ORGANIZACIÓN DE LA UNIDAD CENTRAL DE PROCESO (CPU)

1 Introducción

En este capítulo veremos un posible diseño interno de una unidad central de proceso de una arquitectura de von Neumann, incluyendo sus variaciones de "lógica cableada" y "lógica microprogramada".

Presentaremos como ejemplo la organización del procesador de la arquitectura MIC-1, desarrollada por Andrew S. Tanenbaum con fines didácticos en su libro *Structured Computer Organization* (Organización de Computadoras, un Enfoque Estructurado), en particular para su 3ra Edición.

La **Unidad Central de Proceso** es la encargada de la ejecución de las **instrucciones**.

Para ello la arquitectura de von Neumann propone que la misma cuente con los siguientes elementos básicos:

- **Unidad Aritmética-Lógica (ALU)**: circuito lógico que implementa operaciones de aritmética binaria (típicamente la operaciones básicas para representaciones "binario" y "complemento a 2" y lógicas (típicamente AND, OR, EXOR y NOT bit a bit). También es habitual que implemente operaciones de desplazamiento y/o rotación de bits.
- **Unidad de Control**: circuito secuencial que implementa el denominado "ciclo de instrucción", permitiendo acceder a la siguiente instrucción de un programa, leer sus operandos, efectuar la operación indicada en la ALU y guardar el resultado de la misma.
- **Banco (Set / Conjunto) de Registros**: una serie de posiciones especiales de memoria, ubicadas físicamente dentro de la propia CPU, que permiten un acceso a operandos y lugares de almacenamiento de resultados mucho más veloz que si estuvieran en el sistema de memoria normal. Algunos de estos registros son de uso interno de la propia CPU (más precisamente de su Unidad de Control) y otros son accesibles y utilizables por el programador.

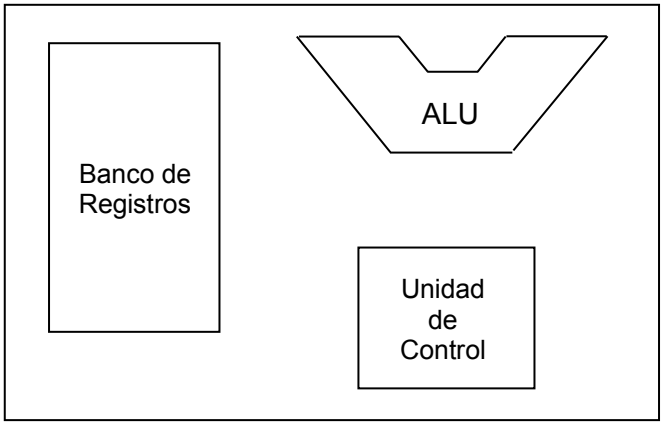
Notemos que el Banco (Set / Conjunto) de Registros no es un elemento esencial de la arquitectura. Es un elemento de carácter tecnológico que tiene que ver con dos aspectos:

- por razones vinculadas a la electrónica, la velocidad de transferencia de información entre dos circuitos está limitada por la distancia geográfica entre los mismos
- la memoria rápida (usada en los pocos registros de la CPU) es más cara que la memoria más lenta (usada en las mucho más abundantes posiciones del sistema de memoria principal).

De allí que se utilice poca memoria rápida (por el costo) y que se ubique en las cercanías de donde se va a utilizar (dentro de la CPU) para minimizar su tiempo de acceso.

2 Componentes Básicos de la CPU

Como ya dijimos antes una CPU (Centrol Processor Unit) está conformada por tres sub-sistemas fundamentales: la ALU (Arithmetic Logic Unit, Unidad Aritmética y Lógica), la CU (Control Unit, Unidad de Control) y el Register Set (también denominado Register Bank) ó sea Conjunto (o Banco) de Registros.



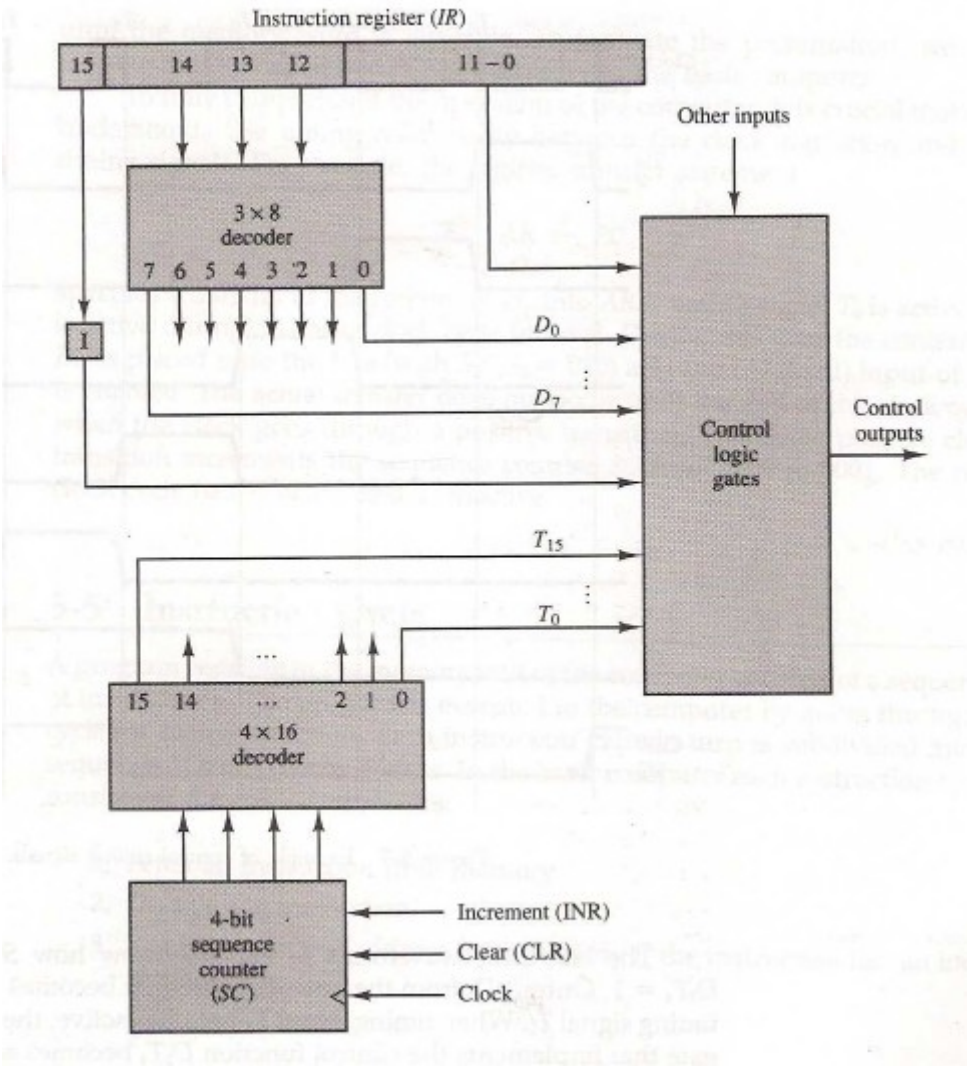
El **Banco de Registros** contiene registros de tres categorías desde el punto de vista de su función en relación con los programas y el funcionamiento interno de la CPU:

- Totalmente visibles: estos son los ya mencionados registros de uso general ó personalizados que contienen operandos o direcciones para su utilización en las instrucciones. El programador de "bajo nivel" los manipula directamente en los programas.
- Parcialmente visibles: son registros que tienen funciones especiales pero participan de algún modo indirecto en las instrucciones. El programador los manipula indirectamente en determinadas instrucciones específicas. Ejemplos de este tipo de registros son el **IP** (**I**nstruction **P**ointer), también denominado **PC** (**P**rogram **C**ounter) que contiene la dirección de la próxima instrucción a ejecutarse (en algunas arquitecturas almacena la dirección de la que se está ejecutando en este momento), el **SP** (**S**tack **P**ointer) que contiene el puntero al primer lugar de la pila en las arquitecturas "de stack" y el **PS** (**P**rocessor **S**tatus) también denominado registro de **FLAGS** (en el caso de Intel) que contiene el estado del procesador incluyendo el valor que tomaron los bits de condición (**N**egative, **Z**ero, **C**arry, **O**verflow) en función del resultado de la última operación realizada por la ALU.
- Internos: son registros que utiliza la Unidad de Control de la CPU para poder ejecutar las instrucciones. Almacenan constantes, el estado de la CU, la instrucción en ejecución (su código binario), resultados intermedios de cálculos de direcciones, etc. No son visibles de ninguna manera al programador.

La **Unidad de Control** es, en definitiva, una máquina secuencial que realiza el "ciclo de instrucción": conjunto de acciones ordenado y secuencial que interconectan adecuadamente los distintos elementos en el tiempo, para lograr el objetivo de ejecutar la instrucción realizando la operación indicada sobre los operandos correspondientes y almacenando el resultado en el lugar indicado. Esta máquina secuencial funciona sincronizada por un reloj, el cual también es utilizado para sincronizar todas las actividades de los otros elementos del sistema (memoria y entrada/salida). En las primeras computadoras el reloj era el mismo para todos los elementos. Últimamente se utilizan relojes independientes (aunque vinculados) para cada sub-sistema. En muchos diseños se utilizan más de un reloj para la CPU, con la misma frecuencia, pero desfasados (0º, 90º, 180º y 270º, por ejemplo) a los efectos de ser utilizados para sincronizar distintas partes del circuito compensando los diferentes retardos de propagación de las señales en los circuitos internos de la CPU.

En la figura siguiente se muestra un ejemplo de organización de una unidad de control (tomado de la "computadora básica" de Arquitectura de Computadoras de Morris Mano). Esta formado por dos decodificadores, un contador secuencial, y cierta cantidad de

compuertas lógicas de control. Una instrucción leída de la memoria se coloca en el registro de la instrucción actual (IR). Se utiliza un decodificador para decodificar el código de operación y enviar las señales de control las compuertas lógicas. Los bits de los operandos se conectan directamente a las compuertas lógicas. Finalmente, se utiliza un decodificador para las señales del contador, cuya salida se utilizará para habilitar las distintas funciones implementadas en las compuertas lógicas de control.



La **Unidad Aritmética y Lógica** es un conjunto de circuitos (típicamente combinatorios) que implementan un conjunto de operaciones, que incluyen suma y resta (en aritmética complemento a 2), operaciones lógicas bit a bit (AND, OR, EXOR, NOT) y operaciones de desplazamiento (shift). Las ALUs más avanzadas incluyen operaciones de multiplicación y división (aunque en este caso se implementan como una máquina secuencial que implementa algún algoritmo para estas operaciones).

3 Ciclo de Instrucción

Se denomina ciclo de instrucción a la secuencia de acciones que realiza la CPU (más específicamente la Unidad de Control) para lograr ejecutar una instrucción del programa almacenado en memoria.

Un ciclo de instrucción típico tiene 5 pasos característicos:

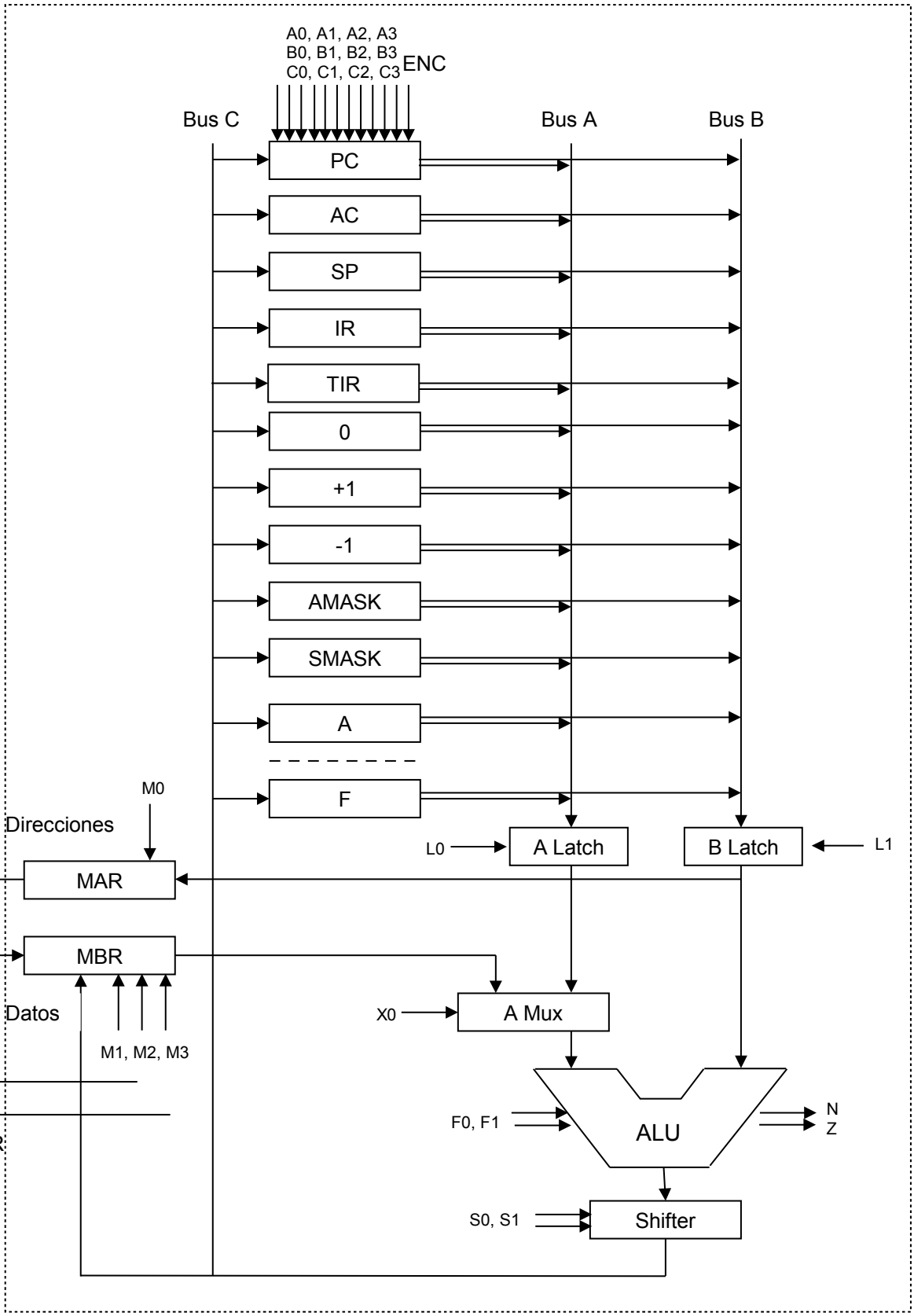
- **Fetch**: este paso consiste en leer la próxima instrucción a ejecutarse desde la memoria.
- **Decode**: en este paso se analiza el código binario de la instrucción para determinar qué se debe realizar (cuál operación, con qué operandos y donde guardar el resultado)
- **Read**: en este paso se accede a memoria para traer los operandos
- **Execute**: es la ejecución de la operación por parte de la ALU sobre los operandos
- **Write**: en el último paso se escribe el resultado en el destino indicado en la instrucción.

Notemos que no todas las instrucciones requieren de todos los pasos indicados para su ejecución. Por ejemplo las instrucciones que tienen sus operandos en registros, no requieren del paso “read”, mientras que las que no guardan un resultado no requieren del paso “write”.

Para realizar estos pasos la unidad de control maneja un conjunto de señales de control interno a la CPU y otras de control externo (en la interfaz con la memoria y con el sistema de entrada/salida), las que habilitan las conexiones apropiadas entre los distintos elementos de forma que el respectivo paso del ciclo se cumpla. Por ejemplo: para hacer el “fetch” conecta el bus de direcciones de memoria con el registro PC (que contiene la dirección de la próxima instrucción a ejecutarse), conecta el bus de datos de memoria con el registro interno IR (Instruction Register) y coloca en el bus de control de la memoria las señales apropiadas para realizar una lectura.

4 Estructura de la CPU

En el siguiente diagrama se muestra la estructura de una CPU MIC-1.



En esta propuesta de organización interna se dispone de los siguientes recursos (es una arquitectura de 16 bits):

- Registro PC: contiene el puntero de instrucción (“Program Counter”)
- Registro AC: es el registro ACumulador
- Registro SP: contiene el puntero a la pila (“Stack Pointer”)
- Registro IR: almacena la instrucción leída desde memoria
- Registro TIR: almacena temporalmente copias de la instrucción
- Registro 0: contiene la constante 0
- Regsitro +1: contiene la constante 1 (positiva)
- Registro -1: contiene la constante -1 (negativa en complemento a 2)
- Registros AMASK: contiene el valor binario 0000111111111111 (0x0fff)
- Registros SMASK: 000000001111111 (0x00ff)
- Registros A a F: para uso interno de la Unidad de Control
- Registro MAR (Memory Address Register): almacena la dirección de memoria que se presenta en el bus de direcciones de la memoria durante una operación de lectura o escritura de la misma.
- Registro MBR (Memory Buffer Register): almacena el dato leído de la memoria (en una operación de lectura) o el dato a escribir en la memoria (durante una operación de escritura).
- AMux: multiplexor que elige entre dos entradas posibles.
- ALU: Unidad Aritmética y Lógica, capaz de ejecutar 4 operaciones: suma A+B, A and B identidad A (la salida es igual a su entrada A), not A.
- Shifter: unidad encargada de realizar la operación de desplazamiento: 0 bit (no desplaza nada), 1 bit a la derecha, 1 bit a la izquierda.
- ALatch/BLatch: son registros intermedios utilizados para simplificar los aspectos de sincronismo de las señales, requeridos porque el diseño original de Tanenbaum prevé la construcción del banco de registros con Flip-Flops asíncronos (un diseño basado en FF con reloj por flanco podría evitar el uso de estos registros intermedios).

Notas:
- La interfaz con la memoria y la E/S se realiza a través del bus de datos, el bus de direcciones y el bus de control (formado por las señales RD y WR).
- Los registros, la ALU y la unidad de Shift son de 16 bits de tamaño de palabra.
- La unidad de desplazamiento no siempre se explicita como en el ejemplo ya que se puede considerar que forma parte de la ALU.
- Normalmente las ALUs calculan también las salidas C (Carry) y V (oVerflow).
- Los valores de las constantes de los distintos registros está vinculada con las necesidades de la “macroarquitectura” asociada a la MIC-1 y no constituyen un requisito en otras microarquitecturas.
- Tener presente que este es tan solo un ejemplo de cómo se puede construir internamente una CPU y ni siquiera es uno que pueda considerarse óptimo.

Por razones de simplicidad se ha omitido dibujar la Unidad de Control en el diagrama anterior. Esta es, como dijimos, una máquina secuencial. Las salidas de este circuito secuencial son todas las señales de control indicadas en el diagrama, que actúan sobre los distintos recursos internos y el bus de control externo de manera de completar los distintos pasos del ciclo de instrucción visto.

Las funciones que cumplirían las distintas señales de control son:

- A3, A2, A1, A0: seleccionan un registro y conectan su salida al bus A (ej: 0000/PC, 0001/AC, 0010/SP, 0010/IR, 0100/TIR, 0101/0, 0110/+1, 0111/-1, 1000/AMASK, 1001/SMASK, 1010/A, 1011/B … 1111/F)
- B3, B2, B1, B0: seleccionan un registro y conectan su salida al bus B (ej: idem a la codificación para A)
- C3, C2, C1, C0: seleccionan un registro y conectan su entrada al bus C (ej: idem a la codificación para A)
- ENC: habilita el bus C para que se guarde el valor en el registro seleccionado
- X0: selecciona si conecta a su salida la entrada que viene desde el MBR o desde el bus A (ej: 0 – MBR, 1 – Bus A)
- F1, F0: codifican la operación a realizar por la ALU (ej: 00 – A+B, 01 – A and B, 10 – A, 11 – not A)
- S1, S0: codifican la operación de desplazamiento (ej: 00 – no desplaza, 01 – un bit a la derecha, 10 – un bit a la izquierda)
- L1, L0: controlan la carga de los registros intermedios del bus B y el bus A respectivamente.
- M0: controla la carga en el MAR (ej: 1 – carga)
- M1, M2, M3: controlan la forma que se cargan los datos en el MBR, cuando y desde donde se produce. M1 activa la carga (ej: 1 – carga). M2 es la señal de lectura de memoria RD (ej: 1 – lee la memoria). M3 es la señal de escritura de memoria (eg: 1 – escribe la memorias).

Veamos como manejaría la Unidad de Control estas señales de control para lograr ejecutar el ciclo de instrucción. Consideremos el ejemplo en el que se va a ejecutar una suma entre un operando en memoria y el acumulador, con modo de direccionamiento directo (la dirección está en los últimos 12 bits de la instrucción).

El primer paso es el *fetch*. Para leer la instrucción de memoria se debe cargar el contenido del registro PC en el MAR, realizar una operación de lectura desde memoria y se debe conectar el bus de datos con el registro IR, de forma de cargar en él la instrucción. Para ello hay que:

- conectar la salida del registro PC al bus B (B3 = 0, B2 = 0, B1 = 0, B0 = 0)
- seleccionar cargar el BLatch (L0 = 1)
- seleccionar cargar MAR (M0 = 1)
- seleccionar cargar el MBR (M1 = 1)
- seleccionar lectura de memoria (M2 = 1)
- seleccionar MBR en el AMux (X0 = 0)
- seleccionar la operación identidad-A en la ALU (F1 = 1, F0 = 0)
- seleccionar no desplazamiento en el Shifter (S0 = 0, S1 = 0)
- conectar la entrada del registro IR al bus C (C3 = 0, C2 = 0, C1 = 1, C0 = 0)
- habilitar el bus C (ENC = 1)

Al colocar todas estas señales en los valores indicados se procederá a la lectura de la instrucción contenida en la posición PC de la memoria. Notemos que de acuerdo a los tiempos de acceso de la memoria se pueden llegar a necesitar mantener las señales por más de un ciclo de reloj a los efectos de tener en cuenta las latencias (tiempos de propagación) de la circuitería involucrada.

Para completar el paso de *fetch* es necesario actualizar el valor de PC, para lo que se requiere las siguientes señales (en el siguiente ciclo de reloj):

- conectar la salida del registro PC al bus B (B3 = 0, B2 = 0, B1 = 0, B0 = 0)
- conectar la salida del registro +1 al bus A (A3 = 0, A2 = 1, A1 = 1, A0 = 0)
- seleccionar cargar el ALatch (L1 = 1)

- seleccionar cargar el BLatch (L0 = 1)
- seleccionar Bus A en el A Mux (X0 = 1)
- seleccionar A + B en la ALU (F1 = 0, F0 = 0)
- seleccionar no desplazamiento en el Shifter (S0 = 0, S1 = 0)
- conectar la entrada del registro PC al bus C (C3 = 0, C2 = 0, C1 = 0, C0 = 0)
- habilitar bus C (ENC = 1)

El siguiente paso del ciclo de instrucción es el *decode*. La decodificación se realiza analizando el código binario de la instrucción. Para ello se implementa en la CU una función mediante un circuito combinatorio que toma como entrada el contenido del IR y da como salida los valores apropiados de las señales de control, condicionando también el próximo estado de la máquina secuencial, como en el caso del ejemplo ya que al reconocer el modo de direccionamiento directo debe realizarse el paso de *read*.

Para realizar el paso de *read* se debe cargar la dirección del operando (contenida en los últimos 12 bits de la instrucción) en el MAR, para luego realizar una operación de lectura desde memoria, cargando el MBR desde el bus de datos, de forma de almacenar en él el operando. Para ello hay que:

- conectar la salida del registro IR al bus B (B3 = 0, B2 = 0, B1 = 1, B0 = 0)
- conectar la salida del registro AMASK al bus A (A3 = 1, A2 = 0, A1 = 0, A0 = 0)
- seleccionar cargar el ALatch (L1 = 1)
- seleccionar cargar el BLatch (L0 = 1)
- seleccionar Bus A en el A Mux (X0 = 1)
- seleccionar A and B en la ALU (F1 = 0, F0 = 1)
- seleccionar no desplazamiento en el Shifter (S0 = 0, S1 = 0)
- conectar la entrada del registro A al bus C (C3 = 1, C2 = 0, C1 = 1, C0 = 0)
- habilitar bus C (ENC = 1)

con estas señales y sincronizado por el reloj se guarda la dirección contenida en la instrucción en el registro A. A continuación (próximo ciclo de reloj) se colocan las señales:

- conectar la salida del registro A al bus B (B3 = 1, B2 = 0, B1 = 1, B0 = 0)
- seleccionar cargar el BLatch (L0 = 1)
- seleccionar cargar MAR (M0 = 1)
- seleccionar cargar el MBR (M1 = 1)
- seleccionar lectura de memoria (M2 = 1)

En el paso siguiente se realiza el *execute*. Para ello se deben activar las siguientes señales:

- conectar la salida del registro AC al bus B (B3 = 0, B2 = 0, B1 = 0, B0 =1)
- seleccionar cargar el BLatch (L0 = 1)
- seleccionar MBR en el A Mux (X0 = 0)
- seleccionar la operación A + B en la ALU (F0 = 0, F1 = 0)
- seleccionar no desplazamiento en el Shifter (S0 = 0, S1 = 0)
- conectar la entrada del registro AC al bus C (C0 = 1, C1 = 1, C2 = 0, C3 =1)
- habilitar bus C (ENC = 1)

Sincronizado por el reloj se realiza la operación en la ALU y se guarda el resultado en el registro AC.

Cabe destacar que al ser una instrucción cuyo resultado se almacena en un registro, no corresponde que exista un paso de *write*.

5 Lógica Cableada vs Lógica Microprogramada

Hay dos filosofías de diseño claramente diferenciadas para la Unidad de Control: la “lógica cableada” y la “lógica microprogramada”.

El diseño de la CU en base a la filosofía de “lógica cableada” se hace como cualquier circuito secuencial. Para el ejemplo anterior deberíamos tomar como salidas todas las señales de control necesarias y como entradas los bits del código binario de las instrucciones.

En este caso el resultado es óptimo desde el punto de vista de la velocidad de los circuitos, pero tiene poca flexibilidad ya que cualquier cambio en el diseño del set de instrucciones de la CPU genera la necesidad de re-escribir el diagrama de estados y, por ende, cambiar todo el circuito lógico que lo implementa.

En cambio la filosofía de “lógica microprogramada” propone que la CU se construya en base a un autómata más sencillo que “ejecute” el ciclo de instrucción de cada instrucción de la CPU siguiendo en secuencia un conjunto de “microinstrucciones” que contienen los valores de las señales apropiados para esa instrucción particular. Es decir que para cada instrucción a nivel de la CPU existirá un “microprograma”, consistente en una secuencia lógica de microinstrucciones que establecerán el orden de los eventos necesarios para lograr la ejecución de la instrucción en la CPU, incluyendo todas las “bifurcaciones” de la secuencia que se requieran en base a los distintos tipos de direccionamiento y cualquier otra variante que admita la instrucción específica. En este caso hay una penalización en la velocidad de proceso porque el microprograma se almacena en una ROM interna de la CPU y cada microinstrucción debe ser leída de ella, pero simplifica enormemente la tarea de modificar el set de instrucciones, ya que simplemente alcanza con cambiar el contenido de la ROM del microprograma. También las máquinas microprogramadas permiten fácilmente implementar instrucciones complejas, tales como la búsqueda de un elemento en un array, ó la copia o la comparación de arrays, etc.

La tendencia en materia de diseño de CPUs era, hasta la década de 1980, utilizar lógica microprogramada. La aparición de las arquitecturas RISCs puso en cuestión esa tendencia, ya que dichas propuestas promovían el uso de lógica cableada, en particular asociado al concepto de optimizar la velocidad de proceso. De todos modos es probable que la ventaja que esos diseños obtuvieron en materia de rendimiento se deba principalmente al uso óptimo de técnicas tales como el “pipelining” permitido por las características de su set de instrucciones. El punto es que si se usa un set “reducido” con fines de optimización, también resulta más sencillo implementar la CPU en “lógica cableada”.