

# Universidad Autónoma de Zacatecas

## Unidad Académica de Ingeniería Eléctrica Ingeniería en Robótica y Mecatrónica



Microcontroladores

Proyecto Microprocesador

Docente: Remberto Sandoval Aréchiga

· Alumnos:

Grupo: 5°"A"

- Marco Antonio Barrón Ramírez
- Gerardo Andrés Vanegas Solís
- Paloma Guadalupe Rocha Castro
- Anahí Verenice Isais Torres

25 / Noviembre / 2020

# Resumen

La práctica consistió en conocer de manera general lo que es un microprocesador, el diseño de un hardware en una arquitectura Harvard. Seguimos una metodología de Top-Down, empieza con una caja negra, la cual literalmente es un cuadro en la cual se identificaran las entradas y las salidas. Se fue bajando en jerarquía, continuando con funciones específicas. El profesor nos facilitó lo que fue el set de instrucciones y la caja negra con sus respectivas entradas y salidas al igual que los bits de cada una, a partir de ellas se realizó un microprocesador personalizando su arquitectura. Se describió la funcionalidad de cada uno de los componentes, como salidas, entrada y que transportaría cada una de las señales.

El microprocesador es de gran importancia en nuestra carrera, es definido como el cerebro de un sistema, maneja las instrucciones y las operaciones lógicas y matemáticas. El microprocesador ha ido mejorando a lo largo de los años, siendo cada vez más eficientes.

En arquitectura de computadoras, 8 bits es usado para describir enteros, direcciones de memoria o para referirse a una arquitectura de CPU y ALU basadas en registros, bus de direcciones o bus de datos.

# Índice

|                                   |          |
|-----------------------------------|----------|
| <b>Resumen</b>                    | <b>2</b> |
| <b>Índice</b>                     | <b>2</b> |
| <b>Introducción</b>               | <b>3</b> |
| Unidad de control                 | 3        |
| La unidad aritmética lógica (ALU) | 4        |
| Unidad de memoria                 | 4        |
| <b>Arquitectura</b>               | <b>5</b> |
| Microprocesador_1 Caja Negra      | 5        |
| Entradas                          | 5        |
| Salidas                           | 6        |
| Microprocesador_1 Caja Blanca     | 6        |
| Unidad de Control UC              | 8        |
| Entradas                          | 9        |
| Salidas                           | 9        |
| Descripción                       | 10       |
| Contador de Programa PC           | 10       |
| Entradas                          | 10       |
| Salidas                           | 10       |
| Descripción                       | 11       |

|                                    |           |
|------------------------------------|-----------|
| Registro de Instrucciones RI       | 11        |
| Entradas                           | 11        |
| Salidas                            | 11        |
| Descripción                        | 12        |
| Registro de Datos RD               | 12        |
| Entradas                           | 12        |
| Salidas                            | 13        |
| Descripción                        | 14        |
| Unidad Aritmética Lógica ALU       | 14        |
| Entradas                           | 14        |
| Salidas                            | 14        |
| Descripción                        | 14        |
| <b>Implementación y Simulación</b> | <b>15</b> |
| <b>Conclusiones</b>                | <b>15</b> |
| <b>Referencias</b>                 | <b>15</b> |
| <b>Apéndice</b>                    | <b>16</b> |
| Archivo TOP Microprocesador_1      | 17        |
| Unidad de Control UC               | 19        |
| Contador de Programa PC            | 21        |
| Registro de Instrucciones RI       | 22        |
| Registro de Datos RD               | 23        |
| Unidad Aritmética Lógica ALU       | 24        |

## Introducción

El microprocesador es un circuito electrónico que actúa como Unidad Central de Proceso (CPU) de una computadora. Es un circuito microscópico constituido por millones de transistores integrados en una única pieza plana de poco espesor. El microprocesador se encarga de realizar todas las operaciones de cálculo y de controlar lo que pasa en la computadora recibiendo información y dando órdenes para que los demás elementos trabajen.

Las tres partes principales de un procesador son:

### Unidad de control

Controla las operaciones de todas las partes del microprocesador, pero no lleva a cabo ninguna operación de procesamiento de datos. Gestiona y coordina todas las unidades del

ordenador, obtiene las instrucciones provenientes de la memoria, se comunica con los dispositivos de entrada y de salida y no procesa ni guarda datos.

### La unidad aritmética lógica (ALU)

Tiene dos subcategorías llamados sección aritmética y sección lógica. La primera realiza operaciones aritméticas (suma, resta, multiplicación, división), las operaciones complejas se realizan utilizando las operaciones básicas. La sección lógica realiza operaciones lógicas, como comparar, seleccionar, emparejar o fusionar datos.

### Unidad de memoria

Es la memoria utilizada para almacenar las programaciones y datos del CPU. Cuanta más memoria tenga el CPU, más programas puede arrancar al mismo tiempo, con más datos para gestionar. Puede almacenar instrucciones, datos o resultados intermedios. Esta unidad nutre de información a otras unidades del CPU que las necesiten. Almacena toda la información y las instrucciones cuyo procesamiento es necesario, almacena los resultados intermedios de procesamiento, guarda los resultados finales de procesamiento antes de que estos salgan a un dispositivo de salida, todos los dispositivos de entrada y salida se comunican o transmiten a través de la memoria principal.

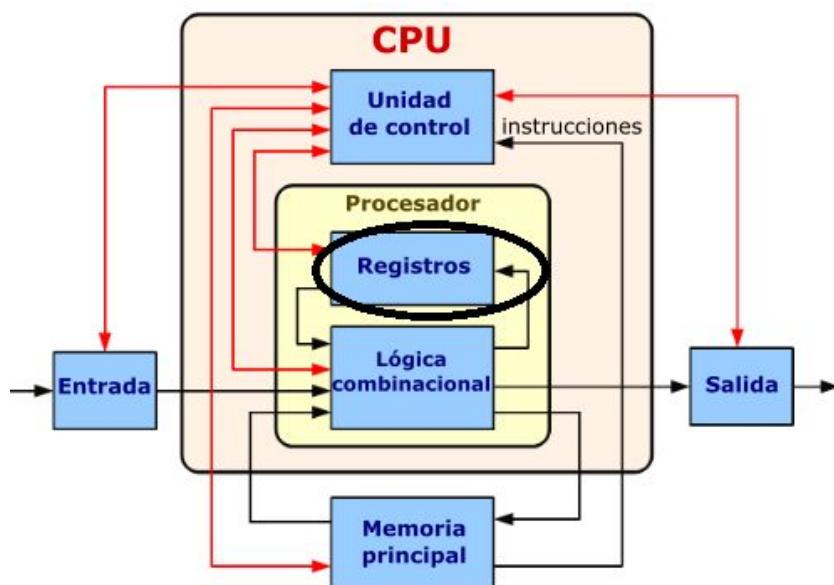


Ilustración 1 Componentes principales de un microprocesador

El Intel 8080 fue un microprocesador temprano diseñado y fabricado por Intel. La CPU fue lanzada en abril de 1974. Corría a 2 MHZ, y generalmente se le considera el primer diseño verdaderamente útil.

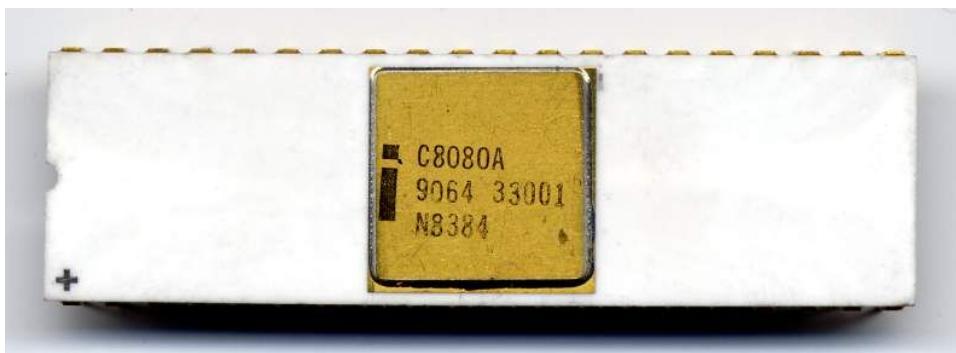


Ilustración 2 Microprocesador Intel 8080

## Arquitectura

### Microprocesador\_1 Caja Negra



Ilustración 3 Diagrama de caja negra de un microprocesador

### Entradas

| Nombre de la señal | Tamaño (Bits) | Descripción   |
|--------------------|---------------|---|
| Instruction        | 9             | Bus de instrucciones de 9 bits que se encarga de mandar llamar las instrucciones desde el set de instrucciones.     |
| Data_In_Bus        | 8             | Bus de datos de entrada de 8 bits que manda llamar a los datos almacenados en la memoria.                           |
| i_Clk              | 1             | Señal de reloj que genera pulsos en intervalos constantes, los cuales se encargan de sincronizar las instrucciones. |
| i_Rst              | 1             | Señal que regresa al microprocesador a sus valores iniciales o por defecto.   |

Ilustración 4 Tabla de entradas con descripción de cada una de ellas.

## Salidas

| Nombre de la señal      | Tamaño (Bits) | Descripción   |
|-------------------------|---------------|---|
| Address_Instruction_Bus | 8             | Bus de salida de 8 bits que se encarga de dirigir las instrucciones a la unidad de control del programa.  |
| Address_Data_Bus        | 8             | Bus de datos de 8 bits que se encarga de avisar a nuestra memoria de datos, los datos que necesita el microprocesador para poder realizar alguna instrucción. |
| DataOut_Bus             | 8             | Bus de datos de salida de 8 bits que lleva los resultados obtenidos en el proceso a la memoria de datos.  |

Ilustración 5 Tabla de salidas con descripción

## Microprocesador\_1 Caja Blanca

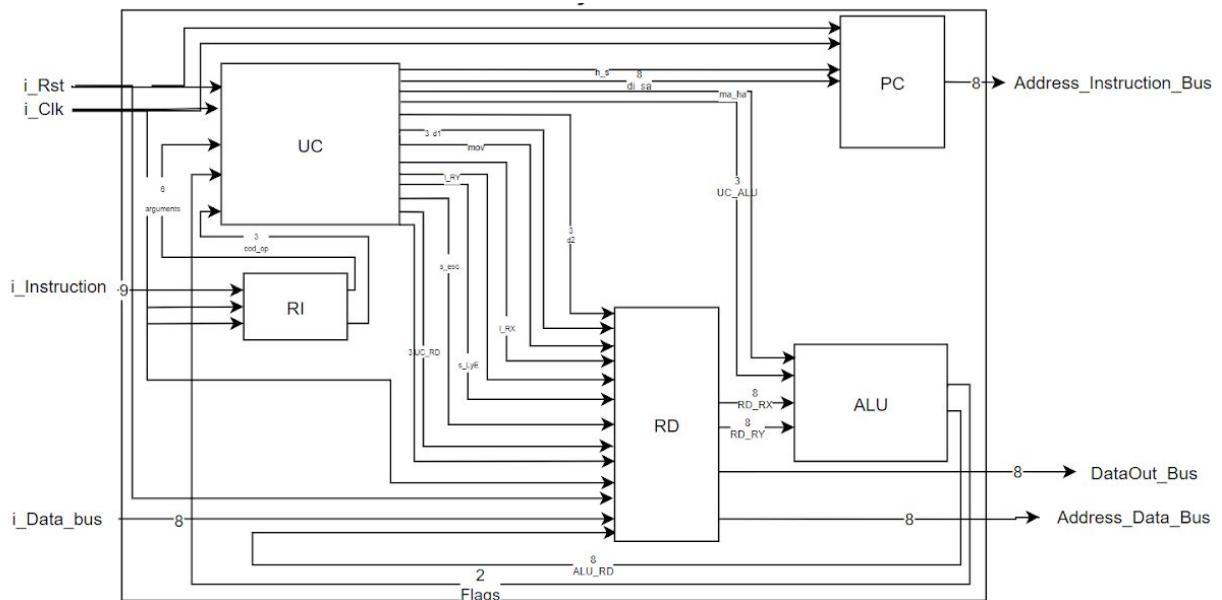
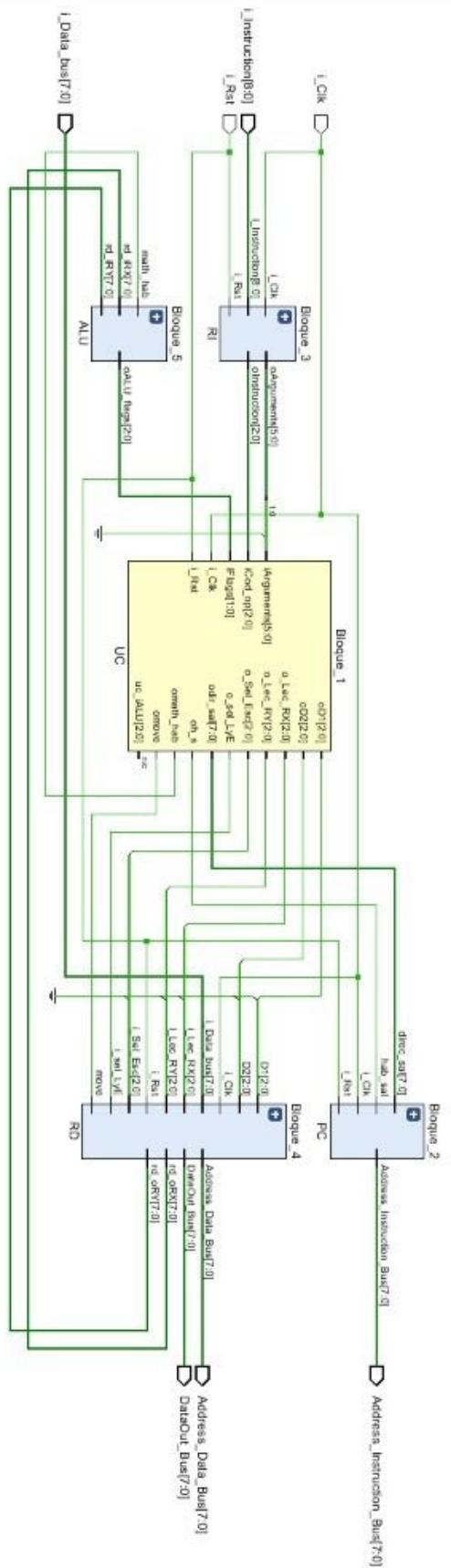


Ilustración 6 Diagrama de caja blanca microprocesador



| Bloque | Funcionamiento  |
|--------|---|
| PC     | Contador de programa.   |
| RI     | Contiene la instrucción que se está ejecutando en cada momento. El PC activa el movimiento de datos después de procesar la instrucción.   |
| RD     | En ellos se almacenan los datos u operandos que intervienen en una instrucción, antes de la realización de las operaciones.   |
| ALU    | Unidad aritmética-lógica es la encargada de realizar las operaciones, las cuales son suma, resta, multiplicación y división además de también trabajar con compuertas lógicas( and, or, not)                        |
| Flags  | Registros de memoria en los que se deja constancia de algunas condiciones que se dieron en la ultima operación realizada.   |
| UC     | Unidad de control, indica a PC la dirección, realiza la instrucción pedida, en caso de que sea una operación, se dirige a la ALU, cuando ya ha terminado la instrucción indica a PC que se dirija a Memory_Address. |

Ilustración 7 Tabla de descripción de bloques de caja blanca

## Unidad de Control UC

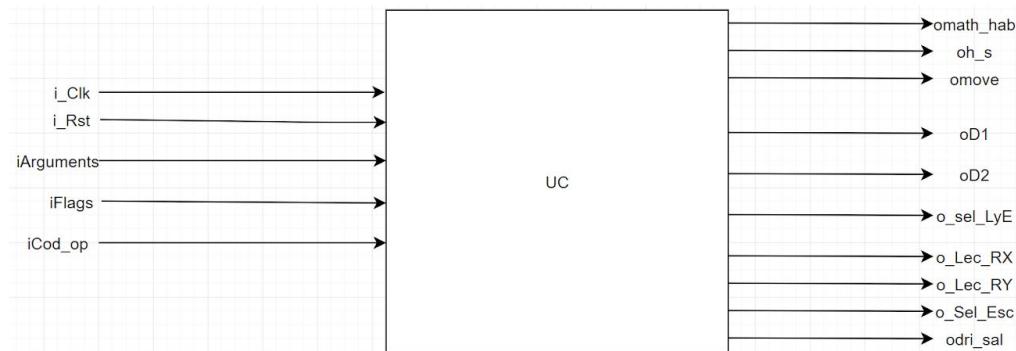


Ilustración 9 Diagrama Unidad de Control

## Entradas

| Nombre de la señal | Tamaño (Bits) | Descripción   |
|--------------------|---------------|---|
| iArguments         | 8             | señal con 6 bits por ambos argumentos que existen   |
| i_Clk              | 1             | Señal de reloj que genera pulsos en intervalos constantes, los cuales se encargan de sincronizar las instrucciones. |
| i_Rst              | 1             | Señal que regresa al microprocesador a sus valores iniciales o por defecto.   |
| iFlags             | 3             | Bus de dirección que le indica a la UC el estado de las operaciones realizadas por la ALU                           |
| iCod_op            | 3             | Señal que conduce los 3 bits mas significativos de la operación (indica la operación que se realizará)              |

Ilustración 10 Tabla de entradas con descripción

## Salidas

| Nombre de la señal | Tamaño (Bits) | Descripción  |
|--------------------|---------------|--|
| omath_hab          | 1             | Cuando ya se decodifica en la UC esta manda a la ALU el habilitador en la que en efecto es math          |
| oh_s               | 1             | habilitador para continuar con el conteo de instrucciones  |
| omove              | 1             | habilitador de un bit, instrucción move que ya fue decodificada en Unidad de Control                     |
| oD1                | 3             | dirección a la sala a la que se hará el movimiento   |
| oD2                | 3             | dirección desde la que se hará el movimiento   |
| o_sel_LyE          | 1             | indica si habrá escritura o lectura, en caso de ser lectura será 0 y en caso de que sea escritura será 1 |
| o_LEC_RX           | 3             | dirección de memoria en la que se va a leer o escribir   |
| o_LEC_RY           | 3             | dirección de memoria en la que se va a leer o escribir   |
| odri_sel           | 8             | entrada que proviene de la UC, la cual es un direccionamiento de salida                                  |
| o_Sel_Esc          | 3             | selecciona el espacio de memoria en el que se va a escribir  |

Ilustración 11 Tabla de salidas con descripción

## Descripción

La unidad de control se encarga como su nombre lo dice de controlar los componentes del microprocesador, es decir, es la encargada de buscar las instrucciones en la memoria, decodificarlas y ejecutarlas.

Unidad de control, indica a PC la dirección, realiza la instrucción pedida, en caso de que sea una operación, se dirige a la ALU, cuando ya ha terminado la instrucción indica a PC que se dirija a Memory\_Address.

## Contador de Programa PC

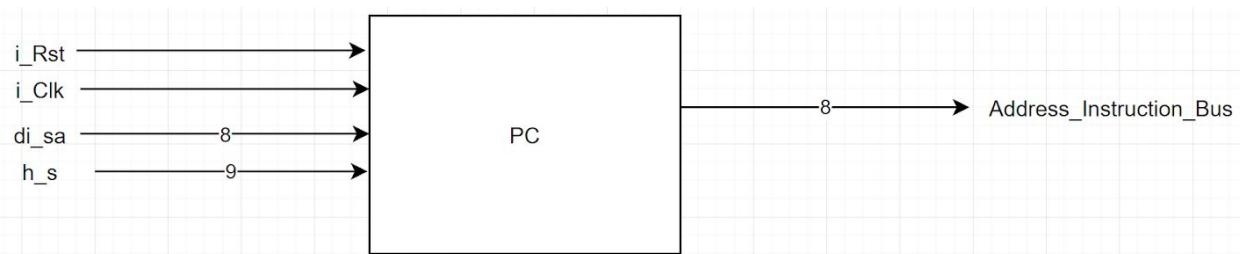


Ilustración 12 diagrama contador de programa

## Entradas

| Nombre de la entrada | Tamaño (Bits) | Descripción   |
|----------------------|---------------|---|
| i_Rst                | 1             | Señal que regresa al microprocesador a sus valores iniciales o por defecto.   |
| i_Clk                | 1             | Señal de reloj que genera pulsos en intervalos constantes, los cuales se encargan de sincronizar las instrucciones. |
| di_sa                | 8             | entrada que proviene de la UC, la cual es un direccionamiento de salida   |
| h_s                  | 1             | habilitador para continuar con el conteo de instrucciones   |

Ilustración 13 tabla de entradas con descripción

## Salidas

| Nombre de la señal      | Tamaño (Bits) | Descripción  |
|-------------------------|---------------|--|
| Address_Instruction_Bus | 8             | Bus de salida de 8 bits que se encarga de dirigir las instrucciones a la unidad de control del programa. |

Ilustración 14 tabla de salidas con descripción

## Descripción

El PC (contador de programa) es un registro en el microprocesador que almacena la dirección de la última instrucción leída. De esta manera se puede saber cuál es la siguiente instrucción a ejecutar.

## Registro de Instrucciones RI

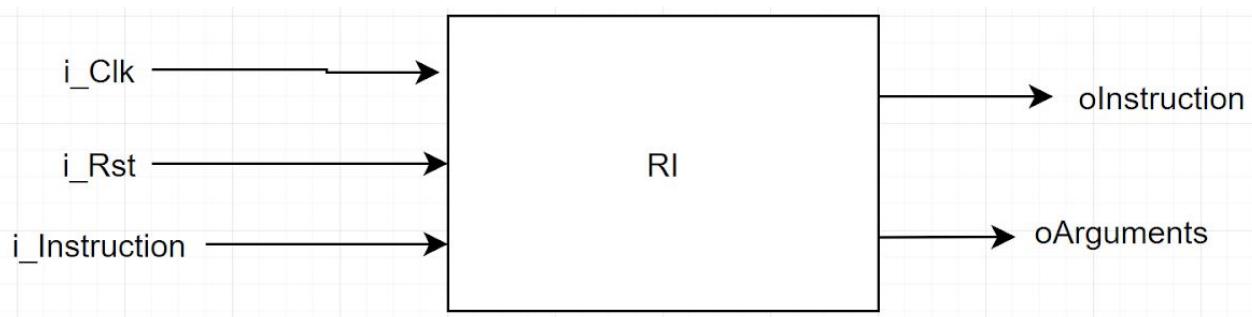


Ilustración 15 diagrama registro de instrucciones

## Entradas

| Nombre de la señal | Tamaño (Bits) | Descripción   |
|--------------------|---------------|---|
| i_Instruction      | 9             | Bus de instrucciones de 9 bits que se encarga de mandar llamar las instrucciones desde el set de instrucciones.     |
| i_Clk              | 1             | Señal de reloj que genera pulsos en intervalos constantes, los cuales se encargan de sincronizar las instrucciones. |
| i_Rst              | 1             | Señal que regresa al microprocesador a sus valores iniciales o por defecto.   |

Ilustración 16 tabla de entradas con descripción

## Salidas

| Nombre de la señal | Tamaño (Bits) | Descripción   |
|--------------------|---------------|---|
| oArguments         | 6             | señal con 6 bits por ambos argumentos que existen   |
| oInstruction       | 9             | Bus de instrucciones de 9 bits que se encarga de mandar llamar las instrucciones desde el set de instrucciones. |

Ilustración 17 tabla de salidas con descripción

## Descripción

El registro de instrucciones se encarga de recibir la instrucción del set de instrucciones para almacenar la instrucción que se está ejecutando. El PC activa el movimiento de datos después de procesar la instrucción.

## Registro de Datos RD

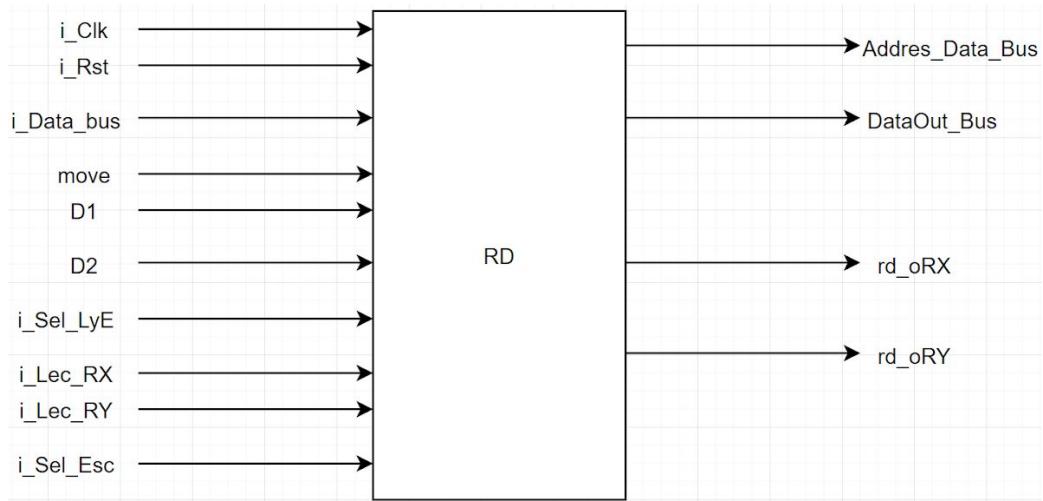


Ilustración 18 diagrama registro de datos

## Entradas

| Nombre de la entrada | Tamaño (Bits) | Descripción   |
|----------------------|---------------|---|
| .i_Data_bus          | 8             | Bus de datos de entrada de 8 bits que manda llamar a los datos almacenados en la memoria.                           |
| i_Clk                | 1             | Señal de reloj que genera pulsos en intervalos constantes, los cuales se encargan de sincronizar las instrucciones. |
| i_Rst                | 1             | Señal que regresa al microprocesador a sus valores iniciales o por defecto.   |
| i_Sel_Esc            | 1             | Seleccionador de opción escritura   |
| move                 | 1             | Mover una dato del registro Y al registro X   |
| D1                   | 3             | Dirección 1 a la que se hará el movimiento de la entrada move   |
| D2                   | 3             | Dirección 2 a la que se hará el movimiento de la entrada move   |
| i_Sel_LyE            | 1             | Entrada seleccionadora de lectura y escritura   |
| i_Lec_RX             | 3             | Entrada de lectura del dato en el registro X  |
| i_Lec_RY             | 3             | Entrada de lectura dell dato en el registro Y   |

Ilustración 19 tabla de entradas con descripción

## Salidas

| Nombre de la salida | Tamaño (Bits) | Descripción   |
|---------------------|---------------|---|
| .Address_Data_Bus   | 8             | Bus de datos de 8 bits que se encarga de avisar a nuestra memoria de datos, los datos que necesita el microprocesador para poder realizar alguna instrucción. |
| rd_oRX              | 3             | Contiene el primer dato almacenado para ser operado por la ALU.   |
| rd_oRY              | 3             | Contiene el segundo dato almacenado para ser operado por la ALU.  |
| DataOut_Bus         | 8             | Bus de datos de salida de 8 bits que lleva los resultados obtenidos en el proceso a la memoria de datos.  |

Ilustración 20 tabla de salidas con descripción

## Descripción

El registro de datos es una parte de memoria que almacena los datos que recibe el microprocesador, contiene los datos copiados desde la memoria que se va a escribir en una dirección de memoria concreta (lectura-escritura).

En ellos se almacenan los datos u operandos que intervienen en una instrucción, antes de la realización de las operaciones.

## Unidad Aritmética Lógica ALU

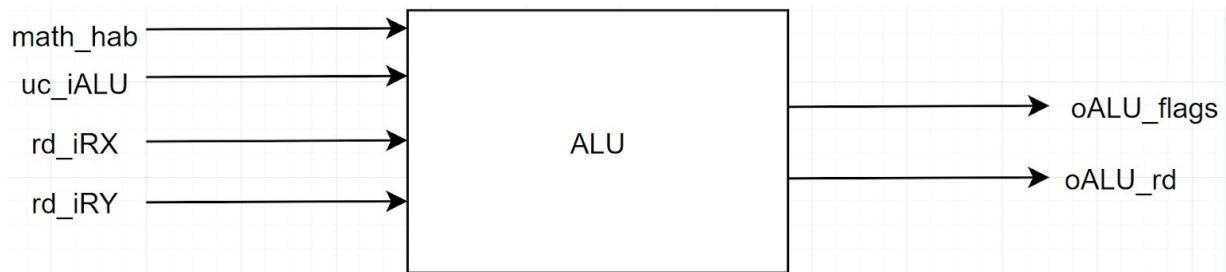


Ilustración 21 diagrama unidad aritmética lógica

## Entradas

| Nombre de la señal | Tamaño (Bits) | Descripción   |
|--------------------|---------------|---|
| .uc_iALU           | 3             | Bus de datos que envía los datos con los que trabajara ALU (operaciones lógicas y matemáticas). |
| .rd_iRX            | 3             | Contiene el primer dato almacenado para ser operado por la ALU.                                 |
| .rd_iRY            | 3             | Contiene el segundo dato almacenado para ser operado por la ALU.                                |
| math_hab           | 1             | Cuando ya se decodifica en la UC esta manda a la ALU el habilitador en la que en efecto es math |

Ilustración 22 tabla de entradas con descripción

## Salidas

| Nombre de la señal | Tamaño (Bits) | Descripción   |
|--------------------|---------------|---|
| .oALU_flags        | 3             | Cambio de estado que indica a las banderas que la ALU realizó alguna operación.                   |
| .oALU_rd           | 8             | Señal que contiene el dato almacenado de la operación a RD para esperar la siguiente instrucción. |

Ilustración 23 tabla de salidas con descripción

## Descripción

Unidad aritmética-lógica es la encargada de realizar las operaciones, las cuales son suma, resta, multiplicación y división además de también trabajar con compuertas lógicas( and, or, not)

## Implementación y Simulación

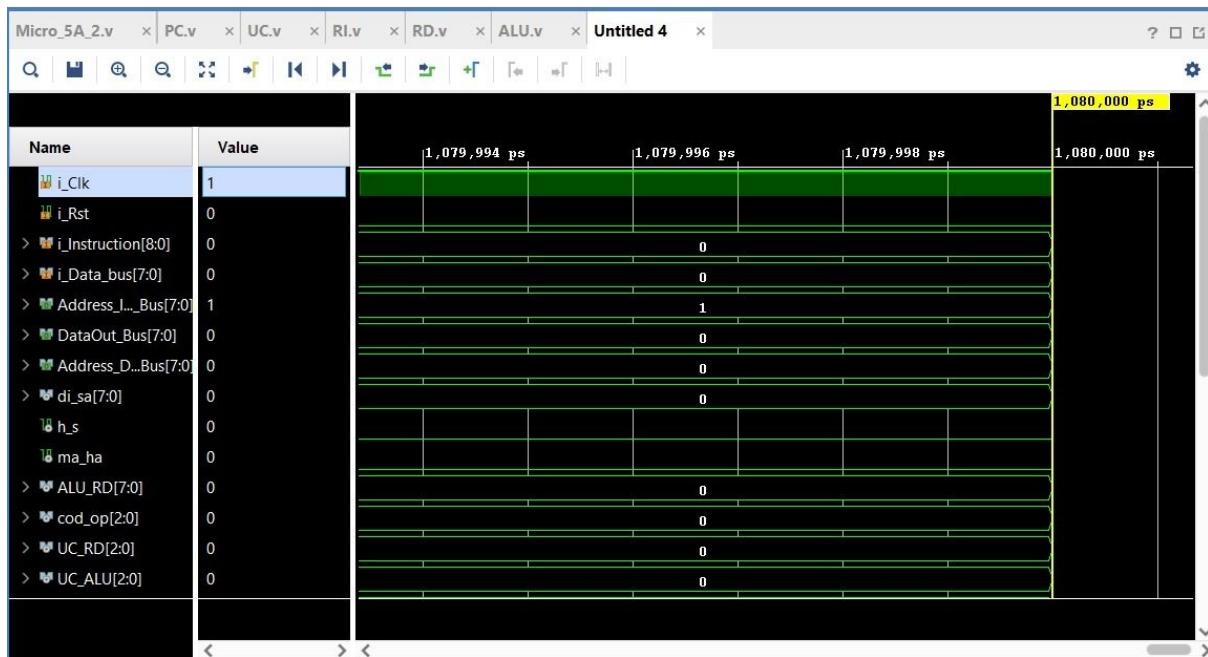


Ilustración 24 Simulación con instrucción 000000000

## Conclusiones

La práctica nos resultó bastante compleja, nos hizo darnos cuenta de las lagunas que tenemos en la materia anterior (electrónica digital). No se entregó a tiempo por las dificultades que presentó el equipo, se mejoró la arquitectura del procesador, aprendiendo poco a poco como funcionaba la codificación, como funciona cada uno de los bloques internos y lo importante que es la planificación con tiempo. Al final logramos obtener un microprocesador de 8 bits funcional. Nos facilitó bastante la metodología que el profesor nos recomendó (top-down) pudiendo comprender de donde surgía cada uno de los componentes.

Conocimos la manera en que se lleva a cabo la ejecución de las instrucciones, los modos de direccionamiento de datos y las arquitecturas. Al momento de que cada uno de los equipos diseñar su microprocesador pudimos trabajar en equipo y explorar más a fondo cómo funcionan los bloques internos.

## Referencias

- <https://sites.google.com/site/arquitecturadecomputadorass4a/home/unidad-1>
- <https://www.profesionalreview.com/2020/01/25/partes-de-un-procesador-cpu/>
- [https://app.diagrams.net/#G1-859EAcP\\_QYZmXMbQFbHlrOPmO-bhJmu](https://app.diagrams.net/#G1-859EAcP_QYZmXMbQFbHlrOPmO-bhJmu)
- <https://microcontroladoressesv.wordpress.com/arquitectura-de-los-microcontroladores/>

# Apéndice

## Archivo TOP Microprocesador\_1

```
1 module Micro_5A_2
2     ( input i_Clk,
3       input i_Rst,
4       input [8:0] i_Instruction,
5       input [7:0] i_Data_bus,
6       output [7:0] Address_Instruction_Bus,
7       output [7:0] DataOut_Bus,
8       output [7:0] Address_Data_Bus);
9
10
11    wire [7:0] di_sa;
12    wire h_s;
13    wire ma_ha;
14    wire [7:0] ALU_RD;
15    wire [2:0] cod_op;
16    wire [2:0] UC_RD;
17    wire [2:0] UC_ALU;
18    wire [7:0] RD_RX;
19    wire [7:0] RD_RY;
20    wire [1:0] Flags;
21    wire [1:0] arguments;
22
23    UC Bloque_1(
24        .i_Clk(i_Clk),
25        .i_Rst(i_Rst),
26        .iArguments(arguments),
27        .iFlags(Flags),
28        .iCod_op(cod_op),
29        .omath_hab(ma_ha),
30        .oh_s(h_s),
31        .omove(mov),
32        .oD1(d1),
33        .oD2(d2),
34        .o_sel_LyE(s_LyE),
35        .o_Lec_RX(l_RX),
36        .o_Lec_RY(l_RY),
37        .o_Sel_Esc(s_esc),
38        .odir_sal(di_sa)
39    );
40
41    PC Bloque_2(
42        .i_Clk(i_Clk),
43        .i_Rst(i_Rst),
44        .Address_Instruction_Bus(Address_Instruction_Bus),
```

```

45      .direc_sal(di_sa),
46      .hab_sal(h_s)
47  );
48
49  RI Bloque_3(
50      .i_Clk(i_Clk),
51      .i_Rst(i_Rst),
52      .i_Instruction(i_Instruction),
53      .oInstruction(cod_op),
54      .oArguments(arguments)
55  );
56
57  RD Bloque_4(
58      .i_Clk(i_Clk),
59      .i_Rst(i_Rst),
60      .i_Data_bus(i_Data_bus),
61      .move(mov),
62      .D1(d1),
63      .D2(d2),
64      .i_sel_LyE(s_LyE),
65      .i_Lec_RX(l_RX),
66      .i_Lec_RY(l_RY),
67      .i_Sel_Esc(s_esc),
68      .rd_oRX(RD_RX),
69      .rd_oRY(RD_RY),
70      .DataOut_Bus(DataOut_Bus),
71      .Address_Data_Bus(Address_Data_Bus)
72  );
73
74  ALU Bloque_5(
75      .uc_iALU(UC_ALU),
76      .rd_iRX(RD_RX),
77      .rd_iRY(RD_RY),
78      .math_hab(ma_ha),
79      .oALU_flags(Flags),
80      .oALU_rd(ALU_RD)
81  );
82
83 endmodule

```

## Unidad de Control UC

```
1 module UC(
2     input i_Clk,
3     input i_Rst,
4     input [5:0] iArguments,
5     input [1:0] iFlags,
6     input [2:0] iCod_op,
7     output [2:0] uc_iALU,
8     output omath_hab,
9     output oh_s,
10    output omove,
11    output [2:0]oD1,
12    output [2:0]oD2,
13    output o_sel_LyE,
14    output [2:0]o_Lec_RX,
15    output [2:0]o_Lec_RY,
16    output [2:0]o_Sel_Esc,
17    output [7:0] odir_sal
18 );
19 reg [2:0] op=0;
20 reg [7:0] dir_sal=0;
21 reg [2:0] Lec_RX=0;
22 reg [2:0] Lec_RY=0;
23 reg [2:0] D1=0;
24 reg [2:0] D2=0;
25 reg [2:0] Sel_Esc=0;
26 reg sel_LyE=0;
27 reg math_hab=0;
28 reg h_s=0;
29 reg move=0;
30
31 always @*
32 begin
33     if(i_Rst) begin//Reset asincrono
34         dir_sal<=8'b00000000;
35         Lec_RX <= 3'b000;
36         Lec_RY <= 3'b000;
37         D1 <= 3'b000;
38         D2 <= 3'b000;
39         Sel_Esc <= 3'b000;
40         sel_LyE <= 0;
41         h_s <= 0;
42         move <= 0;
43         math_hab<=0;
44     end
45     else begin
46         case(iCod_op)
47             3'b000: begin
48                 //LOAD_1
49                 Lec_RX <= iArguments[5:3];
50                 Sel_Esc <= iArguments[2:0];
51                 sel_LyE <= 1;
52
53                 dir_sal<=8'b00000000;
54                 Lec_RY <= 3'b000;
55                 D1 <= 3'b000;
56                 D2 <= 3'b000;
57                 h_s <= 0;
58                 move <= 0;
59                 math_hab<=0;
60
61             end
62         endcase
63         case(iCod_op)
64             3'b001: begin
65                 //LOAD_2
66                 Lec_RY <= iArguments[2:0];
67                 Lec_RX <= iArguments[5:3];
68                 sel_LyE <= 0;
69
70                 dir_sal<=8'b00000000;
71                 Sel_Esc <= 3'b000;
72                 D1 <= 3'b000;
73                 D2 <= 3'b000;
74                 h_s <= 0;
75                 move <= 0;
76                 math_hab<=0;
77
78             end
79         endcase
80         case(iCod_op)
81             3'b010: begin
82                 //STORE_1
83                 Lec_RX <= iArguments[5:3];
84                 Sel_Esc <= iArguments[2:0];
85                 sel_LyE <= 0;
86
87                 dir_sal<=8'b00000000;
88                 Lec_RY <= 3'b000;
```

```

89      D1 <= 3'b000;
90      D2 <= 3'b000;
91      h_s <= 0;                                133          //MATH
92      move <= 0;                               134          op <= iArguments[2:0];
93      math_hab<=0;                            135          math_hab<=1;
94
95      end                                         137          dir_sal<=8'b00000000;
96      endcase                                     138          Lec_RX <= 3'b000;
97      case(iCod_op)                           139          Lec_RY <= 3'b000;
98      3'b011: begin                           140          D1 <= 3'b000;
99          //STORE_Z                           141          D2 <= 3'b000;
100         Lec_RY <= iArguments[2:0];           142          Sel_Esc <= 3'b000;
101         Lec_RX <= iArguments[5:3];           143          sel_LyE <= 0;
102         sel_LyE <= 1;                         144          h_s <= 0;
103
104         dir_sal<=8'b00000000;                145          move <= 0;
105         Sel_Esc <= 3'b000;                  146          end
106         D1 <= 3'b000;                      148          endcase
107         D2 <= 3'b000;                      149          case(iCod_op)
108         h_s <= 0;                          150          3'b110: begin
109         move <= 0;                        151          //JUMP
110         math_hab<=0;                     152          dir_sal<=iArguments[2:0];
111
112         end                                         153          h_s <= 1;
113         endcase                                     154          Lec_RX <= 3'b000;
114         case(iCod_op)                           155          Lec_RY <= 3'b000;
115         3'b100: begin                           156          D1 <= 3'b000;
116             //MOVE                           157          D2 <= 3'b000;
117             D1 <= iArguments[5:3];           158          Sel_Esc <= 3'b000;
118             D2 <= iArguments[2:0];           159          sel_LyE <= 0;
119             move <= 1;                         160          move <= 0;
120
121             dir_sal<=8'b00000000;            162          math_hab<=0;
122             Lec_RX <= 3'b000;                163          end
123             Lec_RY <= 3'b000;                164          endcase
124             Sel_Esc <= 3'b000;              165          case(iCod_op)
125             sel_LyE <= 0;                  166          3'b111: begin
126             h_s <= 0;                      167          //No Operation
127             math_hab<=0;                 168          end
128
129             end                                         169          end
130             endcase                                     170          end
131             case(iCod_op)                           171          endcase
132             3'b101: begin                           172          end
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174

```

## Contador de Programa PC

```
1 module PC(
2     input i_Clk,
3     input i_Rst,
4     input [7:0] direc_sal,
5     input hab_sal,
6     output [7:0] Address_Instruction_Bus
7 );
8
9     // Se declaran los registros y las señales internas
10    reg [7:0] counter=0;
11
12    //Etapa de implemetacion
13    always@ (posedge i_Rst, posedge i_Clk)
14    begin
15        if (i_Rst) begin//El reset es asincrono
16            counter <= 8'b00000000;
17        end
18        else begin
19            if (hab_sal==1) begin
20                counter<- direc_sal;
21            end
22            else begin
23                counter<=counter+1;
24            end
25        end
26    end
27
28    assign Address_Instruction_Bus = counter;
29 endmodule
```

## Registro de Instrucciones RI

```
1 module RI(
2     input i_Clk,
3     input i_Rst,
4     input [8:0] i_Instruction,
5     output [2:0] oInstruction,
6     output [5:0] oArguments,
7     output [7:0] o_Direccionamiento_inmediato);
8
9     reg [2:0] cod_op;
10    reg [5:0] Arguments;
11    reg [7:0] Direccionamiento_inmediato=0;
12
13    always@(posedge i_Rst, posedge i_Clk)
14    begin
15        if (i_Rst) begin
16            cod_op <= 3'b000;
17            Arguments <= 6'b000000;
18        end
19        else begin
20            if (i_Instruction[8:6]==3'b000 || i_Instruction[8:6]==3'b010) begin
21                Direccionamiento_inmediato<=i_Instruction[2:0];
22                cod_op<=i_Instruction[8:6];
23                Arguments[5:3]<=i_Instruction[5:3];
24                Arguments[2:0]<=& 3'b000;
25            end
26            else begin
27                cod_op<-i_Instruction[8:6];
28                Arguments[5:3]<=i_Instruction[5:3];
29                Arguments[2:0]<=i_Instruction[2:0];
30                Direccionamiento_inmediato<=0;
31            end
32        end
33    end
34
35    assign oInstruction = cod_op;
36    assign oArguments = Arguments;
37    assign o_Direccionamiento_inmediato = Direccionamiento_inmediato;
38
39 endmodule|
```

## Registro de Datos RD

```
1 | module RD(
2 |   input i_Clk,
3 |   input i_Rst,
4 |   input [7:0] i_Data_Bus,
5 |   input move,
6 |   input [2:0] D1,
7 |   input [2:0] D2,
8 |   input i_sel_LyE,
9 |   input [2:0] i_Lec_RX,
10 |  input [2:0] i_Lec_RY,
11 |  input [2:0] i_Sel_Esc,
12 |  output [7:0] rd_oRX,
13 |  output [7:0] rd_oRY,
14 |  output [7:0] Address_Data_Bus);
15 |
16 | reg [7:0] RX;
17 | reg [7:0] RY;
18 | reg [7:0] R[7:0];
19 |
20 | always@(posedge i_Rst, posedge i_Clk)
21 | begin
22 |   if (i_Rst) begin
23 |     R[0]<=0;
24 |     R[1]<=0;
25 |     R[2]<=0;
26 |     R[3]<=0;
27 |     R[4]<=0;
28 |     R[5]<=0;
29 |     R[6]<=0;
30 |     R[7]<=0;
31 |   end
32 |   else begin
33 |     if(move) begin
34 |       R[D1] <= R[D2];
35 |     end
36 |     else begin
37 |     end
38 |     case (i_sel_LyE)
39 |       //Lectura
40 |       1'b0: begin
41 |         case (i_Lec_RX)
42 |           3'b000: RX<=R[0];
43 |           3'b001: RX<=R[1];
44 |           3'b010: RX<=R[2];
45 |           3'b011: RX<=R[3];
46 |           3'b100: RX<=R[4];
47 |           3'b101: RX<=R[5];
48 |           3'b110: RX<=R[6];
49 |           3'b111: RX<=R[7];
50 |           default: RX<=8'b00000000;
51 |         endcase
52 |       end
53 |       1'b1: begin
54 |         case (i_Sel_Esc)
55 |           3'b000: R[0]<=i_Data_Bus;
56 |           3'b001: R[1]<=i_Data_Bus;
57 |           3'b010: R[2]<=i_Data_Bus;
58 |           3'b011: R[3]<=i_Data_Bus;
59 |           3'b100: R[4]<=i_Data_Bus;
60 |           3'b101: R[5]<=i_Data_Bus;
61 |           3'b110: R[6]<=i_Data_Bus;
62 |           3'b111: R[7]<=i_Data_Bus;
63 |           default: R[7]<=i_Data_Bus;
64 |         endcase
65 |       end
66 |     endcase
67 |   end
68 |   assign rd_oRX = RX;
69 |   assign rd_oRY = RY;
70 | endmodule
```

## Unidad Aritmética Lógica ALU

```
1  module ALU(
2      input [2:0] uc_iALU,
3      input [7:0] rd_iRX,
4      input [7:0] rd_iRY,
5      input math_hab,
6      output [2:0] oALU_flags,
7      output [7:0] oALU_rd
8  );
9      reg [7:0] Res=0;
10     always@(*)
11         begin
12             case(math_hab)
13                 1'b0: Res = rd_iRY;
14                 1'b1:
15                     case(uc_iALU)
16                         3'b000: Res <= rd_iRY + rd_iRX;
17                         3'b001: Res <= rd_iRY - rd_iRX;
18                         3'b010: Res <= rd_iRY << rd_iRX;
19                         3'b011: Res <= rd_iRY >> rd_iRX;
20                         3'b100: Res <= ~rd_iRX;
21                         3'b101: Res <= rd_iRY & rd_iRX;
22                         3'b110: Res <= rd_iRY | rd_iRX;
23                         3'b111: Res <= rd_iRY ^ rd_iRX;
24                     endcase
25             endcase
26         end
27         assign oALU_rd = Res;
28         assign oALU_flags[0] = &(~Res);
29         assign oALU_flags[1] = Res[6];
30         assign oALU_flags[2] = Res[7];
31     endmodule
```