# Integrador N° 0

Tomás Vidal (69854/4)
*Arquitectura de Computadoras*
*Facultad de Ingeniería, UNLP, La Plata, Argentina.*
*4 de Abril, 2024.*

## I. Introducción

El desarrollo de la computadora Integrador N° 0 se basó en la idea de implementar un cronómetro utilizando displays de 7 segmentos. Este informe detalla las decisiones de arquitectura tomadas y su justificación.

## II. Objetivo

El objetivo principal fue diseñar una arquitectura simple que permitiera la implementación de un cronómetro mediante la manipulación de registros de memoria y la interacción con displays de 7 segmentos.

## III. Instrucciones

Para lograr este objetivo, se diseñaron cuatro instrucciones básicas:

- **LDA (Load Data Address):** Carga un dato en una dirección específica de la memoria de datos.
- **IDA (Increment Data Address):** Incrementa el valor almacenado en una dirección de memoria específica.
- **LDO (Load Data Output):** Carga el dato almacenado en una dirección de memoria en el registro de salida para su visualización en el display de 7 segmentos.
- **CJ (Conditional Jump):** Realiza un salto condicional en el programa basado en la comparación de dos valores almacenados en la ALU.

Estas instrucciones se diseñaron para permitir operaciones básicas de carga, manipulación y visualización de datos, así como control de flujo en el programa.

## IV. Arquitectura de la Computadora

La arquitectura de la computadora se compone de varios bloques funcionales interconectados, las interconexiones se pueden ver en el diagrama de bloques 1.

### *IV-A. PC (Program Counter)*

El PC está compuesto por cuatro registros con flip flops tipo D de 4 bits cada uno. Permite cargar una dirección específica y avanzar en la memoria de programa.

### *IV-B. PROGRAM MEMORY (Memoria de Programa)*

La memoria de programa es una EPROM de 32K con una palabra de tamaño de 8 bits. Almacena las instrucciones del programa y su salida se conecta al IR.

### *IV-C. IR (Instruction Register)*

El IR almacena temporalmente la instrucción actual y su salida se conecta al bus principal.

### *IV-D. ALU (Arithmetic Logic Unit)*

La ALU realiza operaciones aritméticas y lógicas, incluyendo la suma y comparación de valores. Su salida se utiliza para controlar el flujo del programa.

### *IV-E. MAR (Memory Address Register)*

El MAR almacena temporalmente la dirección de memoria a la que se accederá. Controla la lectura y escritura en la memoria de datos.

### *IV-F. DATA MEMORY (Memoria de Datos)*

La memoria de datos consiste en un banco de registros con flip flops tipo D. Almacena los datos utilizados por el programa.

### *IV-G. OUTPUT REGISTERS (Registros de Salida)*

Los registros de salida almacenan los datos que se mostrarán en los displays de 7 segmentos para la visualización del cronómetro.

### *IV-H. CPU (Central Processing Unit)*

La CPU es el núcleo de la computadora y se encarga de ejecutar las instrucciones del programa. Está compuesta por varios componentes clave que trabajan en conjunto para decodificar y ejecutar las instrucciones.

*IV-H.1. Máquina de Estados Finitos de Moore:* La CPU utiliza una máquina de estados finitos de Moore para decodificar las instrucciones. Esta máquina tiene estados que representan las diferentes fases de ejecución de una instrucción, como decodificación, ejecución y finalización. La transición entre estados está determinada por la instrucción actual y el estado actual de la máquina. Para lograr esta máquina de estados finitos se empleó una EPROM de 32K como *lookup table*. La máquina se puede apreciar mejor en la figura 2.

*IV-H.2. Decodificación de Instrucciones:* Las instrucciones se decodifican utilizando los dos bits más significativos de la instrucción actual que entran a la EPROM (corresponden a los 2 bits más significativos en la entrada), y que indican el tipo de operación a realizar (LDA, IDA, LDO o CJ). Estos bits se utilizan para activar los diferentes caminos de datos y señales de control necesarias para ejecutar la instrucción correctamente. Los 2 bits menos significativos de la entrada de la EEPROM provienen de la lógica booleana y del estado de la máquina. Y en el caso particular de CJ que requiere de la comparación de la ALU se emplea el 5to bit de la EMPROM para decodificar la instrucción.

*IV-H.3. Generación de Señales de Control:* La CPU genera una serie de señales de control que coordinan las operaciones de los diferentes bloques de la computadora. Estas señales incluyen señales de lectura/escritura para la memoria, señales de habilitación para los registros y señales de control para la ALU.

*IV-H.4. Control del Flujo del Programa:* Además de ejecutar instrucciones individuales, la CPU también maneja el control del flujo del programa. Esto se logra mediante la generación de señales de salto condicional (CJ) que modifican la dirección de ejecución del programa en función de ciertas condiciones, como el resultado de una comparación realizada por la ALU.

*IV-H.5. Lógica de Decodificación:* La lógica de decodificación se implementó utilizando técnicas digitales, como mapas de Karnaugh y álgebra booleana. Esta lógica toma los bits de la instrucción actual como entrada y genera las señales de control necesarias para ejecutar la instrucción correspondiente mediante computertas lógicas.

En resumen, la CPU coordina todas las operaciones de la computadora, desde la decodificación de instrucciones hasta el control del flujo del programa. Su diseño se basa en una máquina de estados finitos de Moore y utiliza lógica de decodificación para generar las señales de control adecuadas.

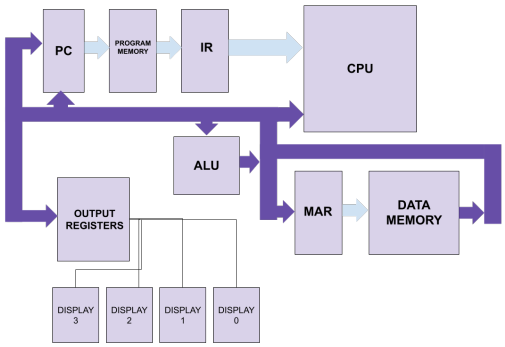


Fig. 1. Diagrama de bloques de la computadora

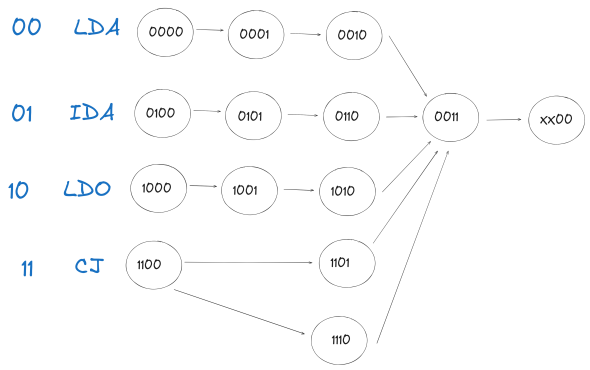


Fig. 2. Diagrama de la máquina de estados finitos de Moore

## V. Programas

### V-A. Cronómetro

El programa del cronómetro, almacenado en el archivo *"cronometro.bin"*, se compone de una serie de instrucciones diseñadas para contar progresivamente el tiempo. Utiliza las instrucciones LDA, IDA, LDO y CJ para cargar datos, manipular registros y controlar el flujo del programa. El programa se basa en sumar 1 al registro 0 hasta llegar a 9 y luego suma 1 al registro 1 y hace 0 el registro 0 y luego salta a la instrucción 0, y así sucesivamente con el resto de registros, logrando tener un cronómetro. Lo que realmente determina que puede contar el tiempo es la cantidad de ciclos de reloj que le toma actualizar el valor en el display, y obviamente la velocidad del reloj principal. Se calculó que para una velocidad de 140Hz aproximadamente se tiene una buena constante de tiempo que hace que el primer display muestre centenas de milisegundos, el segundo display los segundos, el tercero las decenas de segundos y el cuarto display las centenas de segundos.

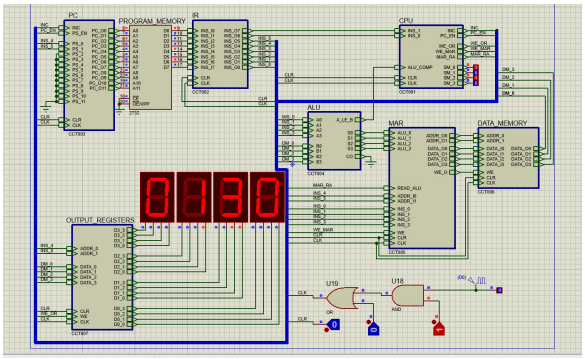

Fig. 3. Diagrama de la máquina de estados finitos de Moore

## VI. Conclusiones

La arquitectura de la computadora Integrador N° 0 se diseñó de manera que fuera simple y eficiente para la implementación del cronómetro. Las decisiones de diseño se basaron en la necesidad de cumplir con los requisitos funcionales del proyecto, permitiendo la manipulación de datos y el control del flujo del programa de manera efectiva.

## VII. Consejos para la prueba en Proteus

Se dejaron algunas llaves que permiten la fácil manipulación de los ciclos del reloj para poder observar el estado de la máquina de estados si se desea iterar y observar los mismos, estas salidas se pueden ver como pines de salida de "prueba" en el CPU (SM_0, SM_1, SM_2, SM_3).