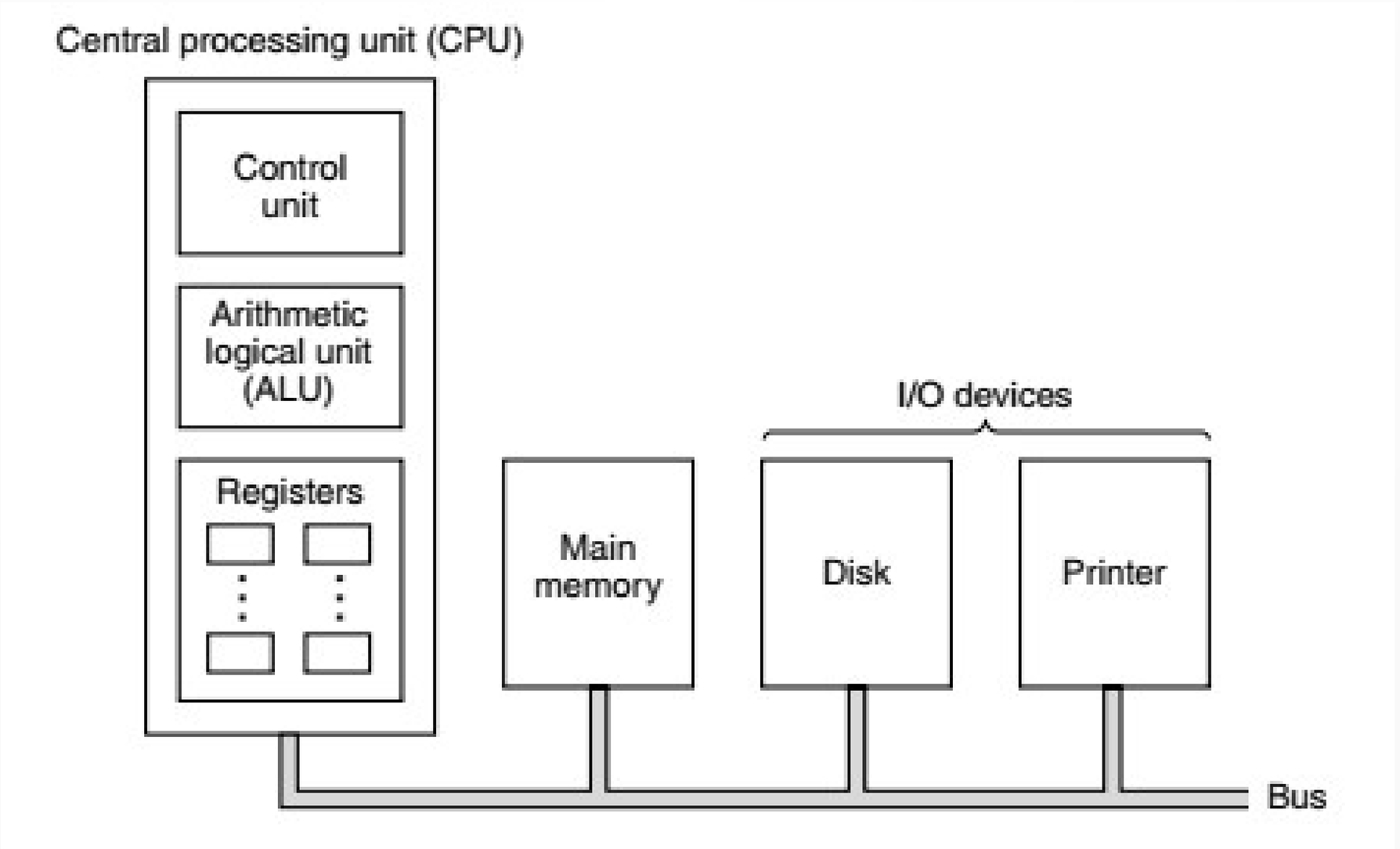
Arquitectura y Organización de Computadoras

El Procesador

MS. Ing. Ticiano J. Torres Peralta

REV2022

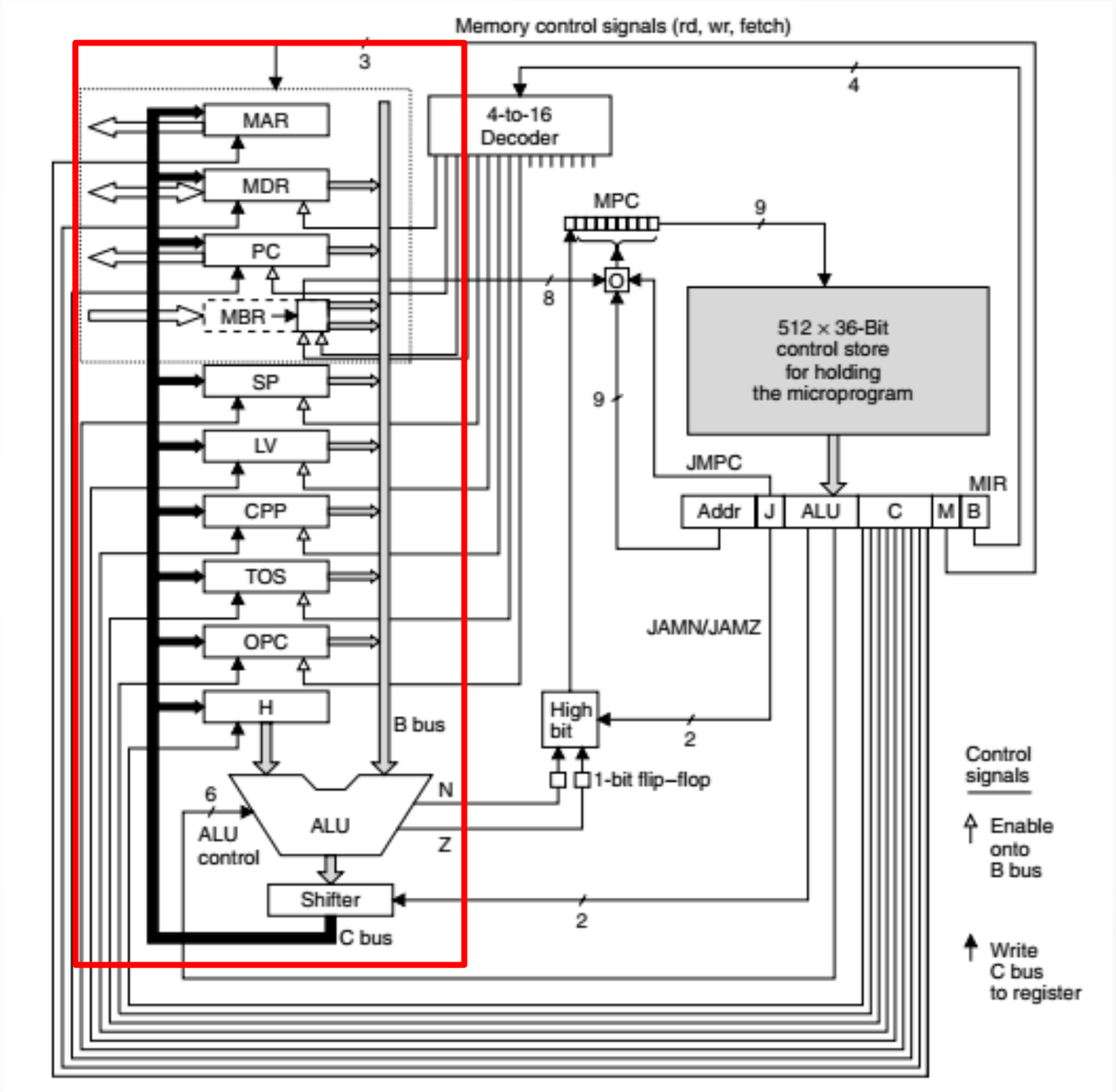
Organización Básica de una Computadora



El procesador

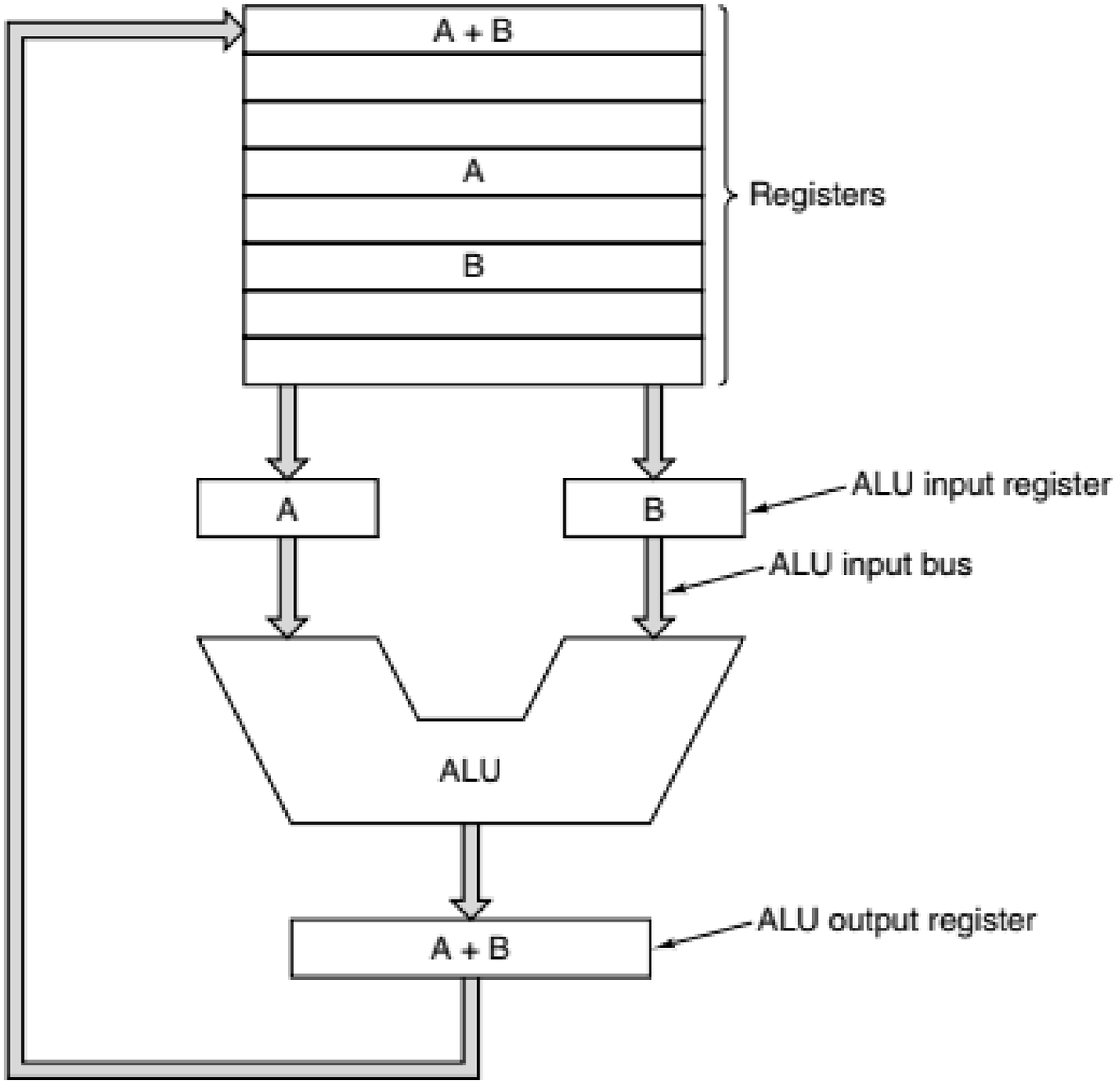
- **Unidad de Procesamiento Central – UPC (Central Processing Unit – CPU):** Conocida como el cerebro de la computadora. Su propósito principal es la ejecución de instrucciones guardadas en memoria. Esto lo hace desde el momento que la maquina inicia hasta el momento que se la apaga. En ningún momento, mientras este prendida la maquina, deja de ejecutar. El proceso de ejecutar instrucciones, se lo conoce como el **ciclo de ejecución**.
- El CPU esta compuesto por varios componentes pero principalmente tienen dos partes, el **camino de datos** y la **unidad de control**.

Camino de Datos



El procesador

- **Camino de datos (Data Path):** es la parte del procesador que típicamente contienen varios registros y la unidad aritmética-lógica, todo conectado a través de uno o mas buses internos al CPU. Hoy en dia, en arquitecturas mas modernas y complejas, es mas común el termino Unidad de Ejecución (Execution Unit – EU), y puede contener muchos tipos de unidades funcionales, como: unidad de punto flotante (Floating-Point Unit – FPU), unidad de generación de direcciones (Address Generation Unit – AGU), unidad de manejo de memoria (Memory Management Unit – MMU), etc.
- **Unidad de Control (Control Unit):** es responsable de, a través de señales de control, modificar el funcionamiento del camino de datos para que se puedan cumplir con las necesidades del ciclo de ejecucion, como por ejemplo, leer una instrucción de memoria.


A + B
A
B
Registers
A
B
ALU input register
ALU input bus
ALU
A + B
ALU output register

Camino de Datos

- **Registros:** memorias temporarias de alta velocidad que pueden ser usados para contener instrucciones, datos o también información de control. El CPU puede tener registros de uso general y registros de uso especifico. Normalmente todos los registros del sistema tienen el mismo tamaño, pero ciertos registros de uso especifico pueden ser diferente.
- **Unidad Aritmetica Logica – UAL (Arithmetic-Logic Unit – ALU):** es una unidad funcional que puede hacer operaciones aritméticas y lógicas como: suma, resta, multiplicación, división, OR, AND, y NOT. Tiene dos entradas y una salida. Puede tener, pero no es requerido, dos registros de entrada y uno de salida para tener los datos de forma temporaria mientras completa su operación.

Registros del Camino de Datos

- **Program Counter (PC):** el registro mas importante del CPU. Contiene en el, la dirección de la próxima instrucción a ser ejecutada.
- **Instruction Register (IR):** otro registro de suma importancia. Contiene la instrucción misma, que esta siendo ejecutada. (En nuestra arquitectura ejemplo, el IR es nombrado MBR – Memory Byte Register).
- **Registros de Entrada Salida:** Estos registros por lo general son usados para temporalmente sostener datos que fueron leídos de memoria o que tienen que ser escritos a memoria, y sus respectivas direcciones. (En nuestra arquitectura ejemplo, son el MAR - Memory Address Register y MDR – Memory Data Register).
- **Registros de Propósito General:** Estos registros pueden ser usados de forma general para sostener datos o direcciones de forma temporaria. El programador tienen accesos directo a estos registros, a diferencia de los registros de uso especifico.

Ciclo de Ejecución del Procesador

- El ciclo de ejecución del CPU es conocido como el ciclo buscar-decodificar-ejecutar (fetch-decode-execute cycle). Es una serie de pasos que toma el procesador para ejecutar una instrucción.
  1) Buscar la próxima instrucción de la memoria y colocarla en el IR.
  2) Modificar el PC para que apunte a la instrucción que sigue.
  3) Decodificar la instrucción que fue buscada. Osea, determinar que tienen que ser ejecutado.
  4) Si la instrucción requiere datos de memoria, determinar adonde están esos datos.
  5) Buscar los datos, y, si es necesario, colocarlos en algún registro del CPU.
  6) Ejecutar la instrucción.
  7) Volver a paso 1.
- Es importante notar que el ciclo de ejecución del CPU puede ser representado como un programa en software.

  **El ciclo de ejecución mostrado es un ciclo especifico a la MIC-1 y está solo para ejemplificar el concepto.

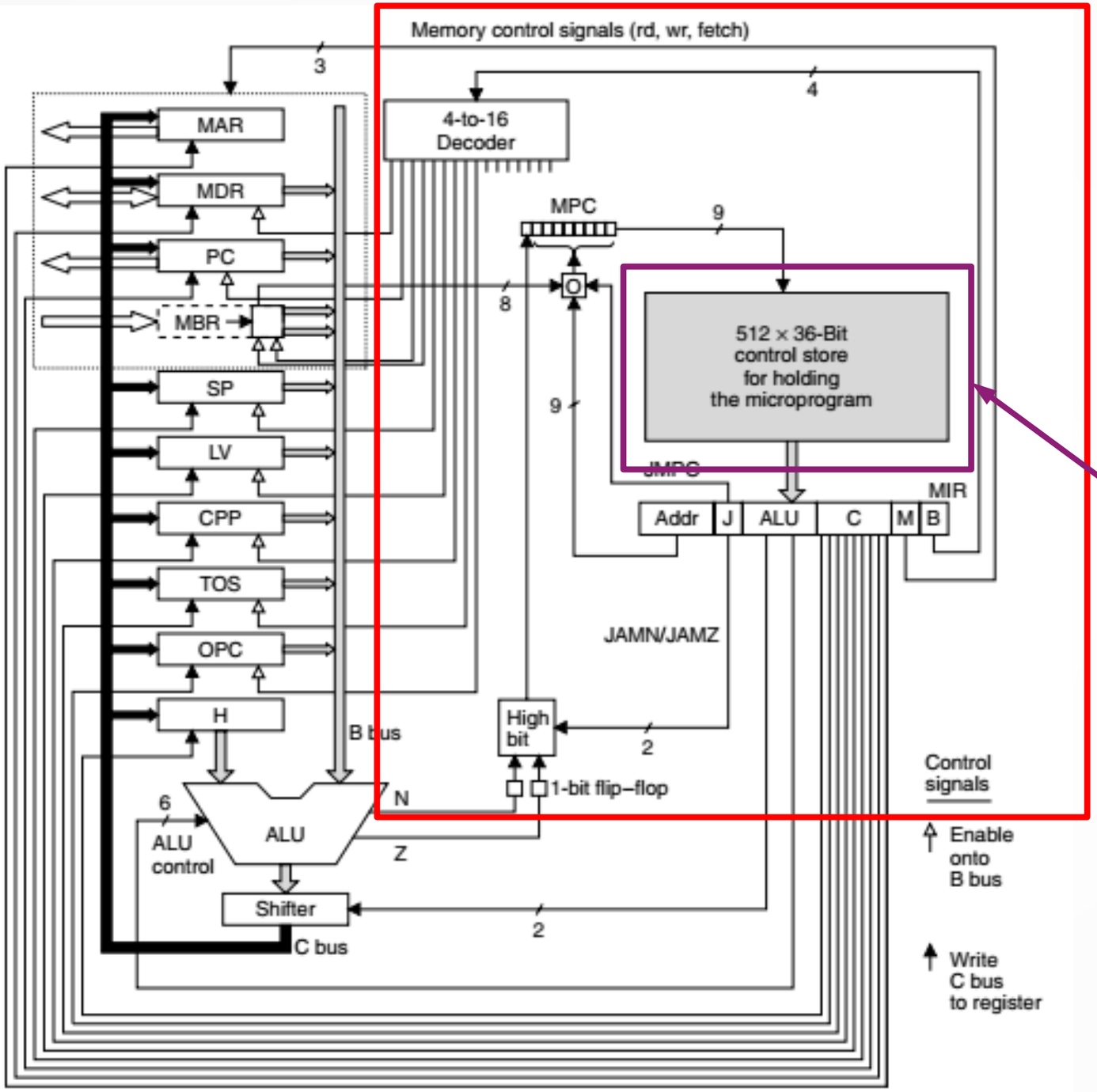
```
public class Interp {
    static int PC;                          // program counter holds address of next instr
    static int AC;                          // the accumulator, a register for doing arithmetic
    static int instr;                       // a holding register for the current instruction
    static int instr_type;                  // the instruction type (opcode)
    static int data_loc;                    // the address of the data, or –1 if none
    static int data;                        // holds the current operand
    static boolean run_bit = true;          // a bit that can be turned off to halt the machine

    public static void interpret(int memory[ ], int starting_address) {
        // This procedure interprets programs for a simple machine with instructions having
        // one memory operand. The machine has a register AC (accumulator), used for
        // arithmetic. The ADD instruction adds an integer in memory to the AC, for example.
        // The interpreter keeps running until the run bit is turned off by the HALT instruction.
        // The state of a process running on this machine consists of the memory, the
        // program counter, the run bit, and the AC. The input parameters consist of
        // the memory image and the starting address.

        PC = starting_address;
        while (run_bit) {
            instr = memory[PC];             // fetch next instruction into instr
            PC = PC + 1;                    // increment program counter
            instr_type = get_instr_type(instr);   // determine instruction type
            data_loc = find_data(instr, instr_type);  // locate data (–1 if none)
            if (data_loc >= 0)              // if data_loc is –1, there is no operand
                data = memory[data_loc];    // fetch the data
            execute(instr_type, data);      // execute instruction
        }

    }

    private static int get_instr_type(int addr) { ... }
    private static int find_data(int instr, int type) { ... }
    private static void execute(int type, int data) { ... }
}
```

Ciclo de Ejecución del Procesador

- Esta equivalencia entre el funcionamiento del hardware y el interprete, tiene implicaciones importantes en el mundo de la organización de la computadora. Si un diseñador diseña un lenguaje de maquina, L, para un nuevo sistema computacional, el puede consecuentemente decidir si quiere diseñar hardware que ejecute directamente el lenguaje L o escribir un interprete para interpretar los programas en L.
- Después de todo, un interprete divide la instrucción en una serie de pasos pequeños. Esto permite que una maquina con interprete sea mas simple y menos cara.

Hardware y Software Como Equivalentes.

- **Hardware y software son lógicamente equivalentes**. La diferencia es que hardware es algo tangible mientras software consiste de algoritmos.
- Lo importante es que cualquier operación que se puede hacer por software también puede ser fabricada directamente en hardware, y vice versa.
- Según las palabras de Karen Panetta: “Hardware, es solo software petrificado”.
- El criterio de cuando un diseñador decide que parte será echo hardware y que parte será emulada en software, depende de costo, velocidad, evolución de la tecnología, y otros factores.



Unidad de Control

Microprograma

El Invento del Microprograma

- En las primeras computadoras digitales, había solo dos niveles: le nivel ISA y el nivel de Circuito Lógico. Toda la programación se hacia a nivel ISA (un horror). Los circuitos eran complicados, difíciles de entender y construir.
- En los 1950s, Maurice Wilkes, un investigador de la Universidad de Cambridge, sugirió un diseño de una nueva maquina, pero de tres niveles, que simplificaría drásticamente el hardware. Esta maquina tendría un interprete fijo, llamado el microprograma, que ejecutaría programas a nivel ISA de una manera interpretada.
- Esto permitiría que que el hardware sea mas simple, porque solo tenias que ser diseñado para ejecutar el microprograma, con instrucciones simples y limitadas, y ya no ejecutar un programa a nivel ISA, con instrucciones mas complejas y mas cantidad de ellas.
- Fue tan dominante este concepto, que los CPUs de hoy en día todavía la usan.

El boom del Microprograma

- Durante los años, diseñadores se dieron cuenta que podían agregar nuevas instrucciones, osea mas “hardware”, con solo extendiendo el microprograma. Esto llevo a una explosión del set de instrucciones de las maquinas.
- Estas nuevas instrucciones no eran esenciales porque tenían equivalentes, pero aveces eran un poco mas rápidas y eran incluidas.
- Por ejemplo, la instrucción INC hace que se incremente un numero por 1. La equivalente es la instrucción ADD donde una entrada es el numero y la otra es el valor 1. Internamente, INC era un poquito mas rápida.

El boom del Microprograma

- Era tan fácil agregar nuevas instrucciones que instrucciones populares en la lista incluían:
  - Instrucciones para cómputos de arreglos.
  - Instrucciones para mover los programas en memoria.
  - Instrucciones para interrumpir el sistema cuando hay alguna entrada/salida.
  - Instrucciones para suspender programas y empezar otro (multitasking y process switching).
  - Instrucciones para procesar multimedia.

Beneficios del Microprograma

- La habilidad de corregir instrucciones que están incorrectamente implementadas, o superar deficiencias en el diseño del hardware.
- La oportunidad de agregar nuevas instrucciones con un costo mínimo, aun con maquinas ya existentes.
- Un diseño estructurado que permitía desarrollos mas eficientes y mejor documentación de instrucciones complejas.

Eliminación del Microprograma

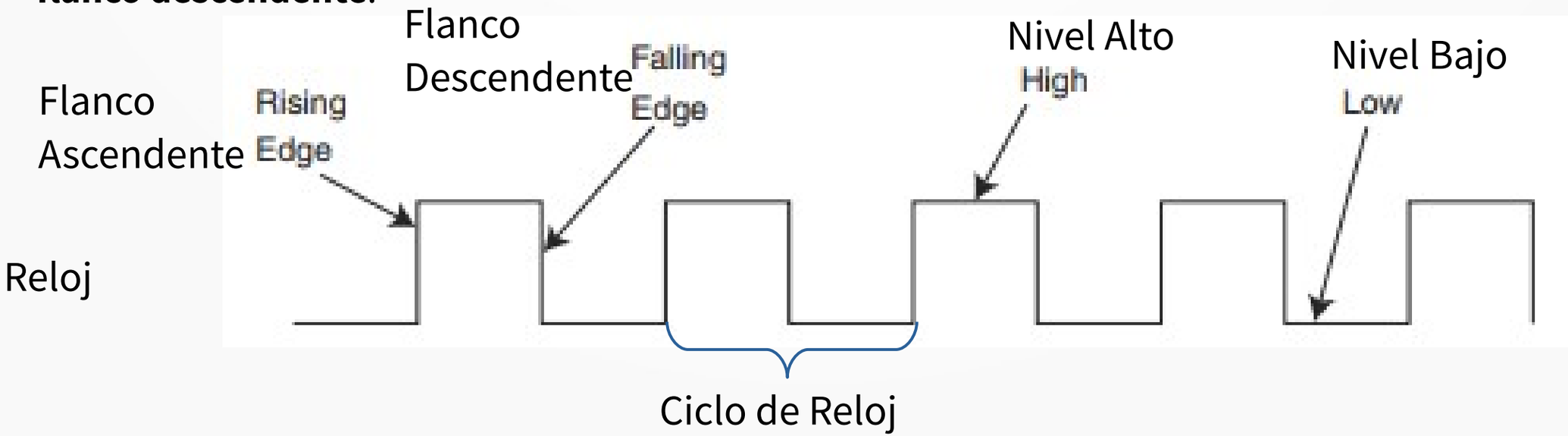
- Los microprogramas eventualmente se volvieron muy gordos, y eso causaba que se volvieran mas y mas lentos.
- Eventualmente unos investigadores se dieron cuenta que sacando el microprograma, reduciendo el set de instrucciones y que las instrucciones se ejecuten directo por hardware, la maquina podía ganar un montón de velocidad. Aquí nacieron las arquitecturas llamadas RISC (AyOCII).

Como Progresa el Tiempo en un CPU

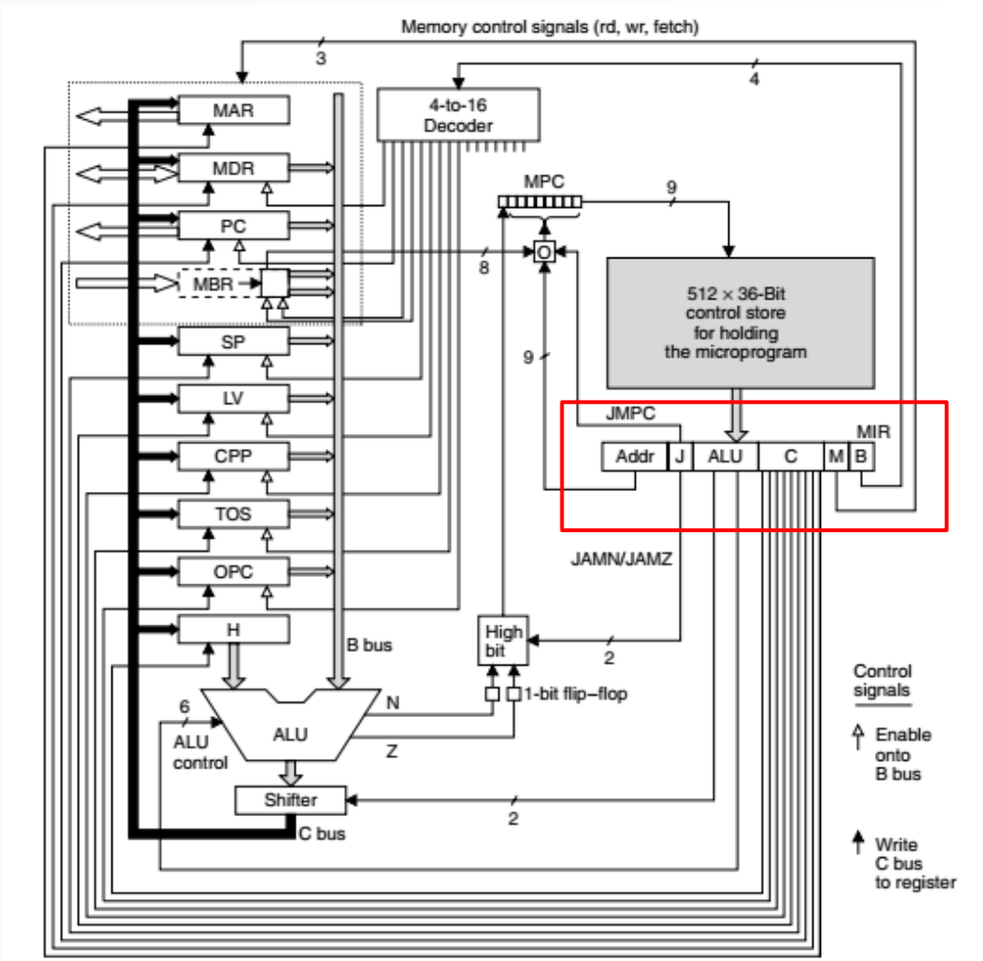
- El procesador puede calcular, leer/escribir de memoria, recibir/mandar un dato por algún bus, y muchas otras cosas mas, ¿pero como se llevan a capo estos procesos complexos a través del tiempo?
- Escencialmetne, hay dos formas que un circuito puede interactuar con el tiempo, te forma **sincronica** o de forma **asincronica**.
- Cuando un circuito es asincronico, eso significa que la salida del circuito cambia desde el momento que hay cambios en las entradas.
- En cambio, si un circuito es sincronico, eso significa que espera alguna señal externa que le indica cuando el cambio en la salida tienen que suceder. Esta señal externa se llama, el **reloj**.

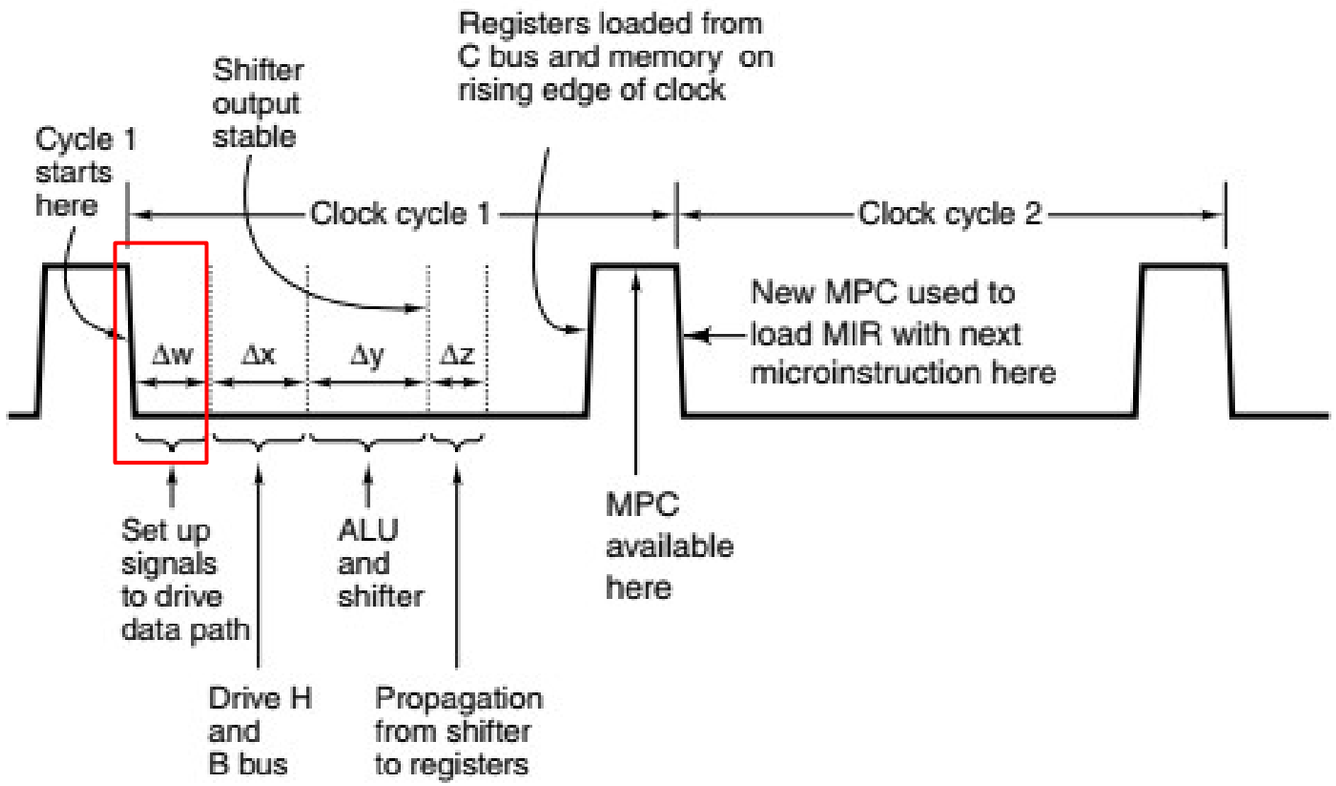
Como Progresa el Tiempo en un CPU

- El reloj es un circuito que emite una serie de pulsos con un ancho preciso y un intervalo preciso entre pulsos consecutivos. El tiempo completo, que incluye el pulso y el intervalo de espera, se llama **Ciclo de Reloj** (Clock Cycle).
- Hoy en día, los ciclos de reloj están comúnmente en los GHz, osea, en los mil millones de ciclos de reloj cada segundo.
- Como explicamos, el reloj se usa para activar un circuito sincronico, y hay cuatro posibilidades durante el ciclo de reloj para hacer esta activación: **nivel alto, nivel bajo, flanco ascendente, y flanco descendente**.



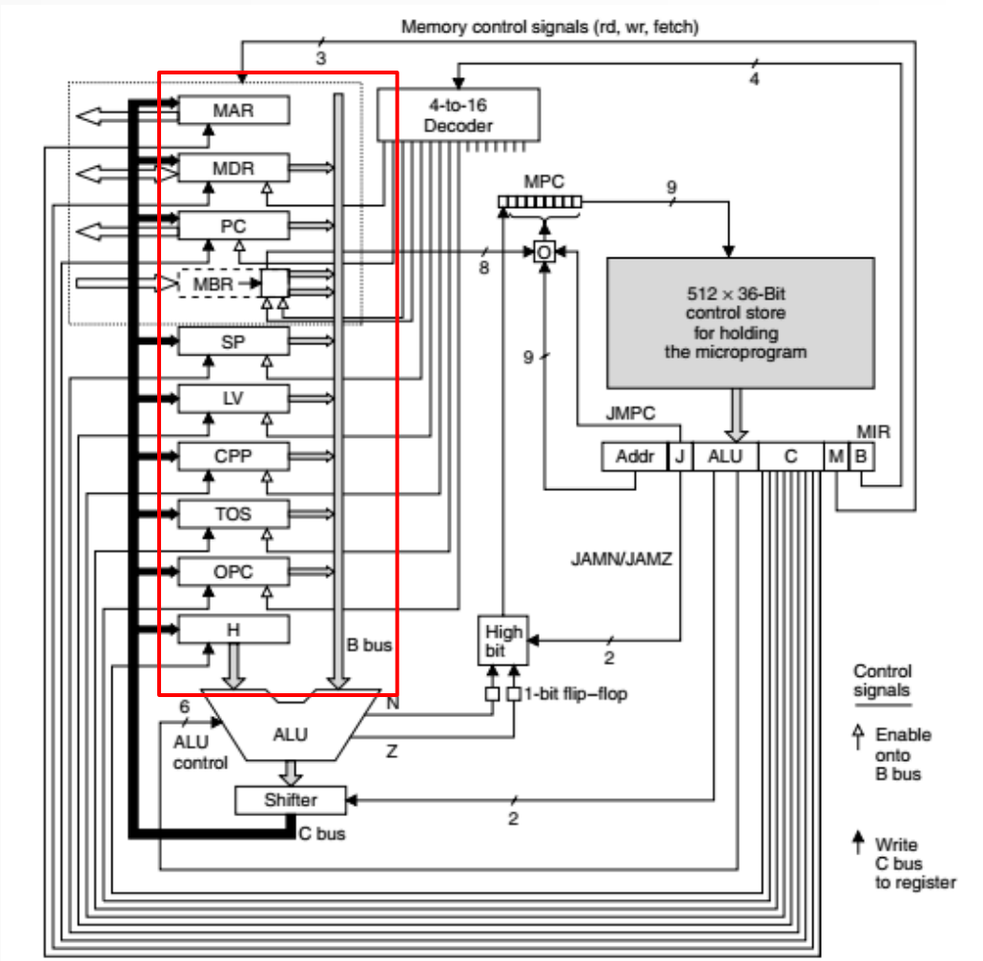
Ejecución de una micro-instrucción en la MIC-1

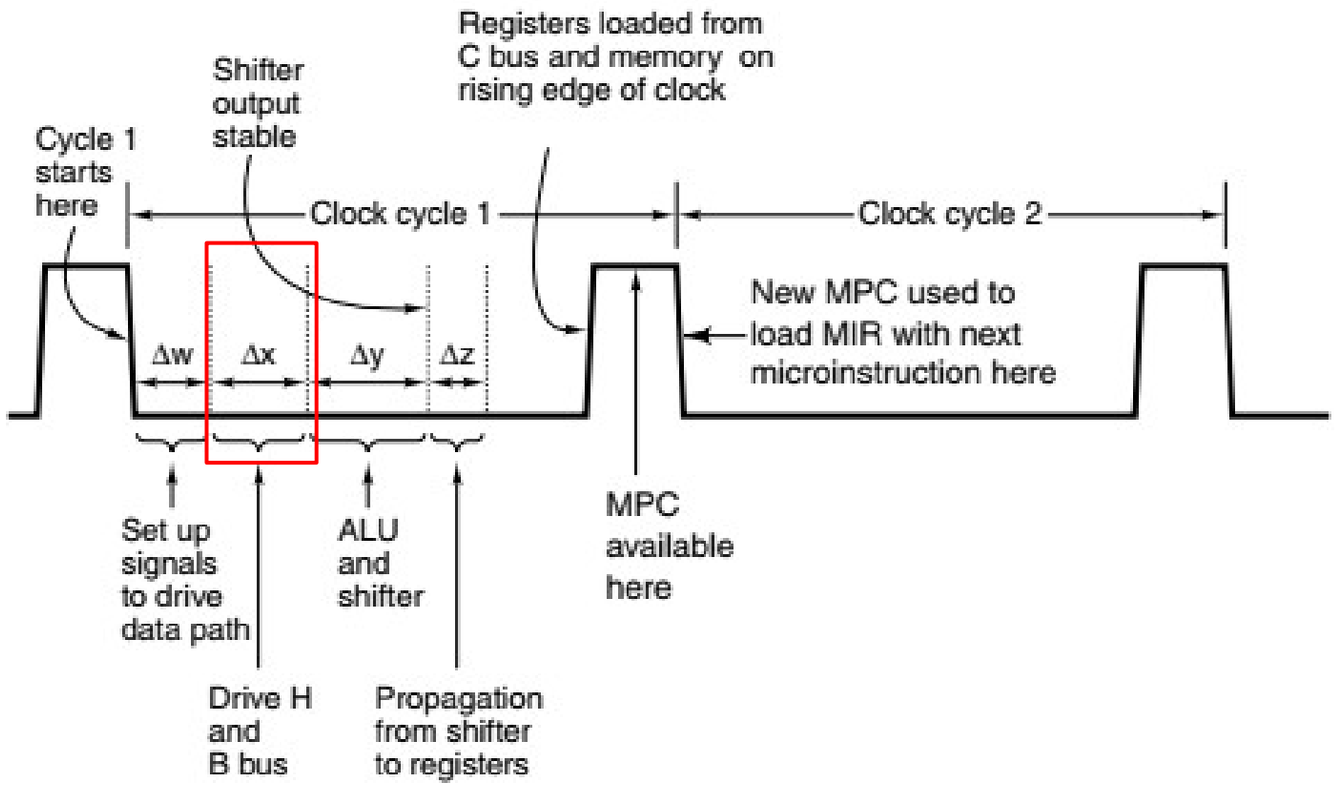




En el comienzo del ciclo, el MIR es establecido (las señales de control).

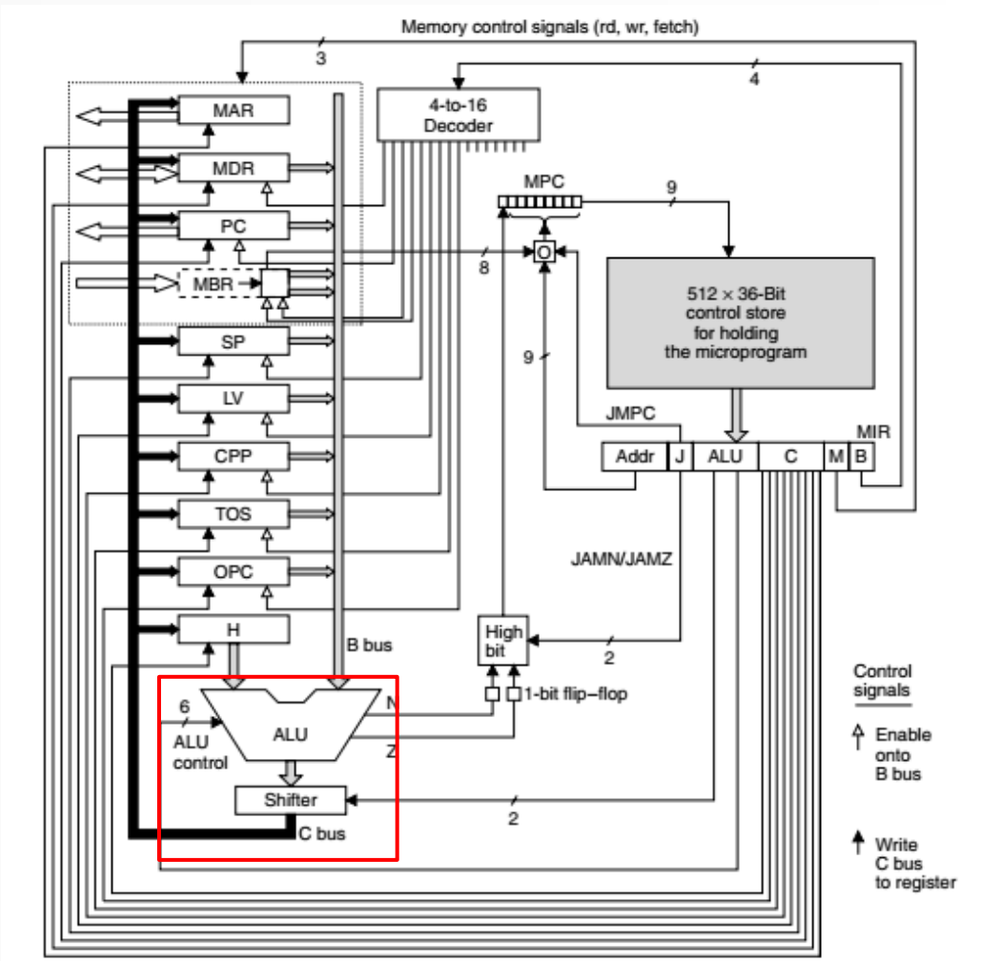
Ejecución de una micro-instrucción en la MIC-1

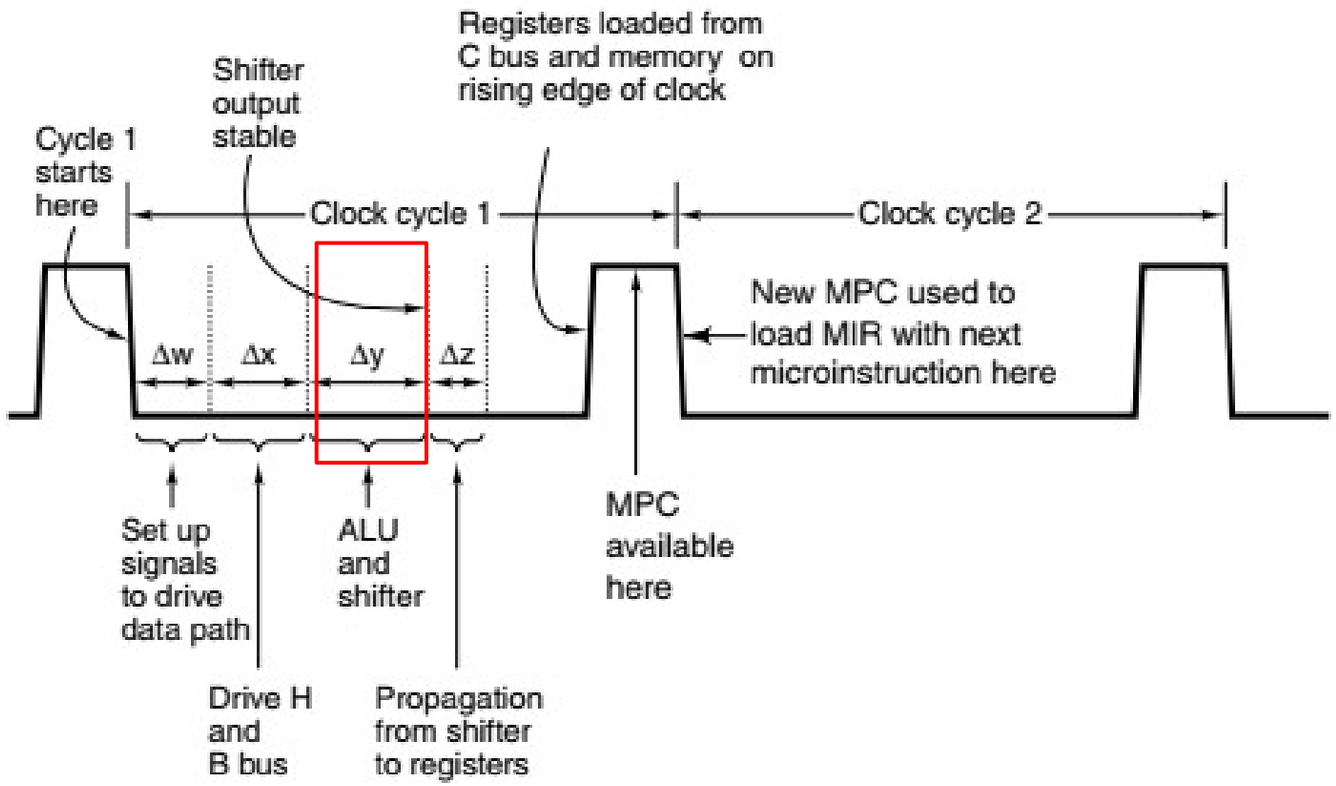




Se prepara el bus H y el B con datos para el ALU.

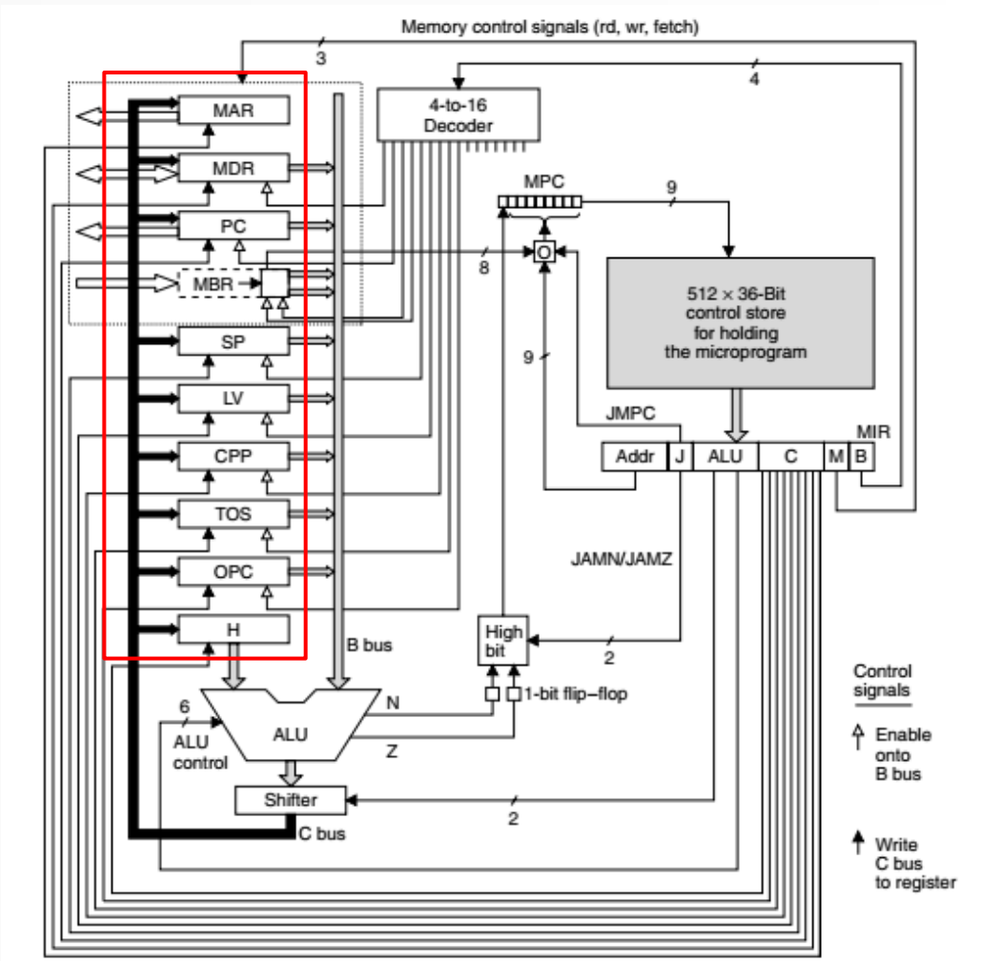
Ejecución de una micro-instrucción en la MIC-1

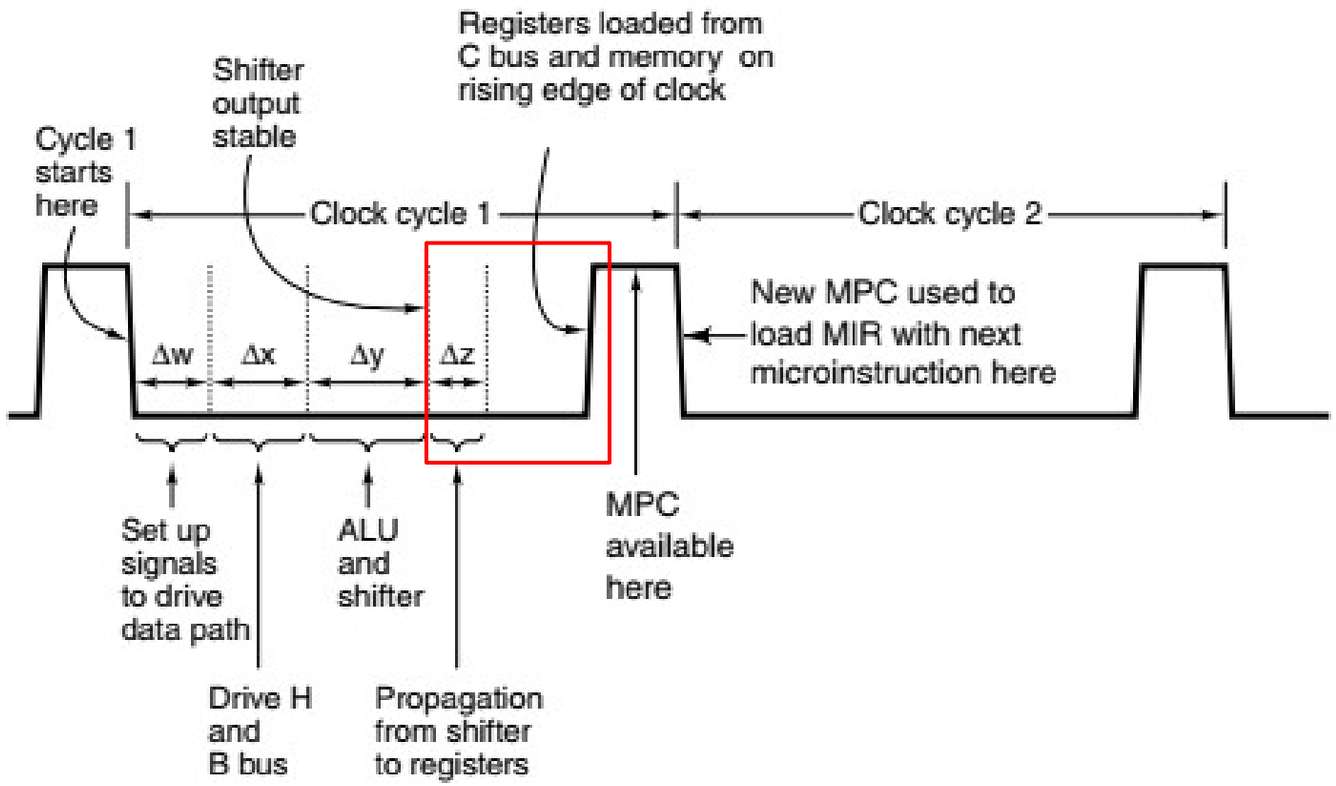




El ALU (Unidad Aritmética Lógica) y el Shifter (Desplazador) ejecutan su función.

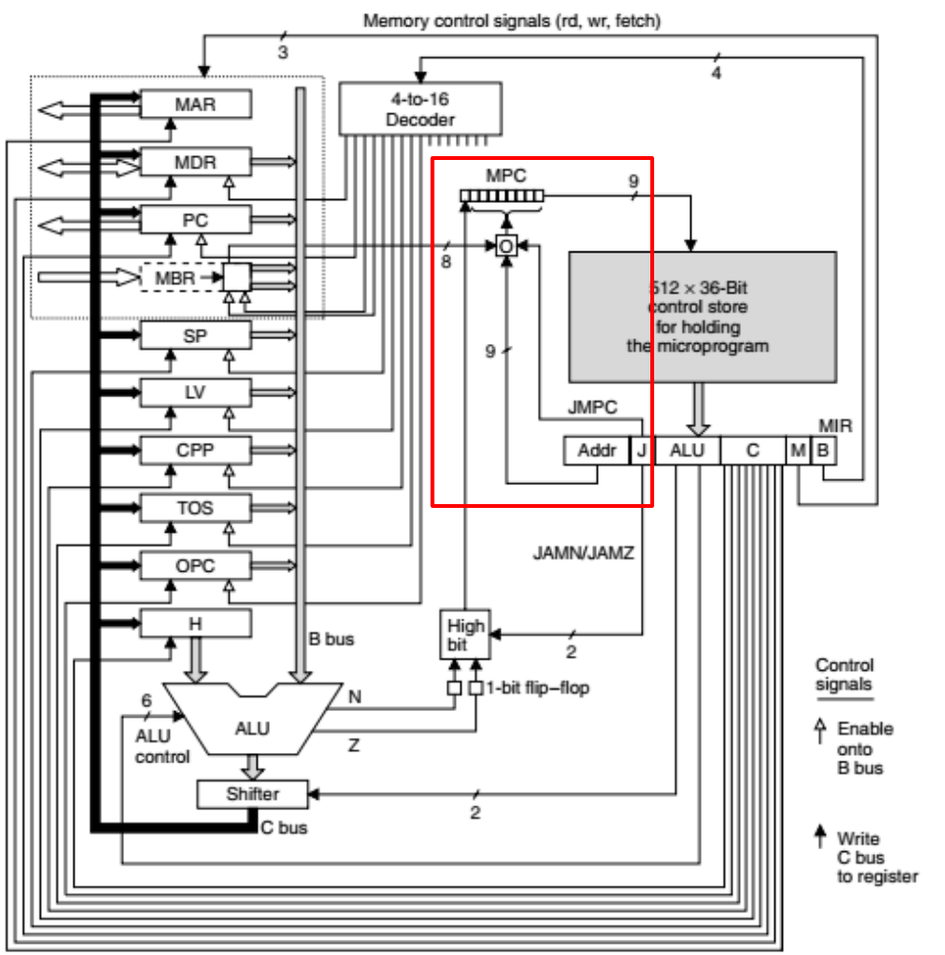
Ejecución de una micro-instrucción en la MIC-1

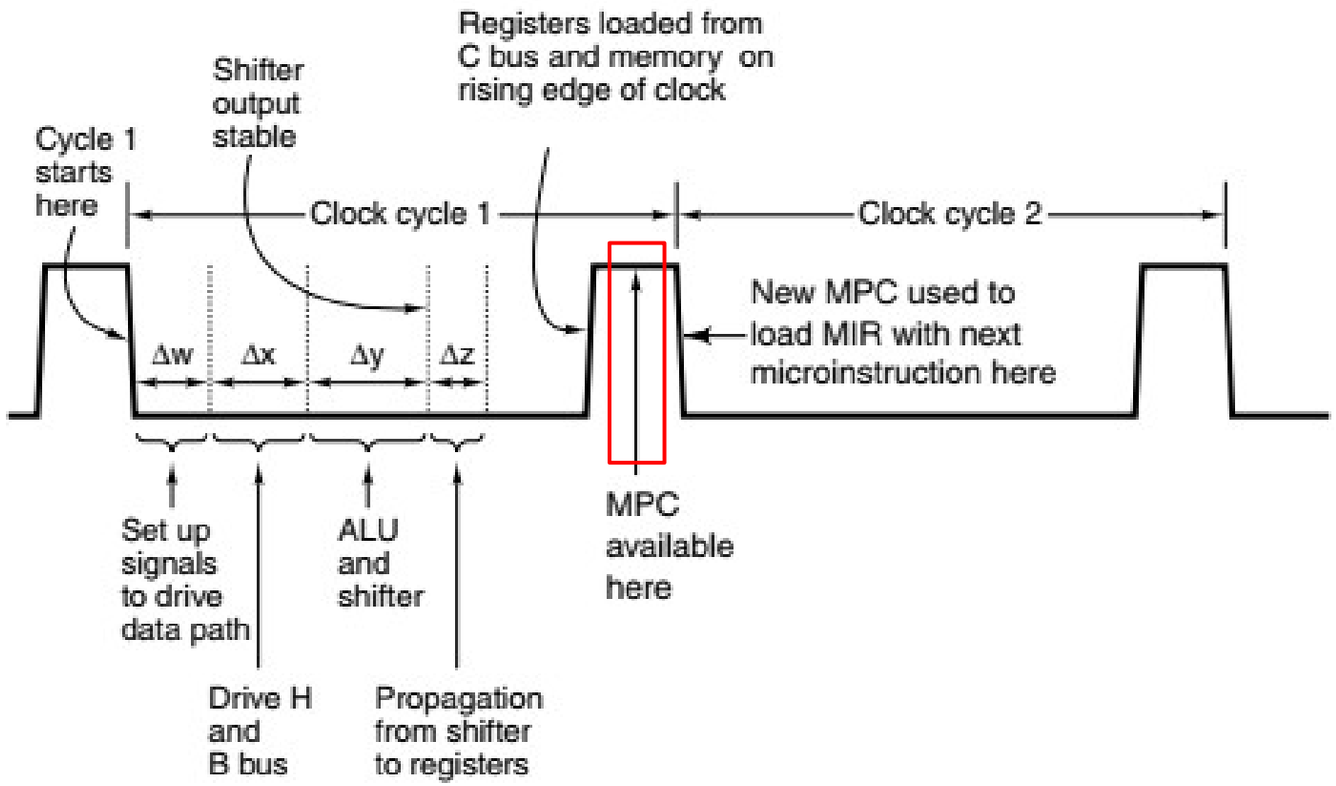




Los resultados son propagados al bus C, y en el flanco ascendente, son guardados a los registros.

Ejecución de una micro-instrucción en la MIC-1

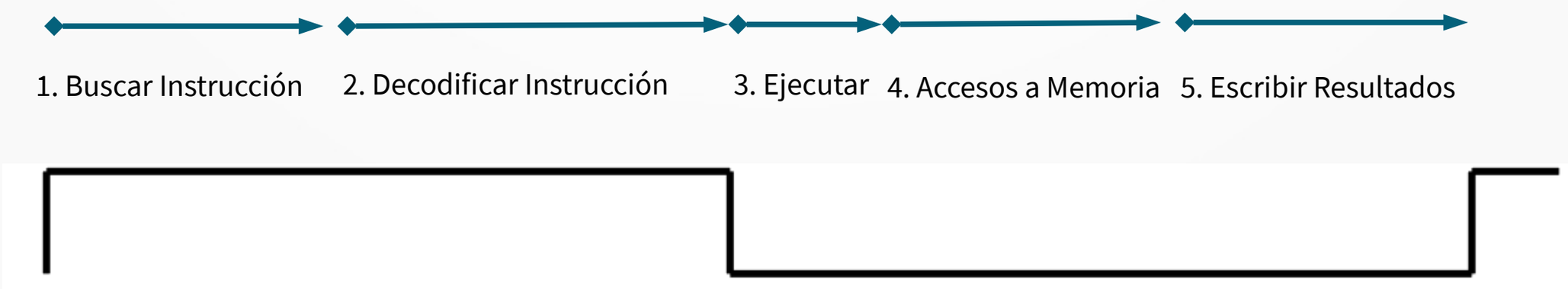




Según el valor del MPC y el MIR, se prepara para ejecutar la próxima micro-instrucción Cada micro-introducción se ejecuta en **un solo ciclo**.

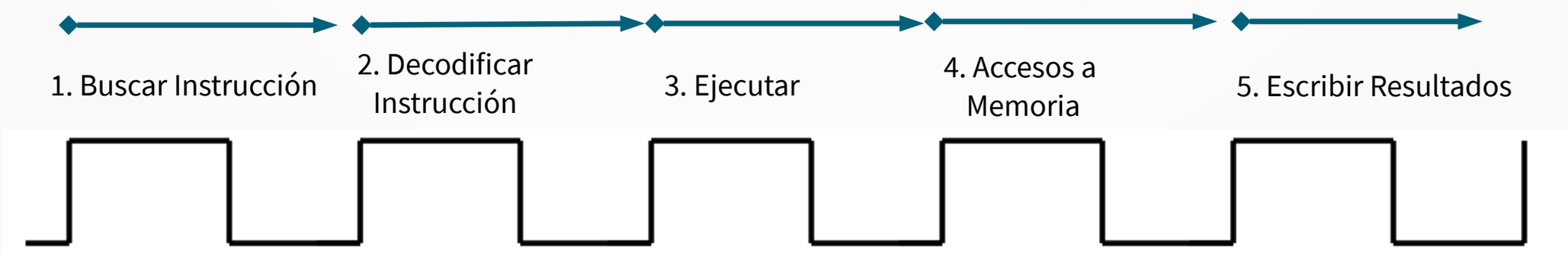
CPU de Ciclo Único vs Múltiples Ciclos

- Como pudimos ver, el ciclo de reloj es muy importante para definir como se lleva cabo la ejecución de una instrucción.
- En términos generales, si tenemos un CPU donde el ciclo de ejecución completo de una instrucción se lleva a cabo durante un ciclo de reloj, este CPU se dice que es de **Ciclo Único**.



CPU de Ciclo Único vs Múltiples Ciclos

- En cambio, si tenemos un CPU donde cada etapa del ciclo de ejecución se lleva a cabo durante un cliclo de reloj, este se dice que es de **Múltiples Ciclos** (y posiblemente Segmentado).



El Concepto de la Palabra Arquitectura

- En los 1950s, en el mismo tiempo del la creación del microprograma, IBM reconoció que soportando una misma familia de maquinas, todas que ejecutaban las mismas instrucciones, tenia muchas ventajas.
- IBM introdujo la palabra arquitectura para describir este nivel de compatibilidad. Esto significaba que una nueva computadora podía tener el mismo ISA aunque su implementación (Nivel 1 y Nivel 0) difiera.

Lecturas

- Lecturas recomendadas:
- Tanenbaum, 2013: Capítulos 1.1.3 (sin la parte del sistema operativo), 2.1, 2.1.1, 2.1.2
- Tanenbaum, 2000: Capítulos 1.1.3 (sin la parte del sistema operativo), 2.1, 2.1.1, 2.1.2