***Caso III.*** Instrucción de salto condicional.

- Z?

- ***Fase 4: Obtención de operandos, ejecución y conclusión de las instrucciones Aritmético − Lógicas.***

- ***Caso I.*** Instrucciones Aritmético − Lógicas.

- A ! A op TEMP

- ***Caso II.*** Instrucciones de referencia a memoria y salto incondicional.

- H ! M[PC] (H ! DirH)
- PC ! PC +1

- ***Caso III.*** Instrucción de salto condicional.

- L ! M[PC] (L ! DirL)
- PC ! PC +1

- ***Fase 5: Conclusión de las instrucciones de acceso a memoria y salto incondicional y obtención de operandos.***

- ***Caso I.*** Instrucciones de referencia a memoria.

- Instrucción de carga.

- A ! M[HL]

- Instrucción de almacenamiento.

- M[HL] ! A

- ***Caso II.*** Instrucciones de referencia a memoria y salto incondicional.

- PC ! HL +1

- ***Caso III.*** Instrucción de salto condicional.

- H ! M[PC] (H ! DirH)
- PC ! PC +1

- ***Fase 6: Conclusión de la instrucción de salto condicional.***

- PC ! HL +1

4.4. Diagrama de fases final.

El organigrama antes mencionado de cada una de las distintas fases de ejecución, después de analizarlas por separado, quedaría simplificado tal y como se muestra en la siguiente figura:

Fase 1

Fase 2

JZ = 0

Fase 3

ADD,SUB,ANA,ORA ADI,SUI,ANI,ORI LDA, STA, JMP JZ =1

Fase 4

Fase 5 LDA STA JMP

4.5. La ruta de datos.

A partir del organigrama simplificado de las fases de ejecución de las instrucciones podemos diseñar la siguiente ruta de datos:

A.L.U.

4.6. Ejemplo.

Veamos ahora un ejemplo de cómo se ejecutaría una instrucción en cada una de sus fases:

** Aritmético − lógica (con direccionamiento directo a registro): ADD A! A + r1.*

1ª Fase (Común a todas):

Paso 1 ! Buscar la instrucción: RI ! M[PC], lo que apunta en memoria el PC me lo llevo a RI.

Paso 2 ! Preparación de la instrucción siguiente PC ! PC + 1.

2ª Fase (Común a todas):

Paso 1 ! Decodificar la instrucción.

Paso2 ! Evaluación del registro de estado.

3ª Fase: Búsqueda del operando 1 en Ac y el operando 2 en un registro que hay que llevarlo a la ruta de datos con lo que se guarda en el registro Temp para ejecutar mientras la instrucción.

4ª Fase: Se opera el dato del Ac con el de Temp y así se ejecuta la operación que indica la instrucción así como al mismo tiempo se busca la siguiente instrucción a ejecutar A! Ac + Temp.

**Aritmético − lógica (con direccionamiento inmediato): En dirección de memoria 1000h ADI 23h se mostrará en memoria como:*

1000 h C.O. ADI

1001 h Dato

Las fases de ejecución son:

1ª Fase (Común a todas):

Paso 1 ! Buscar la instrucción: RI ! M[PC], lo que apunta en memoria el PC me lo llevo a RI.

Paso 2 ! Preparación de la instrucción siguiente PC ! PC + 1.

2ª Fase (Común a todas):

Paso 1 ! Decodificar la instrucción.

Paso2 ! Evaluación del registro de estado.

3ª Fase: Búsqueda del operando en M[PC] que hay que llevarlo a la ruta de datos con lo que se guarda en el registro Temp, TEMP ! M[PC], para ejecutar mientras la instrucción y se incrementa el PC para la siguiente: PC ! PC + 1.

4ª Fase: Se opera el dato del Ac con el de Temp y así se ejecuta la operación que indica la instrucción así como al mismo tiempo se busca la siguiente instrucción a ejecutar A! Ac + Temp.

IMPORTANTE

Nota 4, sobre las fases: Cada fase tiene una duración de un período de reloj con lo que la duración de la instrucción será tantos ciclos como la suma del número total de fases que se realizan durante la ejecución.

**Instrucciones de Referencia a Memoria y Salto Incondicional:*

***LDA 1000h A ! M[dir] ; luego dir = 1000h*

se mostrará en memoria como:

5000 h C.O. LDA

5001 h Dato Parte L (Byte Bajo)

5002h Dato Parte H (Byte Alto)

1ª Fase y 2º Fase (Común a todas).

3ª Fase:

Paso 1 ! Búsqueda del operando en la memoria (posición a la que apunta PC) y llevarlo a la ruta de datos con lo que se guarda la parte baja (byte bajo) de la dirección a cargar en Ac en la parte L del registro HL: L ! M[PC].

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

4ª Fase:

Paso 1 ! Búsqueda del operando en la memoria (posición a la que apunta PC) y llevarlo a la ruta de datos con lo que se guarda la parte alta (byte alto) de la dirección a cargar en Ac en la parte H del registro HL: H ! M[PC].

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

5ªFase:

A ! M[HL]; en Ac meto el contenido de 1000h que es la posición de memoria que tengo en el registro HL y PC está apuntando a la posición de memoria 5003h y ahí está buscando la siguiente instrucción.

***STA 1000h M[dir] ! A; luego dir = 1000h*

se mostrará en memoria como:

5000 h C.O. STA

5001 h Dato Parte L (Byte Bajo)

5002h Dato Parte H (Byte Alto)

1ª Fase y 2º Fase (Común a todas).

3ª Fase:

Paso 1 ! Búsqueda del operando en la memoria (posición a la que apunta PC) y llevarlo a la ruta de datos con lo que se guarda la parte baja (byte bajo) de la dirección a cargar en Ac en la parte L del registro HL: L ! M[PC].

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

4ª Fase:

Paso 1 ! Búsqueda del operando en la memoria (posición a la que apunta PC) y llevarlo a la ruta de datos con lo que se guarda la parte alta (byte alto) de la dirección a cargar en Ac en la parte H del registro HL: H ! M[PC].

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

5ªFase:

M[HL] ! A; el contenido del Ac lo meto como contenido de la posición de memoria que tengo en el registro HL, 1000h, y PC está apuntando a la posición de memoria 5003h y ahí está buscando la siguiente instrucción.

***JMP1000h M[dir] ! A; luego dir = 1000h*

se mostrará en memoria como:

5000 h C.O. JMP

5001 h Dato Parte L (Byte Bajo)

5002h Dato Parte H (Byte Alto)

1ª Fase y 2º Fase (Común a todas).

3ª Fase:

Paso 1 ! Búsqueda del operando en la memoria (posición a la que apunta PC) y llevarlo a la ruta de datos con lo que se guarda la parte baja (byte bajo) de la dirección a cargar en Ac en la parte L del registro HL: L ! M[PC].

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

4ª Fase:

Paso 1 ! Búsqueda del operando en la memoria (posición a la que apunta PC) y llevarlo a la ruta de datos con lo que se guarda la parte alta (byte alto) de la dirección a cargar en Ac en la parte H del registro HL: H ! M[PC].

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

5ªFase:

PC ! HL + 1; meto en PC la posición de memoria (NO el contenido de la posición) que tengo en el registro HL incrementada en una unidad, 1001h, y ésta es la dirección a la que salta el PC y donde se quedará apuntando para buscar la siguiente instrucción.

Ahora bien, se nos presenta un problema en la ruta de datos: ¿Cómo paso al PC la dirección si ésta la tengo en la salida del multiplexor? Existen varias soluciones:

- *Hardware: Dotar a la parte incremento del PC para que incremente o no.*
- *Software: No pongo esa señal y lo que hago es emplear el ensamblador de tal manera que cuando tengo un JMP 2000h, por ejemplo, lo codifique y guarde en memoria esto, pero en memoria tengo como C.O. de JMP el número 1FFF que será lo que interprete la máquina al ejecutar la instrucción JMP dir, 2000h en este caso, para luego decrementar en 1; el mismo proceso se sigue con la instrucción CALL.*
- *Firmware: A mitad de camino del software y el hardware se trata de unas microórdenes que se han introducido en la máquina, por lo que esta solución la resuelve el diseñador de la máquina.*

**Instrucción de Salto Condicional: Se trata de un salto incondicional que se realiza cuando se cumple la condición, como en la instrucción JZ 1000h.*

Las fases de ejecución son:

1ª Fase (Común a todas):

Paso 1 ! Buscar la instrucción: RI ! M[PC], lo que apunta en memoria el PC me lo llevo a RI.

Paso 2 ! Preparación de la instrucción siguiente PC ! PC + 1.

2ª Fase (Común a todas):

Paso 1 ! Decodificar la instrucción.

Paso2 ! Evaluación del registro de estado:

- *Si FZ = 0, entonces se busca la siguiente instrucción a ejecutar en la posición de memoria a la que apunta el PC y la ejecución del salto ya se ha acabado.*
- *Si FZ = 1, entonces 3ª Fase.*

3ª Fase (Común a todas las instrucciones relacionadas con saltos):

Paso 1 ! Búsqueda del operando en la memoria (posición a la que apunta PC) y llevarlo a la ruta de datos con lo que se guarda la parte baja (byte bajo) de la dirección a cargar en Ac en la parte L del registro HL: L ! M[PC].

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

4ª Fase (Común a todas las instrucciones relacionadas con saltos):

Paso 1 ! Búsqueda del operando en la memoria (posición a la que apunta PC) y llevarlo a la ruta de datos con lo que se guarda la parte alta (byte alto) de la dirección a cargar en Ac en la parte H del registro HL: H ! M[PC].

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

5ªFase (Común a todas las instrucciones relacionadas con saltos):

PC ! HL + 1; meto en PC la posición de memoria (NO el contenido de la posición) que tengo en el registro HL incrementada en una unidad, 1001h, y ésta es la dirección a la que salta el PC y donde se quedará apuntado para buscar la siguiente instrucción.

**Instrucciones de Transferencia a Memoria (2), Aritméticas(2) y Lógica:*

***LDAX A ! M[dir]; luego dir = la almacenada en el par de registros D−E,*

se mostrará en memoria como:

5000 h C.O. LDAX

...

E: XXXX h Dato Parte L (Byte Bajo)

D: XXXX h Dato Parte H (Byte Alto)

Las fases de ejecución son:

1ª Fase y 2º Fase (Común a todas).

3ª Fase:

Paso 1 ! Búsqueda del byte alto de la posición de memoria donde se ubica el operando (contenido del registro E) y llevarlo a la ruta de datos con lo que se guarda la parte baja (byte bajo) de la dirección en la parte L del registro HL: L ! E.

Paso 2 ! 4ª Fase.

4ª Fase:

Búsqueda del byte bajo de la posición de memoria donde se ubica el operando (contenido del registro D) y llevarlo a la ruta de datos con lo que se guarda la parte alta (byte alto) de la dirección en la parte H del registro HL: H ! D.

Paso 2 ! 4ª Fase.

5ªFase:

A ! M[HL]; en Ac meto el dato contenido en la posición de memoria que tengo en el registro HL y PC está apuntando a la posición de memoria 5001h y ahí está buscando la siguiente instrucción.

***STAX M[dir] ! A; luego dir = se almacenará en el par de registros D−E,*

se mostrará en memoria como:

5000 h C.O. STAX

...

E: XXXX h Dato Parte L (Byte Bajo)

D: XXXX h Dato Parte H (Byte Alto)

Las fases de ejecución son:

1ª Fase y 2º Fase (Común a todas).

3ª Fase:

Paso 1 ! Búsqueda del operando en la memoria (posición a la que apunta PC) y llevarlo a la ruta de datos con lo que se guarda la parte baja (byte bajo) de la dirección a cargar en Ac en la parte L del registro HL: L ! M[PC].

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

4ª Fase:

Paso 1 ! Búsqueda del operando en la memoria (posición a la que apunta PC) y llevarlo a la ruta de datos con lo que se guarda la parte alta (byte alto) de la dirección a cargar en Ac en la parte H del registro HL: H ! M[PC].

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

5ªFase:

M[HL] ! A; el Ac lo meto como contenido de la posición de memoria que tengo en el registro HL, 1000h, y PC está apuntando a la posición de memoria 5001h y ahí está buscando la siguiente instrucción.

***CMA A ! C1 (Ac); luego CMA es el C1 del Ac*

Las fases de ejecución son:

1ª Fase y 2º Fase (Común a todas).

3ª Fase:

Paso 1 ! Búsqueda del operando en el Ac.

Paso 2 ! A Temp va el complemento a uno del dato ubicado en el Ac.

4ª Fase:

Paso 1 ! A ! TEMP. Operado el dato del Ac y calculado su C1 en Temp, se ejecuta la operación que indica la instrucción así como al mismo tiempo se busca la siguiente instrucción a ejecutar.

Paso 2 ! Se incrementa el PC para el resto de la dirección por ser instrucción de tres bytes de tamaño en la memoria: PC ! PC + 1.

***INC r1*

Las fases de ejecución son:

1ª Fase (Común a todas):

Paso 1 ! Buscar la instrucción: RI ! M[PC], lo que apunta en memoria el PC me lo llevo a RI.

Paso 2 ! Preparación de la instrucción siguiente PC ! PC + 1.

2ª Fase (Común a todas):

Paso 1 ! Decodificar la instrucción.

Paso2 ! Evaluación del registro de estado.

3ª Fase: Búsqueda del operando 1 en Ac o registro correspondiente y llevarlo a la ruta de datos con lo que se guarda en el registro Temp.

4ª Fase: Se opera el dato de Temp incrementándolo en una unidad y así se ejecuta la operación que indica la instrucción: r1! TEMP + 1 así como al mismo tiempo se busca la siguiente instrucción a ejecutar por el PC.

***DER r1*

Las fases de ejecución son:

1ª Fase (Común a todas):

Paso 1 ! Buscar la instrucción: RI ! M[PC], lo que apunta en memoria el PC me lo llevo a RI.

Paso 2 ! Preparación de la instrucción siguiente PC ! PC + 1.

2ª Fase (Común a todas):

Paso 1 ! Decodificar la instrucción.

Paso2 ! Evaluación del registro de estado.

3ª Fase: Búsqueda del operando 1 en Ac o registro correspondiente y llevarlo a la ruta de datos con lo que se guarda en el registro Temp.

4ª Fase: Se opera el dato de Temp decrementándolo en una unidad y así se ejecuta la operación que indica la instrucción: r1! TEMP − 1 así como al mismo tiempo se busca la siguiente instrucción a ejecutar por el PC.

- ***Unidad de Control (U.C.).***

5.1. Identificación de las señales de control.

- ***Señales de control de salida para el banco de registros (B, C, D y E) y transferencia de sus datos.***

Para controlar con microórdenes el banco de registros necesito dos señales: SELreg1 y SELreg0 (selección registro); necesito a la vez una señal Sreg, salida registro, para la salida del dato del registro y otra Lreg, load registro, para la entrada del dato al registro.

Lreg

Sreg

SELreg1

SELreg0

En la siguiente tabla se muestran cada una de estas señales con su descripción correspondiente:

Señal	**Descripción**
	CONTROL DEL BANCO DE REGISTROS
SELreg1 SELreg0	Seleccionan el registro al que se desea acceder del banco de registros: B, C, D o E.
Lreg	Si se encuentra activada (valor 1), permite que el dato que se encuentra en el bus se almacene en el registro seleccionado (escritura).
Sreg	Si se encuentra activada (valor 1), permite que el dato que se encuentra en el registro seleccionado aparezca en el bus (lectura).

- ***Señales de control de salida para la memoria y la unidad aritmético y lógica (A.L.U.).***

Se ha separado la señal de lectura y escritura, L/E, en dos: Lmem y Emem. En MaNoTaS el multiplexor que selecciona el dato a escribir en el bus de direcciones para acceder a una posición de memoria tiene las combinaciones 00, 01 y 10 para los registros PC, SP y HL dejando la 11 sin utilizar.

En cuanto a la entrada de la unidad aritmético y lógica, A.L.U., tiene tres señales ALU2, ALU1 y ALU0 con las que podremos hacer ocho combinaciones de entrada con esos tres bits del bus de entrada que proporcionan estas señales y así poder controlar la operación a realizar mediante distintas microórdenes (se utilizarán 7 combinaciones y quedando 1 libre).

Para acabar mencionar Salu como salida de la unidad aritmético y lógica que si se pone con valor 1 permite que el resultado aparezca en el bus de datos, sino el resultado de la operación NO irá al bus de datos al no estar activada la señal.

En la siguiente tabla se muestran cada una de estas señales con su descripción correspondiente:

Señal	**Descripción**
	CONTROL DE LA MEMORIA
Lmem	Señal de lectura de la memoria. Si su valor es 1, el dato almacenado en la posición de memoria que hay en el bus de direcciones se coloca en el bus de datos.
Emem	Señal de escritura de la memoria. Si su valor es 1, el dato que se encuentra en el bus de datos se almacena en la posición de memoria que hay en bus de direcciones.
SDir2 SDir1	Señales de control al multiplexor que selecciona el dato a escribir en el bus de direcciones para acceder a una posición de memoria. Dependiendo de su valor se accederá a la posición de memoria especificada por el registro PC, SP o HL.
	CONTROL DE LA UNIDAD ARITMETICA Y LOGICA
ALU2 ALU1 ALU0	Estas tres señales de control determinan la operación a realizar por la A.L.U.: suma, resta, and, or, or exclusiva, incremento y decremento.
Salu	Si se encuentra activada permite que el resultado de la A.L.U. aparezca en el bus de datos.

- ***Señales de control de salida para el registro de estado.***

En la siguiente tabla se muestran cada una de estas señales con su descripción correspondiente:

Señal	**Descripción**
	CONTROL DEL REGISTRO DE ESTADO
LF	Si se encuentra activa, carga los datos de la entrada en el registro de estado. El dato puede provenir de la A.L.U. o del registro acumulador.
SF	Si se encuentra activada, escribe en el bus de datos el contenido del registro de estado.
SelO	Señal de control a un multiplexor que selecciona como dato a escribir en el registro de estado como flag de Overflow, el bit 2 del registro acumulador o el indicador de Overflow procedente de la A.L.U..
SelC	Señal de control a un multiplexor que selecciona como dato a escribir en el registro de estado como flag de Carry, el bit 1 del registro acumulador o el indicador de Carry procedente de la A.L.U..
SelZ	

	Señal de control a un multiplexor que selecciona como dato a escribir en el registro de estado como flag de Cero (o Zero flag), el bit 0 del registro acumulador o el indicador de Cero procedente de la A.L.U..

- ***Señales de control de salida para los registros y señales de control de entrada.***

En las siguientes tablas se muestran cada una de las señales tanto de control de entrada como las de control de salida para los registros con su descripción correspondiente:

Señal	**Descripción**
	CONTROL DE ENTRADAS
Z	Representa el indicador Z generado por la unidad aritmético y lógica (A.L.U.).
COP	Código de operación, es el contenido del registro de instrucción (RI).

Señal	**Descripción**
	CONTROL DE REGISTROS
Lri	Si está activa, el dato que se encuentra en el bus de datos se guardará en el registro de instrucción.
Lpc	Si está activa, ordena la carga del registro Contador de Programa con el dato que se encuentra a su entrada y el PC se incrementa en una unidad.
LspL	Si está activa, ordena la carga de la parte baja del registro SP con el dato que se encuentra en el bus de datos.
LspH	Si está activa, ordena la carga de la parte alta del registro SP con el dato que se encuentra en el bus de datos.
Isp	Si está activa, incrementa el contenido del registro SP en una unidad.
Dsp	Si está activa, decrementa el contenido del registro SP en una unidad.
LdirL	Si está activa, ordena la carga de la parte baja del registro HL con el dato que se encuentra en el bus de datos.
LdirH	Si está activa, ordena la carga de la parte alta del registro HL con el dato que se encuentra en el bus de datos.
Lac	Si está activa, el dato que se encuentra en el bus de datos se guardará en el registro acumulador.
Sac	Si está activa, el contenido del registro acumulador aparecerá en el bus de datos
Ltemp	Si está activa, el dato que se encuentra en el bus de datos, se guardará en el registro temporal que se encuentra en la segunda entrada de la A.L.U..

- ***Señales de control a los multiplexores, banco de registros y la A.L.U..***

En las siguientes tablas se muestran cada una de las señales tanto de control de entrada como las de control de salida para los registros con su descripción correspondiente:

SELreg1	**SELreg0**	**Registro**
0	0	B
0	1	C
1	0	D
1	1	E

Sdir2	Sdir1	Dirección
0	0	PC
0	1	SP
1	0	HL
1	1	No utilizada

ALU2	ALU1	ALU0	Operación
0	0	0	Suma
0	0	1	Resta
0	1	0	And
0	1	1	Or
1	0	0	Xor
1	0	1	Incremento
1	1	0	Decremento
1	1	1	No utilizada

SelO	Acción	SelC	Acción	SelZ	Acción
0	O!ALUO	0	C!ALUC	0	Z!ALUZ
1	O!A2	1	C!A1	1	Z!A0

5.2. La ruta de datos y control.

A partir del organigrama simplificado de las fases de ejecución de las instrucciones podemos diseñar la siguiente ruta de datos:

A.L.U.

5.3. Activación de las señales de control.

A continuación mostraremos una tabla en la cual se mostrarán las señales de control que se activan durante cada fase de ejecución de las distintas instrucciones del repertorio de nuestra máquina no tan sencilla, MaNoTaS, y teniendo en cuenta que no se considerarán las siguientes instrucciones: *ADD A, SUB A, ANA A y ORA A.*

Fases	Operación	Activación de señales
FASE 1		
	RI ! M[PC] PC ! PC + 1	Sdir2, Sdir1 (= 00), Lmem, Lpc, Lri
FASE 2		
	Decodificación y evaluación de Z	
FASE 3		Depende de la instrucción a ejecutar:
ADD, SUB, ANA, ORA	TEMP ! r1	SELreg1, SELreg0 (= r1), Sreg, Ltemp
ADI, SUI, ANI, ORI	TEMP ! M[PC] PC ! PC + 1	Sdir2, Sdir1 (= 00), Lmem, Lpc, Ltemp

LDA, STA, JMP, JZ	L ! M[PC] PC ! PC + 1	Sdir2, Sdir1 (= 00), Lmem, Lpc, LdirL
FASE 4		
Aritmético − Lógicas	A ! A op TEMP	ALU2, ALU1, ALU0 (= operación), Salu, Lac
Transferencia y salto	H ! M[PC] PC ! PC + 1	Sdir2, Sdir1 (= 00), Lmem, Lpc, LdirH
FASE 5		
LDA	A ! M[HL]	Sdir2, Sdir1 (= 10), Lmem, Lac
STA	M[HL] ! A	Sdir2, Sdir1 (= 10), Emem, Sac
JMP, JZ	PC ! HL + 1	Sdir2, Sdir1 (= 10), Lpc

5.4. Ejemplo: ejecución de LDA.

Hemos escogido como ejemplo de instrucción a ejecutar fase a fase y viendo qué señales de la U.C. se activan la instrucción *LDA*; las fases de ejecución serían las siguientes:

- ***Fase 1: Búsqueda de la instrucción.***

Consta de dos pasos:

- RI ! M[PC]
- PC ! PC + 1

- ***Fase 2: Decodificación.***

Tanto esta fase como la anterior son comunes a todas las instrucciones.

- ***Fase 3: Obtención de operandos y evaluación del código de condición Z.***

- L ! M[PC] (L ! DirL)
- PC ! PC + 1

- ***Fase 4: Obtención de operandos, ejecución y conclusión de las instrucciones Aritmético − Lógicas.***

- H ! M[PC] (H ! DirH)
- PC ! PC + 1

- ***Fase 5: Conclusión de las instrucciones de acceso a memoria y salto incondicional y obtención de operandos.***

- Instrucción de carga (para nuestro caso *LDA*).

- A ! M[HL]

- Instrucción de almacenamiento (si fuera *STA*).

- M[HL] ! A

En cuanto a cómo se vería cada una de las fases de la instrucción *LDA* una a una en la ruta de datos y control se muestran en las siguientes figuras (los flujos de información se muestran en azul) a continuación:

- ***Fase 1: Búsqueda de la instrucción.***

Consta de dos pasos:

- RI ! M[PC]
- PC ! PC +1

A.L.U.

- ***Fase 2: Decodificación.***

Tanto esta fase como la anterior son comunes a todas las instrucciones. Aquí se decodifica la instrucción y se coge el código de operación que tiene : *70 h*.

A.L.U.

- ***Fase 3: Obtención de operandos y evaluación del código de condición Z.***

- L ! M[PC] (L ! DirL)
- PC ! PC +1

A.L.U.

- ***Fase 4: Obtención de operandos, ejecución y conclusión de las instrucciones Aritmético − Lógicas.***

- H ! M[PC] (H ! DirH)
- PC ! PC +1

A.L.U.

- ***Fase 5: Conclusión de las instrucciones de acceso a memoria y salto incondicional y obtención de operandos.***

- Instrucción de carga.

- A ! M[HL]

A.L.U.

Si el seguimiento de la activación de las señales de control lo realizáramos mediante un reloj, éste nos proporcionaría un cronograma en el que se nos mostrará el valor de cada una de las distintas señales así como el valor del reloj en cada ciclo o pulso que realizara. Ahora veremos cómo queda el cronograma perteneciente a la ejecución de la instrucción de nuestro ejemplo:

- ***Fase 1:***

Leo del PC y cargo en RI (antes tiene basura) por flanco de subida e incremento el PC por flanco de bajada, el resto de la U.C. a cero o alta impedancia (Z) para que no pase nada.

- ***Fase 2:***

Decodifico y evalúo el registro de estado, dura 1 ciclo y todas las señales están en Z.

- ***Fase 3:***

Búsqueda de operandos por lo que por flanco de subida cargo en parte baja del registro HL, es decir, en L y a la vez por flanco de bajada el PC se ve incrementado en una unidad.

- ***Fase 4:***

Búsqueda de operandos por lo que por flanco de subida cargo en parte alta del registro HL, es decir, en H y a la vez al acabar el pulso y bajar el flanco el PC se ve incrementado en una unidad.

- ***Fase 5:***

Ejecuta la instrucción: leer lo que apunta HL (Sdir2,1 = 10 para habilitarlo) y dejar en Ac.

El cronograma quedaría como sigue:

Fase 1 Fase 2 Fase 3 Fase 4 Fase 5

Reloj

Sdir2

Sdir1

Lmem

Lri

Lpc

LdirL

LdirH

Lac

Fase 1 Fase 2 Fase 3 Fase 4 Fase 5

IMPORTANTE

Nota 5, sobre las señales de control: Cada señal de control, del multiplexor por ejemplo, se puede representar de otra forma:

Una va como complemento de otra, es decir, por nivel alto una y por bajo la otra, así 102 = 2

0 0 2

- ***Diseño de la Unidad de Control (U.C.).***

6.1. Control Cableado.

Las características de este control son:

- Implementación en hardware, por lo tanto muy rápido.
- No es flexible: una modificación posterior implica cambiar el circuito entero.
- Con este control la U.C. se diseña para una determinada ruta de datos.

6.2. Control Microprogramado.

Las características de este control son:

- Representación programada para el control.
- Más lento, al tener que acceder a la memoria de control.
- Flexible, permite cambios posteriores sin tener que cambiar el circuito entero.
- Con este control la U.C. se modificará fácilmente.

6.3. Dos métodos para el diseño de la Unidad de Control Cableada.

Existen dos formas de diseñar la U.C. Cableada (que explicaremos más adelante):

- Método de la tabla de estados.
- Método del contador de secuencia.

6.3.1. Método de la tabla de estados.

Las características de este método son:

- Basada en una máquina de estados finitos.
- Una máquina de estados finitos consta de:

- Memoria interna que contiene el estado. Cada estado corresponde a un ciclo de reloj y contiene las operaciones a realizar en ese ciclo.

- Dos funciones combinacionales:

- La función de estado siguiente: es una función combinacional que a partir de las entradas y el estado actual determina el estado siguiente, es decir, me dice a qué estado paso dependiendo del estado en el que me encuentre.

- La función de salida: produce el conjunto de señales de control a partir de sus entradas y el estado actual. En la figura siguiente se mostrará la f(s) ! función de salida " I ! microinstrucciones aplicadas a cada registro de tal manera que permita que los datos circulen bien por la ruta de datos:

R.E.

4 f (s) " I

8

C.O. (R.I.)

Este método se basa en una máquina de estados finitos representada en un grafo de estados en cual se representan las distintas fases de ejecución, cuatro o cinco dependiendo del tipo de instrucción que se trate en cada instante, de las siguientes instrucciones: *SUB, ANA, ORA, ADD, ADI, SUI, ANI, ORI, LDA, STA, JZ y JMP* con sus Cód. Op. : *18 h, 20 h, 24 h, 30 h, 35 h, 36 h, 68 h, 69 h, 70 h, 71 h, 72h y 74 h.*

La situación de la máquina en cada estado se puede resumir de la siguiente manera:

- ***Estado 0: Es el primero por el que se tiene que pasar y tiene asociada una función de salida S0; dicho estado se corresponde con la Fase 1. Dependiendo de las entradas iré a un nodo o a otro. Z = X independientemente del R.E..***
- ***Estado 1: Distingo el tipo de las operaciones que quiero realizar: +, −, or y and.***
- ***Estado 2, 3: Me voy al nodo para hacer la operación específica. Los operandos están en el A y Temp.***
- ***Estado 4 .. 7: En cada nodo de este estado se encuentran las operaciones: +, −, or y and; en cada nodo se hará las operaciones inmediatas o no dependiendo de la búsqueda de operandos en el estado 2. Normalmente se corresponde con el último estado que se identifica con la Fase 4 de la ejecución de la instrucción y que, en ningún caso, se corresponde con una instrucción de tipo salto.***
- ***Estado 8, 9: Cargo L y H leyendo una vez en memoria en cada estado y consecutivamente. Cuando el flag Z = 1 estas dos operaciones hacen lo mismo.***
- ***Estado 10 .. 12: Me voy al nodo para hacer el salto según el salto que indique la instrucción que se esté ejecutando; este nodo al que llego se corresponde con la Fase 5 y última de las instrucciones de tipo salto.***

La **máquina de estados** quedaría representada por el **grafo** siguiente (de los estados S2 a S12 tenemos que Z = X siempre):

CO = XXXX, Z = X CO = 72h, Z = 0

30h, 18h, 20h, 24h, Z = X 70h, 71h, 72h, 74h, Z = 1

35h, 36h, 68h, 69h, Z = X

35h

24h

s 20h 36h

30h 18h 68h 69h XXh

70h 71h 72h,74h

Para seguir implementando la U.C. Cableada por el Método de la tabla de estados veremos la función de salida que tiene las siguientes características:

- Existe una función de salida para cada estado.
- Tengo trece estados en total para codificar.
- Cada estado corresponde a una fase, por lo que debo generar sus señales. Estas señales para cada fase ya las hemos visto en los cronogramas, pero para implementar la función de salida se hace necesario transformar los cronogramas en una tabla con números de tal manera que los estados componen las fases de un cronograma.

La **función de salida** para cada estado quedaría representada por la **tabla** mostrada a continuación (aunque es válida para todas las instrucciones se ha de mencionar que por instrucciones Aritmético − Lógicas se consideran únicamente: *ADD B, SUB B, ANA B* y *ORA B*):

Función Salida	**Estados**												
Señales de Control	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Lri	1	0	0	0	0	0	0	0	0	0	0	0	0
Sdir2,1	00	XX	XX	00	XX	XX	XX	XX	00	00	10	10	10
Lmem	1	0	0	1	0	0	0	0	1	1	1	0	0
Emem	0	0	0	0	0	0	0	0	0	0	0	1	0
Lpc	1	0	0	1	0	0	0	0	1	1	0	0	1
LspL	0	0	0	0	0	0	0	0	0	0	0	0	0
LspH	0	0	0	0	0	0	0	0	0	0	0	0	0
Isp	0	0	0	0	0	0	0	0	0	0	0	0	0
Dsp	0	0	0	0	0	0	0	0	0	0	0	0	0
LdirL	0	0	0	0	0	0	0	0	1	0	0	0	0
LdirH	0	0	0	0	0	0	0	0	0	1	0	0	0
SELreg1,0	XX	XX	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
Lreg	0	0	0	0	0	0	0	0	0	0	0	0	0
Sreg	0	0	1	0	0	0	0	0	0	0	0	0	0
Lac	0	0	0	0	1	1	1	1	0	0	1	0	0
Sac	0	0	0	0	0	0	0	0	0	0	0	1	0
Ltemp	0	0	1	1	0	0	0	0	0	0	0	0	0
ALU2,1,0	XXX	XXX	XXX	XXX	000	001	010	011	XXX	XXX	XXX	XXX	XXX
Salu	0	0	0	0	1	1	1	1	0	0	0	0	0
SelO	X	X	X	X	0	0	0	0	X	X	X	X	X
SelC	X	X	X	X	0	0	0	0	X	X	X	X	X
SelZ	X	X	X	X	0	0	0	0	X	X	X	X	X
LF	0	0	0	0	1	1	1	1	0	0	0	0	0
SF	0	0	0	0	0	0	0	0	0	0	0	0	0

El siguiente paso en la implementación por este método es ver la función de estado siguiente en la que se codifican los estados sin seguir ningún criterio a la hora de realizar esta codificación, por ejemplo:

La **función de estado siguiente** para cada estado quedaría representada por la **tabla** siguiente:

	ESTADO ACTUAL				**ESTADO SIGUIENTE**			
INSTRUCCION	**E 3**	**E 2**	**E 1**	**E 0**	**PE 3**	**PE 2**	**PE 1**	**PE 0**
	0	0	0	0	0	0	0	1
72h,0 (JZ)	0	0	0	1	0	0	0	0
30h,18h,20h,24h, X (ADD,SUB,ANA,ORA)	0	0	0	1	0	0	1	0
35 h,36 h,68 h,69 h, X (ADI,SUI,ANI ORI)	0	0	0	1	0	0	1	1
70h,71h,74h,X(LDA,STA,JMP); 72h,1	0	0	0	1	1	0	0	0

(JZ)								
30 h, X (ADD)	0	0	1	0	0	1	0	0
18 h, X (SUB)	0	0	1	0	0	1	0	1
20 h, X (ANA)	0	0	1	0	0	1	1	0
24 h, X (ORA)	0	0	1	0	0	1	1	1
35 h, X (ADI)	0	0	1	1	0	1	0	0
36 h, X (SUI)	0	0	1	1	0	1	0	1
68 h, X (ANI)	0	0	1	1	0	1	1	0
69 h, X (ORI)	0	0	1	1	0	1	1	1
	0	1	0	0	0	0	0	0
	0	1	0	1	0	0	0	0
	0	1	1	0	0	0	0	0
	0	1	1	1	0	0	0	0
	1	0	0	0	1	0	0	1
70 h, X (LDA)	1	0	0	1	1	0	1	0
71 h, X (STA)	1	0	0	1	1	0	1	1
72 h, 74 h, X (JMP, JZ)	1	0	0	1	1	1	0	0
	1	0	1	0	0	0	0	0
	1	0	1	1	0	0	0	0
	1	1	0	0	0	0	0	0

La **implementación de la U.C.** tendría las siguientes características:

- U.C. construida como una máquina de estados finitos.
- Circuito combinacional que podría implementarse mediante una ROM o una PLA.
- U.C. con doce entradas, lo que implica que se puede representar a la entrada un total de 212 = 4 K posibles combinaciones.
- U.C. con 28 señales de salida

Se puede implementar **con una memoria ROM**, pero necesito saber el número de E/S. Para ello sabemos que tenemos 8 bits para el C.O. y el flag Z así como 4 bits para el R.E. y otros tantos para el Estado Actual, con lo que necesitamos 16 bits para las entradas y, por consiguiente, 216 posiciones de memoria, es decir, desde la posición de memoria 0000 h hasta la FFFF h que conforman 64 Kb. En cuanto al número de salidas sabemos que las señales de control requieren 28 bits y el Estado Siguiente 4 más, por lo que necesitaremos 32 bits para las salidas. Luego el bloque de memoria ROM para la U.C. sería como el mostrado en el siguiente esquema:

32 bits

0000 h

0001 h

.

.

.

FFFE h

FFFF h

En cuanto al **esquema** por **bloques** de la **implementación de la U.C.** por **Método de la tabla de estados** es el que vemos en la figura:

4 bits

Ahora bien ¿cómo se rellena la memoria? Se puede ver fácilmente en la siguiente tabla donde hemos representado las entradas con 16 bits Ai y las salidas con 32 bits Bi:

Reg. Entr. o Flag			**Código de Operación**							**Estado Actual**				**Líneas de Control**				**Estado Siguiente**		
A15	**..**	**A12**	**A11**	**A10**	**A9**	**..**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**B31**	**B30**	**..**	**B4**	**B3**	**B2**	**B1**