4. Microprogramación (Nivel 1)

La ejecución de una instrucción de máquina (nivel 2) requiere de varias etapas internas dentro de la CPU para ser ejecutadas. Así cada instrucción de nivel 2, dependiendo del tipo de arquitectura de CPU, puede requerir de varias transferencias internas o **micro-instrucciones**. Adicionalmente cada micro-instrucción se divide en fases internas o sub-ciclos donde se sincronizan de manera secuencial los eventos que ésta implica. Estos eventos son uniformes e independientes de la funcionalidad de la micro-instrucción que se esta ejecutando. Las CPU que presentan este tipo de organización basado en la ejecución de micro-instrucciones recibe el calificativo de **micro-programadas**.

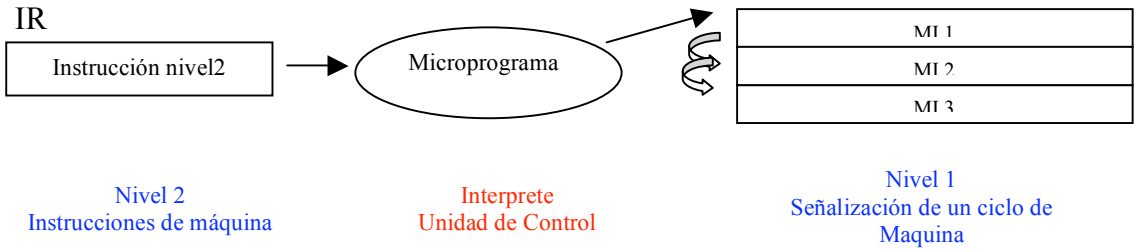
Para la correcta ejecución de cada ciclo de máquina (o trayectoria de datos), es necesario entregar la señalización de control a cada una de las unidades involucrada en dicho proceso (ALU, buses internos, registros, interfaz de bus, etc). La información de dicha señalización esta íntegramente incluida en la codificación de la **micro-instrucción** y corresponde a la información que es directamente interpretada por el hardware de la CPU.

4.1 Ejecución de instrucciones en el nivel de Microprogramación

El nivel de microprogramación esta determinado por el hardware de la CPU y sus funcionalidades básicas. El proceso de transformar una instrucción de lenguaje de máquina en sus micro-instrucciones (ciclos de maquina) es llevado a cabo por un interprete, llamado **microprograma**.

El Microprograma

Este interprete es el encargado directo de hacer el proceso de decodificación, es decir transforma las instrucciones de máquina (nivel 2) en la señalización de control (nivel 1) que gobiernan las transferencias de información (*trayectorias de datos* o *data-path*) entre las unidades funcionales de la CPU. La codificación de las señalizaciones de control, para los diferentes tipos de trayectorias de datos son las **micro-instrucciones**.
De esta forma cada micro-instrucción corresponde a la información de las señales que permiten controlar de manera completa un ciclo de procesador (data-path). Dicha funcionalidad de decodificación pertenece a la *Unidad de Control* de la CPU, luego el **microprograma** es el motor de esta unidad.



Observaciones:

- Jerarquía de Software (Nivel1-Nivel2): El beneficio de esta técnica es que abstrae la problemática de las funcionalidades básicas de hardware, proporcionando un conjunto de instrucciones al nivel del lenguaje de maquina (asembler) de mayor complejidad, y por otro lado implementadas en un diseño de hardware más simplificado. Así el hardware no necesita tener implementadas las funcionalidades del lenguaje de máquina, sino que solamente las funcionalidades más básicas (micro-instrucciones) con las cuales se puedan ejecutar las instrucciones de lenguaje de máquina.

- RISC: Pese a que el microprograma tiene incidencia en la simplificación del diseño (costo), esto implica que la ejecución de una instrucción de lenguaje de maquina ocupa más ciclos de los que usaría si dicha instrucción estuviera completamente implementada en la CPU. Existen arquitecturas que eliminan el nivel de microprogramación, permitiendo que las instrucciones de máquina sean precisamente las funcionalidades básicas de la CPU (arquitecturas ***RISC***), es decir cada instrucción de máquina es una trayectoria de datos. Esto implica un mayor costo de diseño, hardware de prestaciones más complejas y un lenguaje de máquina menos flexibles para programadores de nivel 2.

- Microprograma: El **microprograma**, por ser un programa es un conjunto de instrucciones que no pueden ser interpretadas por otra cosa que por el **hardware**, por tanto sus instrucciones están codificadas en micro-instrucciones. El microprograma típicamente esta almacenado en un bloque de memoria interna (de sólo lectura, ROM) perteneciente a la Unidad de Control**.**

4.2 Unidades involucradas

Registros y buses internos:

Los registros son las unidades básicas de almacenamiento de la CPU, y se accede a estos mediante los buses internos. Los buses comunican los registros internos con los registros de entrada o salida de la ALU o con los registros asociados con la interfaz del bus de datos y direcciones, para la transferencia con las restantes unidades de la arquitectura de un computador (memorias, I/O, etc.),

Típicamente la comunicación entre la ALU y los registros es mediante buses unidireccionales (permiten transferencias en una única dirección), siendo necesario por lo tanto un bus de entrada y otro independiente de salida entre la ALU y los registros.

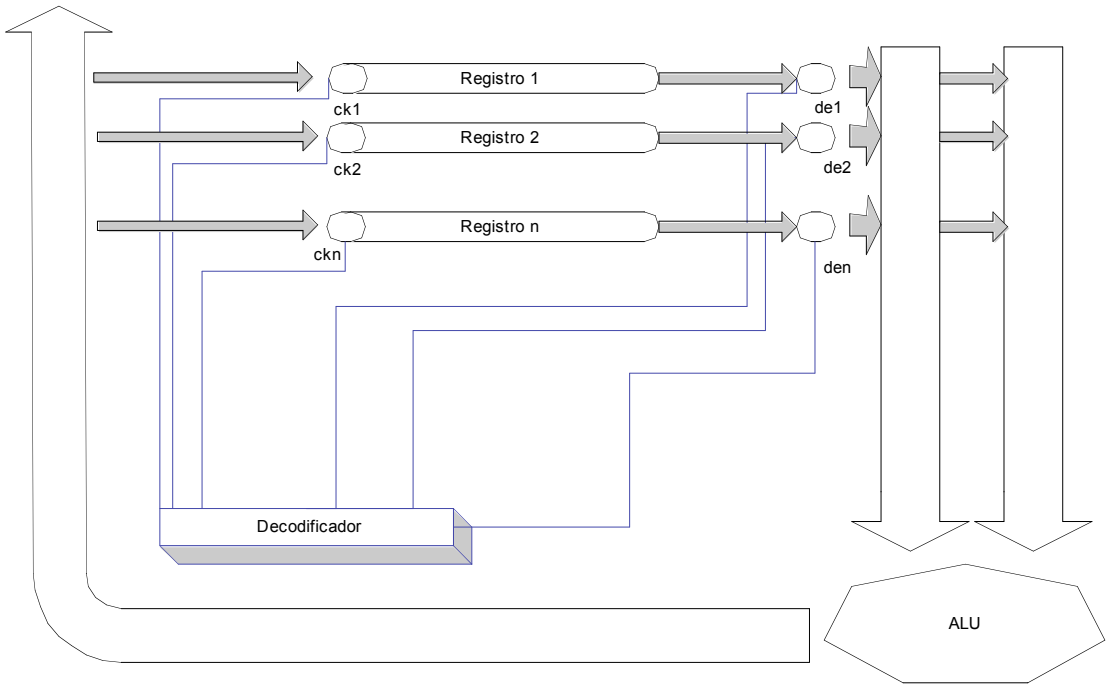


Figura m.1: *Diagrama de micro-arquitectura*

En la figura se puede apreciar que para cargar un registro desde el bus de salida de la ALU se debe levantar la señal de control ck_n, permitiendo que los flip-flop D del registro n-esimo carguen el contenido de las líneas de dicho bus. De forma recíproca levantando la señal de_n se pasa al estado de baja impedancia y las señales retenidas en los flip-flop D del registro n-esimo se propagan por las líneas del bus de entrada a la ALU. De esta manera las líneas ***ck y de*** de cada registro determinan cual de estos es direccionado.

La implementación de un registro paralelo, utilizando lógica secuencial, puede ser observada en la siguiente figura:

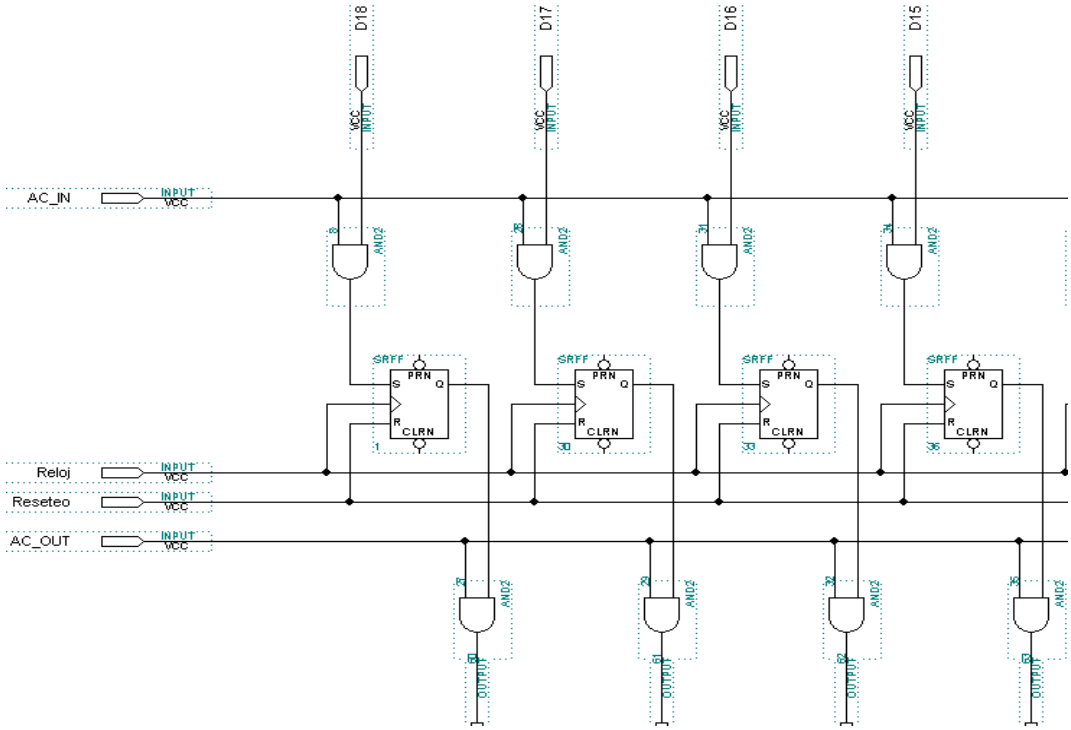


Figura m.2: Registro paralelo implementado mediante flip-flop JK

Multiplexores

Dispositivo digital que posee 2^n entradas y 1 salida. Mediante sus n líneas de control direcciona una de sus entradas a la salida. Se utiliza para multiplexar buses o unidades de direccionamiento.

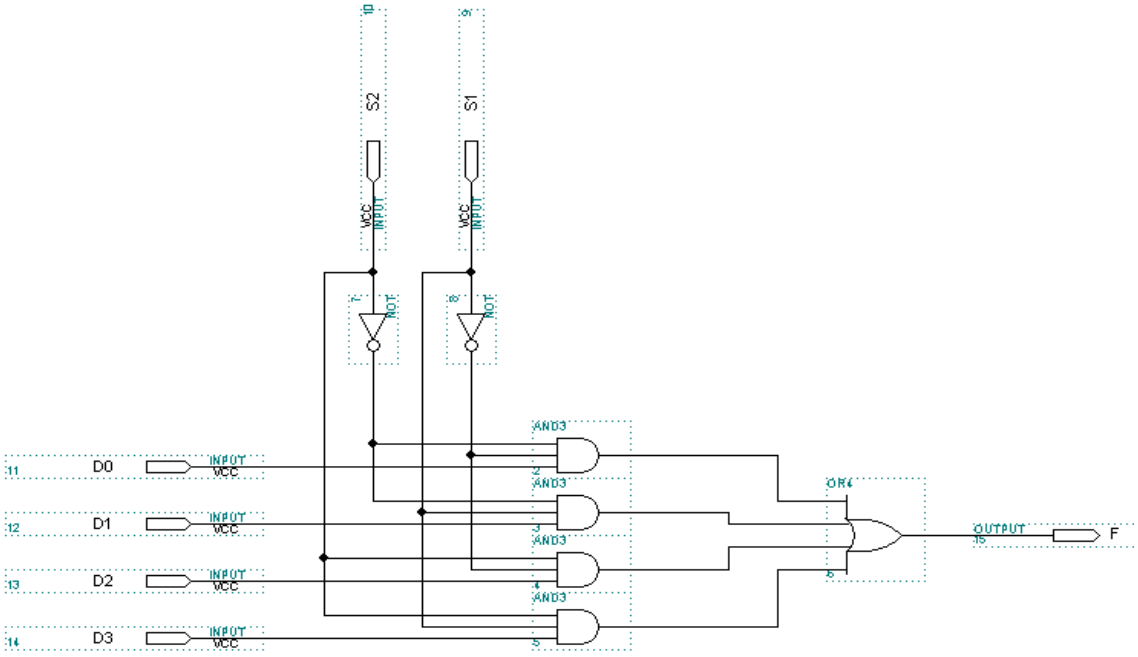


Figura m.3: Multiplexor de 4 es a 1

Decodificadores

Este dispositivo tiene n líneas de entrada y 2^n de salida. Se utiliza como decodificador de direcciones en el acceso a los registros internos de la CPU.

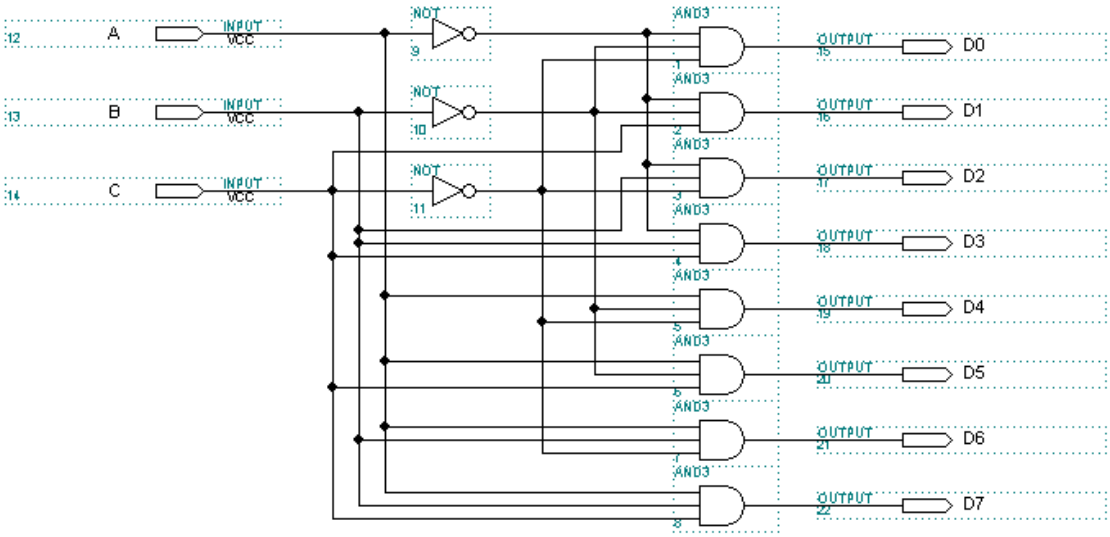


Figura m.4: Decodificador 3 es a 8. Con tres líneas de control puede direccionar 8 unidades

Relojes

Este dispositivo provee una señal cuadrada periódica que permite sincronizar el funcionamiento de las diferentes unidades internas de la CPU, en el proceso de ejecutar una micro-instrucción. Sus flancos definen ciclos de máquina. Cada ciclo implica el tiempo necesario para ejecutar una micro-instrucción. A la vez cada micro instrucción se divide en sub-etapas las cuales necesitan tener una referencia que las ejecute de manera ordenada y sincronizada. Estas sub-etapas son gobernadas por la misma señal de referencia desfasada en fracciones de su periodo fundamental. En el ejemplo de arquitectura de nivel de micro programa que se muestra a continuación, los ciclos de máquina se dividen en 4 fases.

Memoria Principal

La comunicación entre la CPU y la memoria principal es mediante el bus del sistema (interfaces de dirección, control y datos). Internamente la CPU tiene una unidad encargada de manejar toda la interfaz con el bus externo y en este punto describiremos sus elementos básicos, relevantes al nivel de micro-programa.

Un acceso a memoria (independiente del tipo que esta sea), necesita más tiempo que el necesario para ejecutar un ciclo de máquina (en promedio varios ciclos de máquina). Este hecho implica que la CPU debe mantener los valores correctos en los buses de datos y direcciones por varias micro-instrucciones. Una solución para esto es que la interfaz de bus posea registros, para nuestro ejemplo **MAR** (*Memory Address Register* o registro de dirección de memoria) y **MBR** (*Memory Buffer Register* o registro de intercambio de memoria) , los que actúan como interfaz con el bus externo.

De esta manera estos registros están conectados por el lado de la CPU con los buses internos y permiten mantener la información de un acceso a memoria hacia el bus del sistema. Esto permite que la CPU realice otras operaciones mientras se ejecuta el ciclo de bus.

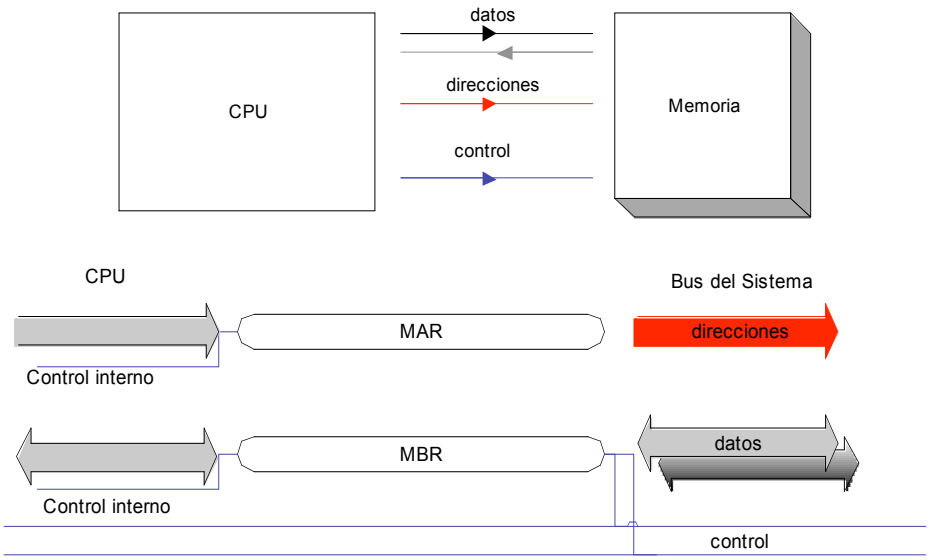


Figura m.5: *Diagrama interfaz CPU bus del sistema*

- Bus de Direcciones: El bus de direcciones es unidireccional, pues la CPU es el único dispositivo maestro, es decir el único que tiene la facultad de accesar a las demás unidades en la arquitectura (memoria, puertos de los I/Os). El registro **MAR** actúa como interfaz entre la CPU y el bus de direcciones, el cual se carga del lado de la CPU y se propaga hacia el bus del sistema.

- Bus de Datos: El caso del registro **MBR** es la interfaz con el bus de direcciones y su flujo puede ir en ambas direcciones. Por tanto esta conectado internamente a los buses de entrada y de salida de datos de la CPU. Adicionalmente existen al menos 2 señales de control en este registro, que van a las líneas de control del bus del sistema y determina el tipo de acceso a memoria (lectura o escritura).

4.3 Ejecución

Ruta de datos

La ruta de datos se asocia al proceso de transferencia de información entre la ALU y las unidades de almacenamiento presentes en la CPU (registros), utilizando como medio o canal de comunicación los buses internos. Se puede decir que cada ciclo de máquina ejecuta una trayectoria de datos y por lo tanto todas las instrucciones de nivel 2 corresponden a una secuencia ordenada de estos eventos de transferencias básicos (micro-instrucciones) gobernados por la Unidad de Control. Ejemplificaremos estos conceptos en una micro-arquitectura elemental, pero al mismo tiempo suficientemente rica para ilustrar la naturaleza de las arquitecturas de CPU micro-programadas.

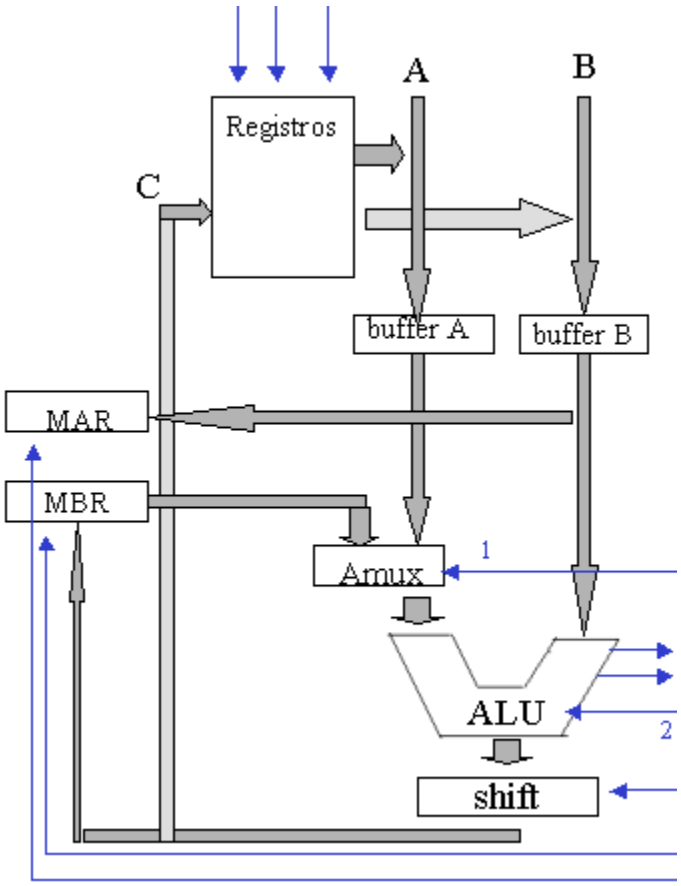


Figura m.6: *Esquema de las unidades más importantes en una micro-arquitectura básica. Las líneas azules corresponden a las señales de control que gobiernan las unidades.*

Las características de esta micro-arquitectura son:

- Palabras de 16 bits, es decir registros y buses internos de ancho de 16 líneas.
- Hay 16 registros (codificables para efecto de direccionamiento con 4 bits)
- La ALU tiene dos buses de entrada (**A, B**) y es capaz de ejecutar las siguientes cuatro operaciones básicas. (A+B; A and B ; A ; not A). Estas operaciones se codifican con 2 bits.
- La salida de la ALU pasa a un registro de desplazamiento (shift register) de un bit a la derecha o izquierda.
- Los buses no están conectados directamente a la ALU si no que por medio de registros de retenimiento (**buffer A** y **buffer B**). Esto es necesario pues la ALU es un dispositivo de naturaleza combinacional y por lo tanto si ejecuta una operación sobre alguno de sus argumentos, dicho cambio no debe manifestarse en la entrada. De esta manera los registros mantienen los valores de los argumentos previos a la operación de la ALU durante todo el ciclo de máquina.
- El bus de salida permite la conexión con todos los registros y la interfaz de datos del bus del sistema (**MBR**)
- Para la interfaz de bus del sistema están los registros **MAR** y **MBR**. El **MAR** puede cargarse del registro **buffer B** en paralelo con una operación de la ALU. El **MBR** se controla con una señal binaria que indica si se carga o no del bus de salida (**bus C**)
- Hay dos señales de control binarias que determinan el tipo de acceso al bus del sistema (lectura o escritura).

- El multiplexor **AMUX** permite direccionar, mediante una línea de control binaria, si la entrada a la ALU es del registro **buffer A** o del registro **MAR**.

Micro-instrucción (*señales de control*)

La micro-instrucción es la información de la señalización, que permite dirigir la trayectoria de datos de un ciclo de máquina, mediante el control de todas las unidades internas involucradas en la ejecución. Nos basaremos en analizar el ejemplo de arquitectura ya presentado, en la cual cada ciclo de máquina necesita las siguientes señales de control (cada señal con dos posibles estados):

- 16 señales para controlar el **bus A** (entrada a la ALU), direccionar uno de los 16 registros al bus de entrada como argumento.
- 16 señales para controlar el **bus B** (entrada a la ALU)
- 16 señales para controlar el **bus C** (salida de la ALU), direccionar uno de los 16 registros para almacenar el resultado de la operación.
- 2 señales para controlar la carga de los registros **buffer A** y **buffer B**.
- 2 señales para controlar la operación de la **ALU**
- 2 señales de entrada al registro de desplazamiento (**shift**)
- 2 señales para controlar la carga de los registros **MAR** y **MBR**
- 2 señales para indicar lectura y escritura de memoria (bus de control)
- 1 señal para controlar el multiplexor **AMUX**.

Con estas 59 señales se dirige por completo la trayectoria de datos en un ciclo de máquina de la CPU, lo que implica básicamente cargar los argumentos de la ALU (desde los registros internos o el MBR), ejecución de la operación y almacenamiento del resultado en uno de los registros internos o el MBR.

Codificación de la Micro-instrucción

Una manera de codificar esta información es tener por cada señal un bit (codificando los dos estados posibles de la señal, activa o inactiva). En la práctica se trata de comprimir dicha información en menos bit, con el costo de poner dispositivos adicionales que hagan el proceso de decodificación y entreguen las 59 señales necesarias de control.

Observaciones de diseño:

- Las 16 señales de control que direccionan los registros, tanto para los buses de entrada como salida de la ALU, pueden codificarse con 4 bits. De esta forma estas 4 señales serían las señales de entrada a un decodificador de 4 es a 16, como el de la ***figura m.4***.

- Otro aspecto es que independiente del ciclo de máquina siempre deben actualizarse los **buffer A** y **buffer B**, por tanto no es información de la naturaleza de la micro-instrucción.

- Adicionalmente se debe incorporar un bit para indicar si se desea almacenar la salida en un registro interno (señal **ENC**), esto debido a que el resultado de la operación de la ALU puede no ser relevante o puede ser utilizado solamente como interfaz con el bus de direcciones y por lo tanto no sea de interés almacenar dicho valor.

De esta manera se pasa de 59 bits a 22 bits para codificar una micro instrucción, la que determina por completo la ruta de datos en un ciclo de máquina. El detalle del formato de dicha micro instrucción y sus campos se muestra a continuación.

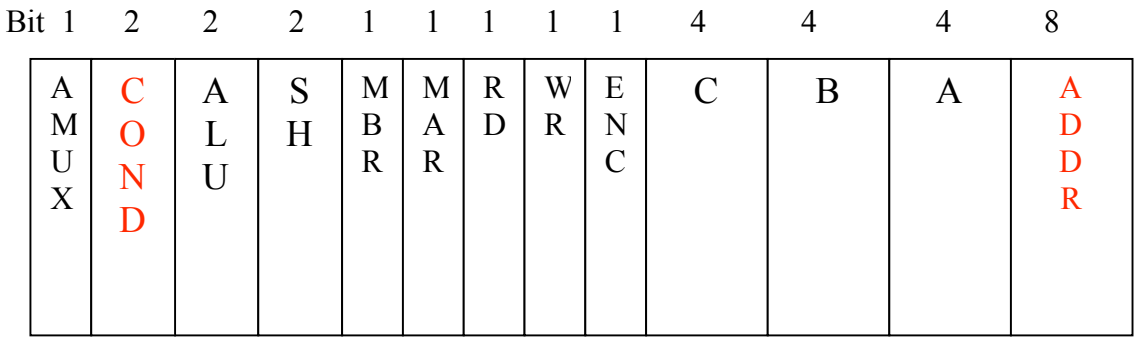


Figura m.7: *Formato de la trama de bit de la micro-instrucción*

Cada campo esta asociado a las señales previamente descritas. El campo **ENC** permite cargar la salida del **bus C** en uno de los registros direccionado por el campo C. Los campos **COND** y **ADDR** permiten implementar saltos condicionales o incondicionales en el flujo lógico de ejecución de las micro-instrucciones.

Observación:

En la micro-instrucción no esta la información de la secuencia de eventos que la componen, lo relacionado a la lógica de secuenciamiento es parte de la unidad de control, la cual se sincroniza con las señales de reloj antes mencionadas.

Cronología de ejecución de Micro-instrucciones

Se ha visto en detalle todo lo relacionado con la señalización básica que gobierna un ciclo de máquina y la manera de codificar dicha información, pero un aspecto que no se ha mencionado es la cronología de los eventos (basados en principios de causalidad) en la ejecución de dicho ciclo. Es decir en que parte del ciclo de máquina la salida de cada unidad debe ser considerada válida.

Para este ejemplo los eventos básicos y su relación de causalidad vienen dados por lo siguiente:

1. Cargar los registros de entrada a la **ALU**;
2. Una vez que las entradas estén estabilizadas (nivel de voltaje estable posterior al flanco de subida), dar tiempo a la **ALU** y al **shift** para estabilizar su salida;
3. Almacenar el resultado.

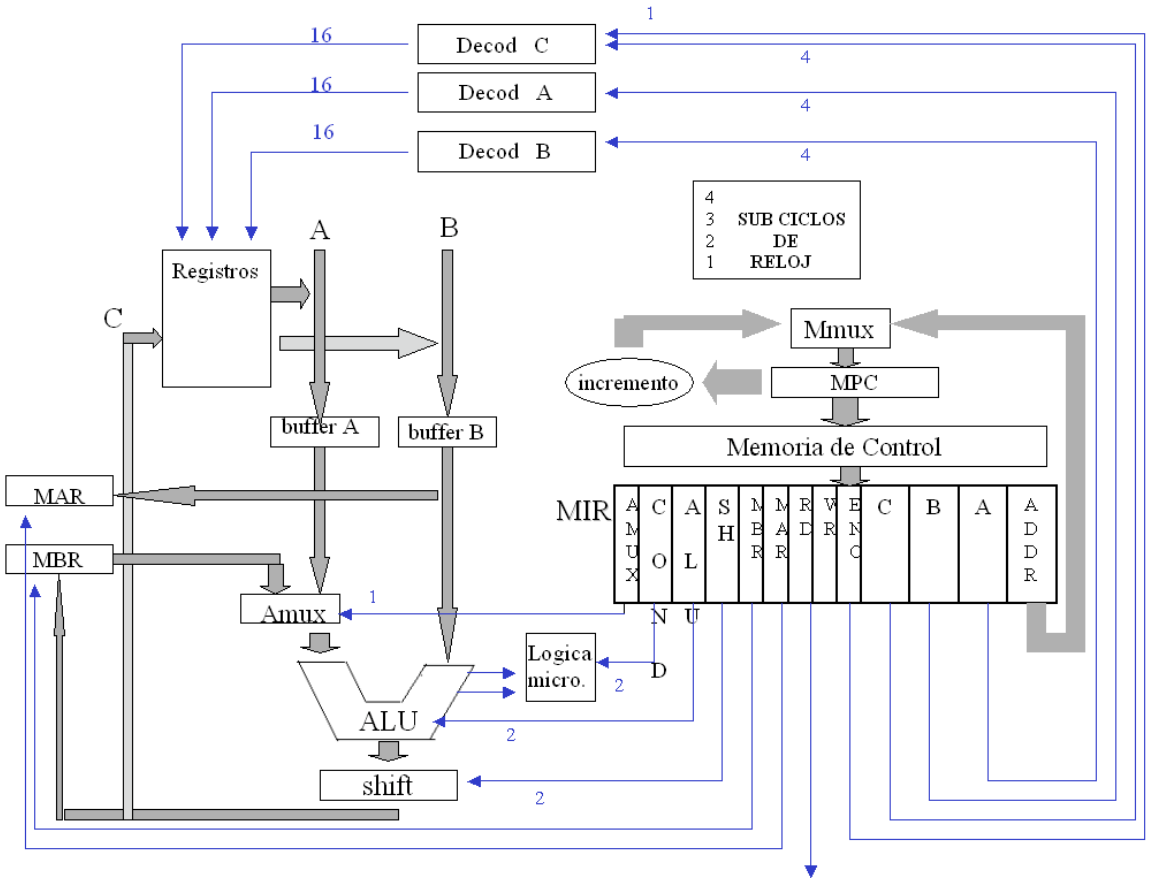


Figura m.7: *Micro-arquitectura de estudio*

De la ***figura m.7*** se pueden apreciar las unidades funcionales ya mencionadas, pero adicionalmente se explicitan las unidades que proporcionan la señalización de control para gobernar un ciclo de máquina, todas parte integral de la **Unidad de Control**. Dentro de estas unidades se distinguen:

- **Memoria de control**: memoria de tipo ROM en la cual esta la codificación de todas las posibles trayectorias de datos (micro-instrucciones)
- **MPC** (Micro Program Counter): registro que almacena la información de la dirección en memoria de control de la siguiente micro-instrucción a ejecutar. El proceso de decodificación que hace el ***microprograma*** implica definir cual es la secuencia de valores para este registro (es el equivalente al PC pero en el ámbito de microprogramación).
- **MIR** (Micro Instruction Register): registro almacena la codificación de la micro-instrucción que esta siendo ejecutada. Sus bits directamente proporcionan la información de control para cada unidad funcional en la arquitectura. Se actualiza al comienzo de cada ciclo de máquina.
- **Buses internos (direccionamiento de los registros de control)**
- **Decodificadores**
- **Señal de sincronización (clock)**

Para este ejemplo de arquitectura el periodo de máquina se divide en cuatro etapas (cada una gobernada por un flanco de reloj), lo que da cabida a 4 fases en el proceso de ejecución de una micro instrucción. Estas fases son:

- **Fase** 1: Cargar la micro-instrucción (con el formato mostrado) en un registro interno de la Unidad de Control llamado **MIR,** para esto se direcciona la información del registro **MPC** sobre la memoria de control.

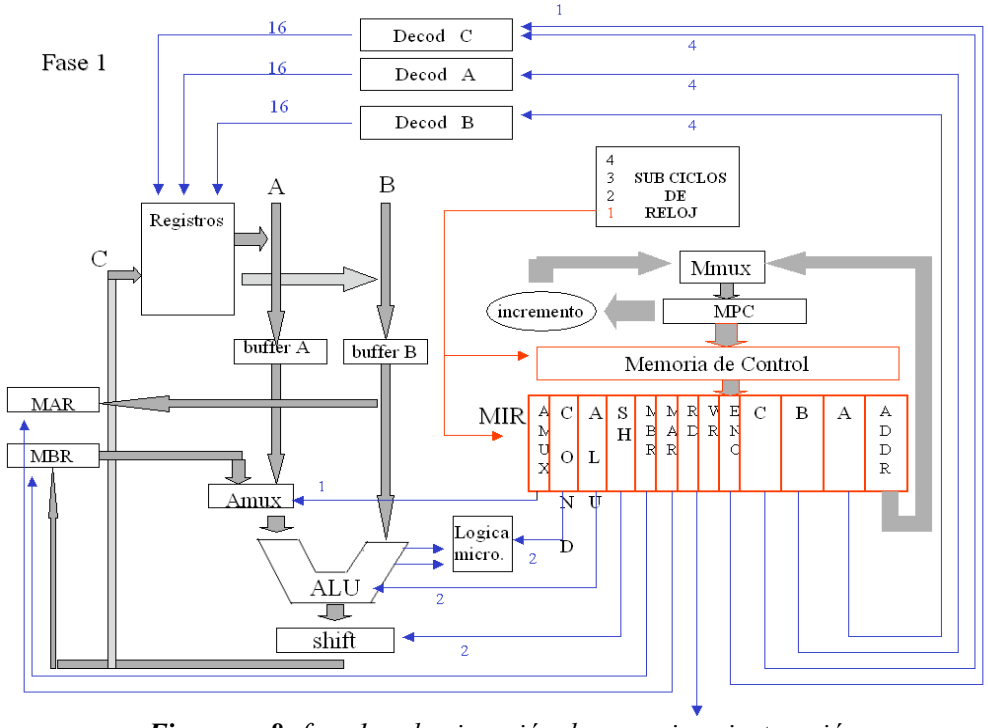


Figura m.8: *fase 1 en la ejecución de una micro-instrucción*

- **Fase 2:** Una vez que las señales del **MIR** están estabilizadas, estas comienzan a gobernar la trayectoria de datos, direccionando los registros internos a los buses A y B, y posteriormente reteniendo dicha información en los buffer A y B, respectivamente. Paralelamente la unidad de incremento calcula **MPC +1** (lo que sería correspondiente a la posición de la siguiente micro-instrucción a ejecutar).

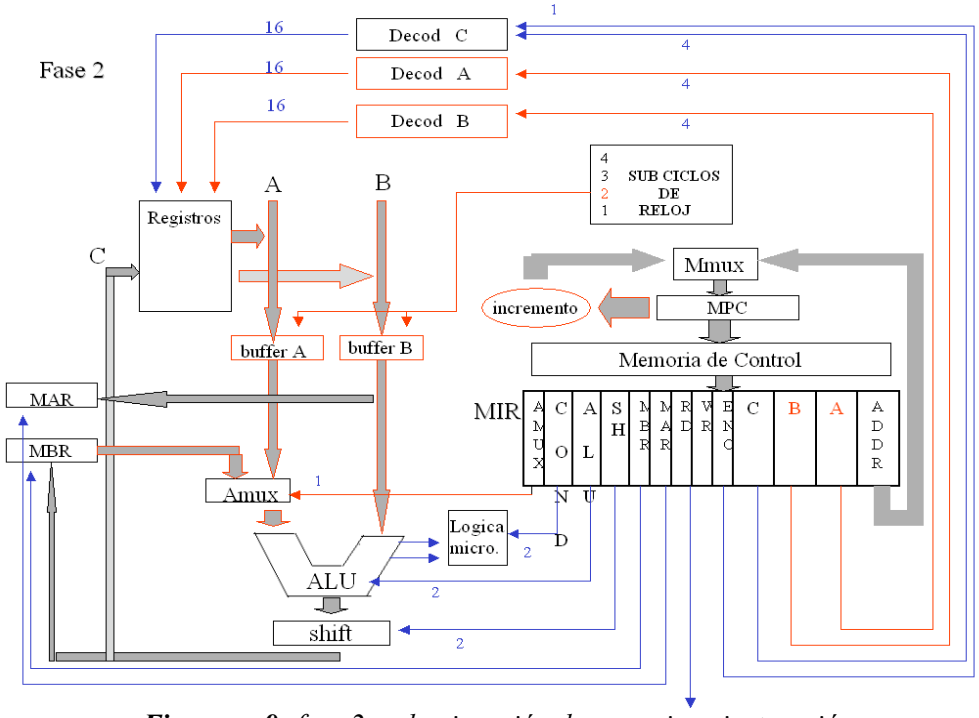


Figura m.9: *fase 2 en la ejecución de una micro-instrucción*

- **Fase 3** Dar tiempo a la **ALU** y al registro de desplazamiento para producir una salida estable a través del bus **C**. Paralelamente se carga el registro **MAR** con el contenido del bus B, dependiendo del campo **MAR**.

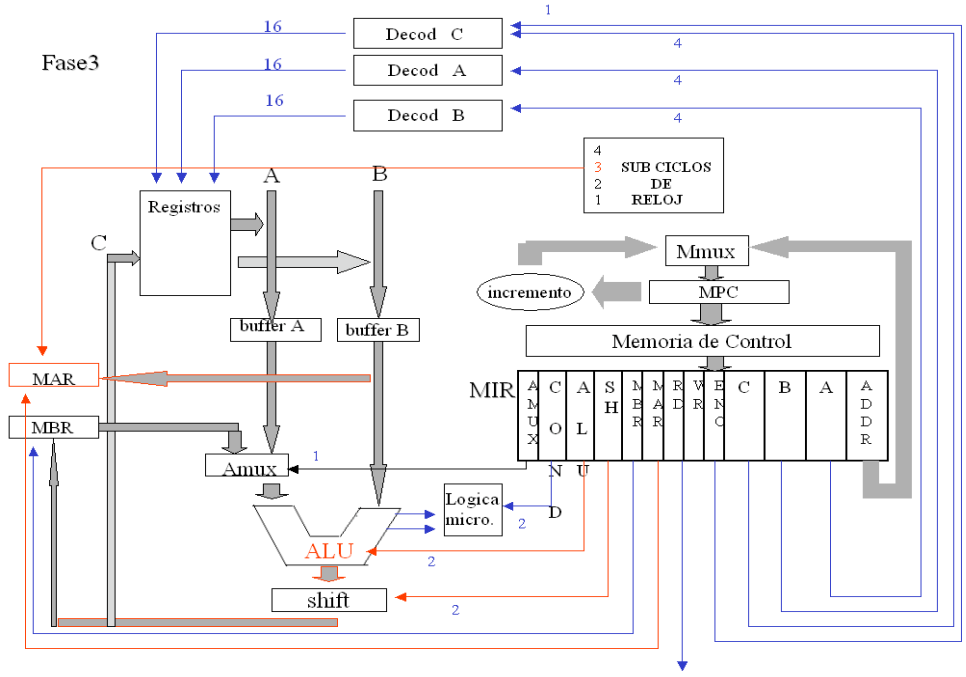


Figura m.10: *fase 3 en la ejecución de una micro-instrucción*

- **Fase 4** Almacenar la información retenida en el bus C en una de los registros internos o en el registro de interfaz del bus de datos (**MBR**), según el valor de los campos ENC, C y MBR.

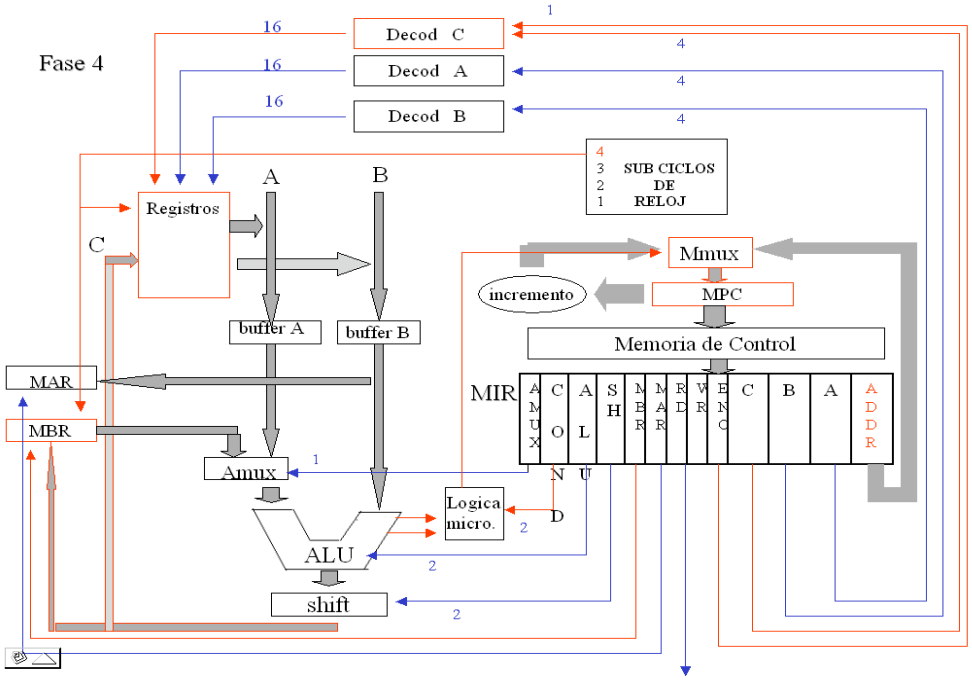


Figura m.11: *fase 3 en la ejecución de una micro-instrucción*

Secuenciamiento de Micro-instrucciones

A nivel de micro-programación también existe un flujo lógico de ejecución de las micro-instrucciones, para lo cual la propia micro-instrucción tiene campos que permiten hacer este manejo. Al igual que lo que ocurre en memoria principal al extraer las instrucciones de máquina, en la mayoría de los casos la siguiente micro-instrucción es la que se encuentra adyacente a la actual en memoria de control.

Cada micro-instrucción posee un campo ADRR que permite bajo ciertas condiciones cargar dicha información en el MPC y por lo tanto que la siguiente micro-instrucción sea dada por ese campo en la memoria de control. Esta condición de salto esta dada por el estado de la ALU, el cual queda caracterizado por las señales de salida N (indica si la operación de la ALU fue negativa), Z (indica si la operación de la ALU fue cero), y por un campo de condición que indica como interpretar dichas salidas para generar la condición de salto.
En este diseño hay cuatro posibles condiciones de salto:

0: No saltar (MPC=MPC+1)
1: Saltar a ADRR si N = 1 (operación resultó negativa)
2: Saltar a ADRR si Z=1 (operación resultó cero)
3: Saltar a ADRR incondicionalmente

Ejemplo de ejecución de micro-instrucciones

Como se mencionó anteriormente cada *micro-instrucción* se encuentra contenida en memoria de control, por tanto el **microprograma** debe determinar para cada instrucción de nivel 2 la posición en la memoria de control donde se encuentran la secuencia de micro-instrucciones que la representan.

A continuación se presenta un ejemplo de como se ejecuta una instrucción de máquina, distinguiendo los ciclos de máquina necesarios y la forma de codificar las micro-instrucciones que caracterizan cada uno de esos ciclos. La instrucción de máquina a ejecutar es de tipo **Load**, es decir cargar la posición de memoria contenida en un registro de direccionamiento de la CPU (r1), en otro registro interno de la CPU. La sintaxis en un lenguaje assembler seria como sigue:

Assembler	Significado
load A, (r1)	A = Memoria (r1)

Además supondremos que se requiere sólo un ciclo de lectura es decir el bus de datos del sistema es de 16 bit. Luego este proceso puede ser descompuesto en dos micro-instrucciones o trayectorias de datos.

M I 1:

***Descripción*:** - se pone el registro interno **r1** en el bus B
- se pone el contenido del bus B en el registro MAR
- se da la señalización de un ciclo de lectura

Señalización: - el campo B debe tener la codificación del registro r1.
- el campo MAR debe indicar leer del bus B
- los campos ENC y MBR deben estar en 0 para no almacenar la salida de la ALU.
- los campos RD y WR deben indicar un ciclo de lectura a memoria.
- el campo COND debe estar en 0 para ejecutar la siguiente micro-instrucción en memoria contigua.

Cronología de ejecución

1. se carga el MIR
2. los decodificadores direccionan el registro r1 al bus B
3. no interesa lo que haga la ALU, pero en este ciclo se lleva el buffer B al registro de direcciones MAR
4. no se guarda la información del bus C y se actualiza el MPC apuntando a la siguiente micro instrucción.

M I 2:

Estado inicial:

Se supone que el ciclo de lectura ha finalizado y por lo tanto el registro MBR contiene la información de memoria direccionada en la micro-instrucción anterior.

***Descripción*:**

- AMUX debe direccionar el contenido del MBR como argumento de entrada de la ALU (bus A)
- se ordena que la ALU implemente la función identidad, es decir propague la salida del bus A sobre su salida y adicionalmente el **shift** no debe generar ningún desplazamiento.
- la salida se direcciona al registro interno A de la CPU por el bus C.

Señalización:

- el campo AMUX debe tener el bit que direcciona el registro MBR
- el campo ALU debe indicar la codificación de la identidad
- el campo C debe direccionar el registro A
- el campo ENC debe habilitar la escritura en los registros
- los campos MBR y MAR deben estar en 0.
- el campo SH debe indicar que no haya desplazamiento
- los campos RD y WR deben indicar que no hay acceso al bus del sistema.

Cronología de ejecución

1. se carga el MIR

2. se direcciona el MBR sobre el bus A

3. la ALU propaga el bus A sobre su salida y el shift no desplaza su entrada

4. se guarda la información del bus C en el registro A y se actualiza el MPC apuntando a la siguiente micro instrucción.